

8

ИИКТ - БАН

eISSN: 2367-8666

Лекции по компютърни науки и технологии

Научни изчисления с Java и Android

Практическо ръководство

Тодор Балабанов Илиян Занкински Петър Томов

eISBN: 978-619-7320-05-3

Поредицата „Лекции по компютърни науки и технологии на Института по информационни и комуникационни технологии при Българската академия на науките“ публикува в електронен вид учебници и учебни помагала, предназначени за студенти и докторанти по различни програми по информатика, изчислителна математика, математическо моделиране, комуникационни технологии, и др., както и за всички читатели, интересувачи се от тези научни области. Учебниците се базират върху курсове лекции, водени от учени на Института по информационни и комуникационни технологии – БАН в различни български университети и в Центъра за обучение на докторанти в БАН. Публикуваните материали са с отворен достъп - те са свободно достъпни без заплащане.

## Редактори

Геннадий Агре (Главен редактор) – ИИКТ-БАН  
e-mail: [agre@iinf.bas.bg](mailto:agre@iinf.bas.bg)

Вера Ангелова – ИИКТ-БАН  
e-mail: [vangelova@iit.bas.bg](mailto:vangelova@iit.bas.bg)

Пенчо Маринов – ИИКТ-БАН  
e-mail: [pencho@bas.bg](mailto:pencho@bas.bg)

eISSN: 2367-8666

*Настоящото издание е обект на авторско право. Всички права са запазени при превод, разпечатване, използване на илюстрации, цитирания, разпространение, възпроизвеждане на микрофилми или по други начини, както и съхранение в бази от данни на всички или част от материалите в настоящето издание. Копирането на изданието или на част от съдържанието му е разрешено само със съгласието на авторите и/или редакторите*

# Теми

Предговор	1
1 Научни изчисления	3
2 Евристики за прогнозиране	6
3 Софтуерна архитектура	13
4 Android активен тапет	18
5 Централизиран сървър	104
6 Комуникация клиент-сървър	166
7 Обучение на мрежата	183
Заклучение	207
Библиография	208
Списък на фигурите	212
Азбучен указател	213

# Съдържание

<b>Предговор</b>	<b>1</b>
<b>1 Научни изчисления</b>	<b>3</b>
1.1 Последователно програмиране . . . . .	3
1.2 Паралелно програмиране . . . . .	4
1.3 Супер компютри и грид изчисления . . . . .	4
1.4 Изчисления в разпределена среда . . . . .	5
1.5 Дарена изчислителна мощност . . . . .	5
<b>2 Евристики за прогнозиране</b>	<b>6</b>
2.1 Изкуствени невронни мрежи . . . . .	6
2.2 Генетични алгоритми . . . . .	8
2.3 Финансови времеви редове . . . . .	9
2.4 Прогнозиране в разпределена среда . . . . .	10
2.5 Фонови пресмятания върху мобилни устройства . . . . .	11
<b>3 Софтуерна архитектура</b>	<b>13</b>
3.1 Избор на развойни средства . . . . .	13
3.1.1 От страна на сървъра . . . . .	13
3.1.2 От страна на клиента . . . . .	14
3.1.3 За комуникация между сървъра и клиента . . . . .	14
3.2 Компоненти на системата . . . . .	14
3.2.1 Подход за разработка . . . . .	15
3.3 Лиценз и хранилище за проекта . . . . .	16
3.3.1 Лиценз . . . . .	16
3.3.2 Хранилище за програмен код . . . . .	16
<b>4 Android активен тапет</b>	<b>18</b>
4.1 Манифест файл на мобилното приложение . . . . .	18
4.2 Екран с настройки . . . . .	25
4.2.1 Описание на потребителския интерфейс под формата на XML файлове . . . . .	28
4.2.2 Програмен код за управление на интерфейса . . . . .	36
4.3 Пресмятане на фонов режим . . . . .	38
4.4 Двигател на услугата . . . . .	54
4.5 Представяне на информацията върху локалното устройство . . . . .	86
<b>5 Централизиран сървър</b>	<b>104</b>
5.1 Релационна база данни . . . . .	105
5.2 Алгоритмична обработка на суровите данни . . . . .	113
5.3 Сървър скриптове . . . . .	120
5.3.1 Зареждане на мрежа по идентификатор . . . . .	124
5.3.2 Зареждане на случайно избрана мрежа . . . . .	131
5.3.3 Зареждане на най-добрата жизненост от глобалната популация . . . . .	132
5.3.4 Зареждане на брой неврони по идентификатор . . . . .	135

5.3.5	Зареждане на тренировъчно множество по информация за валутна двойка . . . . .	137
5.3.6	Зареждане на брой екземпляри по идентификатор или информация за валутна двойка . . . . .	143
5.3.7	Зареждане на брой тренировъчни примери по информация за валутна двойка . . . . .	149
5.3.8	Съхраняване на екземпляр изкуствена невронна мрежа . . . . .	152
5.3.9	Съхраняване на екземпляр изкуствена невронна мрежа след дообучение . . . . .	158
5.3.10	Съхраняване на тренировъчно множество . . . . .	159
<b>6</b>	<b>Комуникация клиент-сървър</b>	<b>166</b>
6.1	Hypertext Transfer Protocol . . . . .	166
6.2	JavaScript Object Notation . . . . .	175
<b>7</b>	<b>Обучение на мрежата</b>	<b>183</b>
7.1	Обратно разпространение на грешката . . . . .	183
7.2	Генетични алгоритми . . . . .	188
7.2.1	Кодиране на хромозомите . . . . .	194
7.2.2	Случайна равномерна мутация . . . . .	203
	<b>Заклучение</b>	<b>207</b>
	<b>Библиография</b>	<b>208</b>
	<b>Списък на фигурите</b>	<b>212</b>
	<b>Азбучен указател</b>	<b>213</b>

# Предговор

Това учебно помагало е предназначено за ученици и студенти, които биха искали да се запознаят с възможностите за реализиране на изчисления в разпределена среда с използването на програмния език Java и мобилната платформа Android.

В съвременното ни ежедневие ние все по-често сме заобиколени от мобилни изчислителни устройства. Най-често това са мобилни телефони, таблети, часовници или други форми на wearables (миниатюрна електроника под формата на модни аксесоари или части от дрехи). До преди десетилетие този вид мобилни устройства бяха рядкост, а изчислителните им възможности бяха изключително ограничени. Тези два факта не позволяваха мобилните устройства да бъдат използвани за нещо повече освен основния принцип на употреба за който са създадени. С развитието на микроелектрониката и миниатюризацията на компонентите, съставляващи мобилните устройства, техните възможности значително нараснаха за последното десетилетие. Това позволява върху този вид устройства да се извършват и допълнителни задачи, които не са били предвидени при първоначалното им проектиране. Паралелно с развитието на мобилните технологии бурен подем претърпяха и възможностите за мобилна комуникация като GSM, 3G, 4G, Wi-Fi, Bluetooth, NFC и други. Комбинацията между относително мощни мобилни изчислителни устройства и добре развита комуникационна среда открива безгранични възможности за приложение на мобилните устройства при извършването на допълнителни изчисления в разпределена среда.

В настоящето учебно помагало запознаваме читателите с интересните възможности, които предлагат съвременните Android мобилни устройства, за постигането на резултати в научни изчислителни задачи, разпределяйки изчисленията върху физически отдалечени едно от друго устройства. Изложението на материала е организирано в следните глави.

Глава 1 - Научни изчисления: Дава кратко описание на идеята за научни изследвания и по какъв начин те могат да се впишат в ежедневния живот и използването на „умни“ мобилни устройства.

Глава 2 - Евристики за прогнозиране: Представят се някои от популярните евристични подходи за решаване на сложни изчислителни задачи. Обяснява се и разликата между точните числени методи и приближените изчисления. По-сериозно внимание се отделя на генетичните алгоритми и изкуствените невронни мрежи, тъй като те са в основата на разработваната система.

Глава 3 - Софтуерна архитектура: Излагат се различни аргументи за изготвянето на една софтуерна архитектура и се дават насоки за реалната разработка на един софтуерен проект. Изборът за архитектура на системата е насочен към трислойните модели, както в контекста на системата клиент-сървър, така и в контекста на локално мобилно приложение, което се състои от потребителски интерфейс, работна логика и локална база данни.

Глава 4 - Android активен тапет: Детайлно се представя процесът по създаването на едно мобилно приложение, което има за основна задача извършване на изчисления във фонов режим чрез използване на възможностите, които операционната система Android дава. Това включва приложение за активен тапет, екран за настройки и група класове за вътрешно представяне на информацията.

Глава 5 - Централизиран сървър: Показва реализацията на отдалечената част от системата, а именно веб базиран сървър с релационна база данни. За съхраняването на суровата информация за валутните котировки и параметрите на изкуствените невронни мрежи е представено MySQL базирано решение

под формата на релационна база данни. За обмен на информацията между отдалечената база данни и мобилните приложения е представено РНР базирано решение.

Глава 6 - Комуникация клиент-сървър: Набляга на комуникацията между отдалечения сървър и локалните клиенти, изпълнявани върху мобилни устройства. Тъй като от страната на сървъра е избрано уеб базирано решение, то в основата на комуникацията е заложен HTTP протоколът и неговите възможности да предава JSON пакетирани съобщения.

Глава 7 - Обучение на мрежата: Разглежда обучението на изкуствени невронни мрежи, което е свързано с намиране на такива оптимални стойности за теглата, че мрежата максимално добре да изпълнява задачата, за която е предназначена. При липсата на обратни връзки се използва обратно разпространение на грешката, а при наличието на обратни връзки генетични алгоритми.

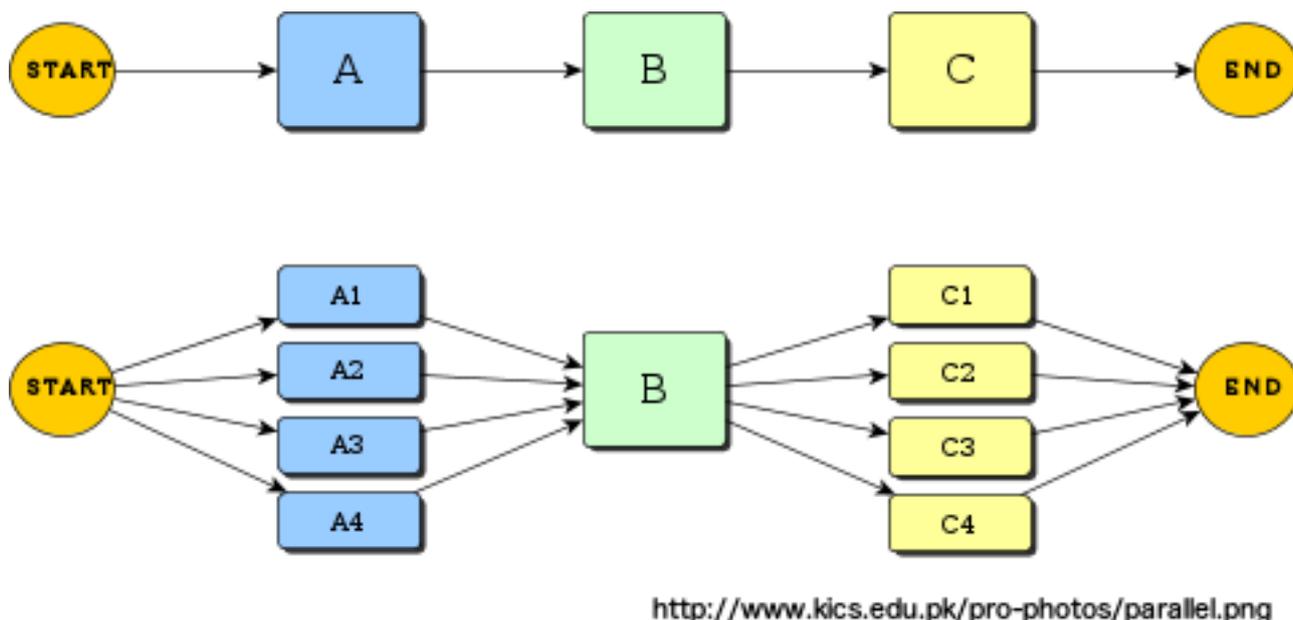
## Глава 1

# Научни изчисления

Още в зората на съвременната изчислителна техника най-съществените пресмятания са били с научна насоченост и военно дело. Този факт не се е променил значително за последните десетилетия. Дори в наши дни най-сериозните изчислителни ресурси са насочени в областта на науката. Това дава основание да обърнем значително внимание на начините, по които можем да изпълняваме научни изчисления дори и върху изчислителни устройства, чието основно предназначение не е с научна цел.

### 1.1 Последователно програмиране

При последователното програмиране всяка изчислителна инструкция следва всички предходни. В зората на изчислителната техника пресмятанията са извършвани по този начин. Дори в наши дни значителна част от алгоритмите се изпълняват само последователно, тъй като входните данни за всяка инструкция зависят от изходните данни на предходните инструкции. Последователните алгоритми не подлежат на декомпозиране и поради тази причина са неприложими за паралелни пресмятания (Фиг. 1.1).

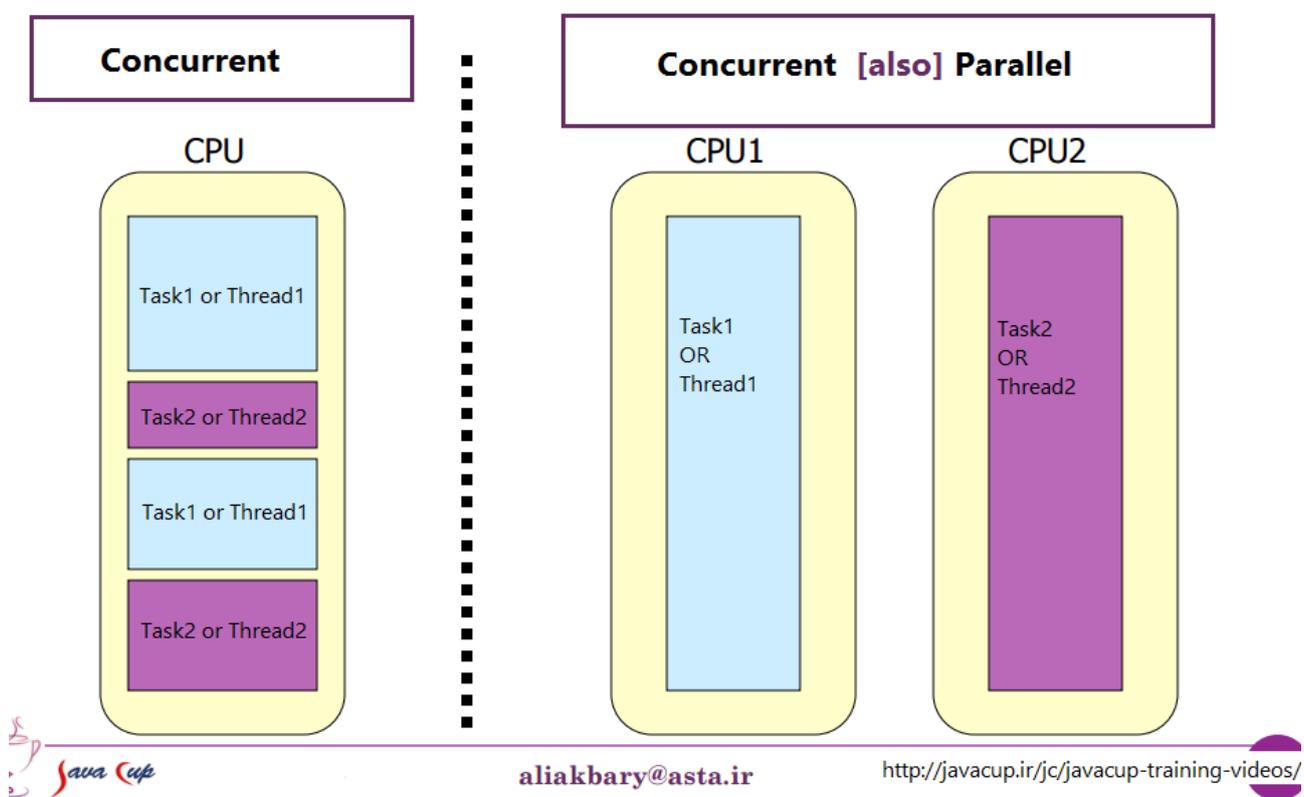


Фигура 1.1: Сравнение между последователни пресмятания и паралелни пресмятания

## 1.2 Паралелно програмиране

При паралелното програмиране основно се говори за две разновидности - конкурентни пресмятания и паралелни пресмятания. Конкурентните пресмятания са в среда, където група от задачи могат да се пресметнат едновременно без да има значение от реда на пресмятане. В същото време, паралелните пресмятания се отнасят за едновременно пресмятане на отделни задачи върху отделни процесори. В този контекст всички паралелни пресмятания са конкурентни пресмятания, но не и обратното (Фиг. 1.2).

### Concurrency & Parallelism



Фигура 1.2: Сравнение между конкурентни пресмятания и паралелни пресмятания

## 1.3 Супер компютри и грид изчисления

Когато паралелните алгоритми се изпълняват на изчислителни машини с множество процесори и/или множество ядра на процесорите, този вид изчисления се определят като супер компютърни (supercomputing). Същественото при този вид пресмятания е, че се използва много бърза вътрешна шина (понякога оптична) и споделена оперативна памет. За разлика от супер компютрите, грид изчисленията се осъществяват на множество машини свързани в обща мрежа, но работещи автономно без да споделят обща памет. При грид системите отделните изчислителни машини могат да са териториално отдалечени една от друга. Съществено е да се отбележи, че и при супер компютрите собственикът на системата има пълен контрол над нея. Това може малко да се различава за грид системите, ако към грида са включени компютри под чужд контрол.

## 1.4 Изчисления в разпределена среда

Преходът от грид системите към системи за изчисления в разпределена среда се състои в това, че изчислителните машини в разпределената среда са абсолютно автономни. Тези машини не споделят общи ресурси като процесор или оперативна памет. Много характерно е в разпределената среда контролът над изчислителните машини да не е от страната на организиращия изчисленията. Това води до два основни проблема - ненадеждна (често и твърде бавна) комуникация, липса на гаранция за коректност на пресмятанията (манипулации от страна на притежаващия изчислителните ресурси). Също така, при грид системите често се наблюдава хомогенност на изчислителните ресурси по отношение на хардуерни конфигурации и операционна система, докато в разпределената среда изчислителните машини са основно хетерогенни, което може да води до големи разлики в хардуера и операционните системи.

## 1.5 Дарена изчислителна мощност

За решаването на някои по-мащабни научни проблеми изчислителната мощност и/или финансовите ресурси често са недостатъчни. В такива ситуации не малко научни институции прибягват до така наречените дарени изчислителни ресурси в разпределена среда. Един от най-изчерпателните списъци с проекти от този вид може да бъде открит в уеб сайта Distributed Computing Info [1]. Съществено е да се отбележи, че не всеки изчислителен проблем е подходящ за решаване в разпределена среда с дарена изчислителна мощ. На първо място проблемът трябва да подлежи на декомпозиране, така че отделни части от него да се пресмятат едновременно. Второто важно нещо е да не е от съществено значение в кой момент от времето и в какъв ред ще бъдат получени пресметнатите резултати. И третото съществено нещо е да е наличен механизъм за проверка на достоверността от пресмятанията, тъй като изчисленията се извършват на машини с различна хардуерна конфигурация и различни операционни системи, а освен това са възможни манипулации от страна на хората притежаващи тези машини. Най-известният проект за дарена изчислителна мощ е SETI@home [2] като неговата цел е да търси сигнали от космоса, които да са създадени от интелигентни форми на живот.

Когато изчисленията се извършват на мобилни устройства, то разпределената среда се превръща в мобилна среда за разпределени изчисления. В останалата част от това учебно помагало ще бъде представено точно изграждането на система за извършване на разпределени изчисления върху мобилни устройства.

## Глава 2

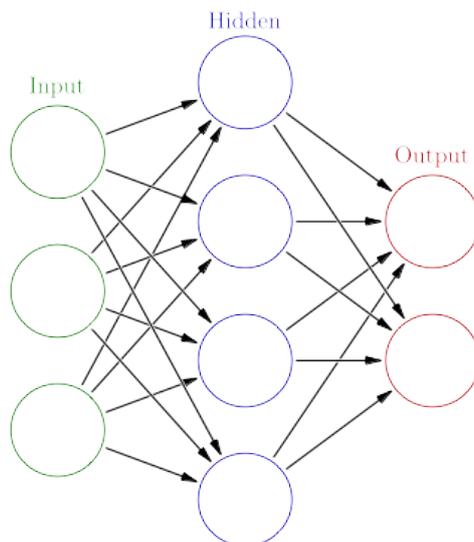
# Евристики за прогнозиране

Евристичните алгоритми са подход за решаване на изчислителни проблеми, основаващи се на опита и интуицията. Този вид алгоритми не гарантират оптимално решение за поставената задача. Евристичните алгоритми намират широко приложение при задачи, които трудно се поддават на точни числени методи или аналитични решения. Макар и да не могат да предложат оптимално решение, евристичните алгоритми често водят до достатъчно приемливи в практиката, близки до оптималното решения. Благодарение на своята не детерминистична природа, евристичните алгоритми са изключително подходящи за реализация в супер компютърни изчисления. Две изключително популярни евристики са изкуствените невронни мрежи и генетичните алгоритми. Те са обект на особена популярност през последните две десетилетия и това дава основание да бъдат заложени в основното изложение на настоящото учебно помагало. Сами по себе си евристиките са безполезни, ако не бъдат приложени върху достатъчно сложна изчислителна задача. Точно такава сложност предлагат задачите за прогнозиране. Прогнозирането е залегнало в множество дейности от човешкото ежедневие като започнем от средните дневни температури и стигнем до потреблението на определени стоки и услуги. Под една или друга форма почти всяка човешка дейност бива остойностена в термините на финансовите ресурси. Този факт дава основание да бъдат разгледани точно прогнози за промяната в цените на различни финансови инструменти. Отчитането на промяната в цената на определени интервали (равни или неравни) води до представяне на информацията във времеви ред. При времевите редове по абсцисната ос се означава времето, а по ординатната ос стойността на измерваната величина (в конкретния случай цената).

### 2.1 Изкуствени невронни мрежи

Изкуствените невронни мрежи са се появили в следствие на опитите за изграждане на математически модел за биологичните нервни системи. Този вид системи се научават (прогресивно подобряват възможностите си) с изследването на примерни данни. Приложението им е основно при задачи, за които традиционните алгоритми не дават приемливи резултати. Най-широко приложение изкуствените невронни мрежи намират в задачи за класификация. Когато става въпрос за финансови времеви редове, са възможни две събития – повишаване на стойността или понижаване на стойността. Според информацията в отминалите периоди време, изкуствената невронна мрежа може да раздели данните в два основни класа – данни показващи промяна към повишение или данни показващи промяна към понижение.

Структурата на класическите изкуствени невронни мрежи се състои от възли, наречени изкуствени неврони и връзки, наречени тегла (Фиг. 2.1). Връзките между невроните служат за предаване на сигнал от един неврон към друг. Невроните които получават сигнали ги обработват по предварително заложено правило и след това могат на свой ред да ги разпространят към други неврони. В класическия случай невроните имат стойност (обикновено това е реално число). Връзките между невроните също са представени със стойност и според правилото за обучение точно тази стойност подлежи на промяна, така че мрежата да заучава необходимата информация. При най-използваните изкуствени невронни мрежи нев-



Фигура 2.1: Трислойна изкуствена невронна мрежа

роните са организирани в слоеве. Сигналите при този вид мрежи се разпространяват от входния слой към изходния слой, като преминават междинните слоеве. Това разпространение на сигналите се нарича пас в права посока и е характерно за режима на употреба. Освен режим на употреба изкуствените невронни мрежи работят и във втори режим, наречен режим на обучение. За последните няколко десетилетия са предложени множество начини за обучение на изкуствени невронни мрежи като най-значими резултати се получават при използването на алгоритъма за обратно разпространение на грешката. Обратното разпространение на грешката представлява точен числен метод от групата на градиентните методи, който разчита на грешката, която мрежата допуска при изпълнението на обучаващите примери. Тази допусната грешка се установява в изходния слой и след това се разпространява обратно по предходните слоеве, от където идва и названието на метода.

Според своята линейна природа, обратното разпространение на грешката трудно се поддава на реализация в термините на паралелното програмиране. Ако се погледне на теглата в една изкуствена невронна мрежа като на многомерно безкрайно пространство от реални числа, то намирането на стойности за теглата е математическа оптимизационна задача. Точните числени методи дават добри резултати при пространства с относително малка размерност, но срещат сериозни затруднения при по-големите размерности. Точно противоположно на точните числени методи, евристичните алгоритми предлагат приемливи решения в разумно време. Предимство е не само ефективността, но и значително по-големите възможности евристичните алгоритми да се реализират в термините на паралелното програмиране.

Класическите неврони реализират трансферна функция и активационна функция. Най-често използваната трансферна функция е линейната, която представлява сума от умножение на входните сигнали по теглата, които ги доставят. Най-често използваните активационни функции са сигмоидната функция (2.1) и хиперболичният тангенс (2.2).

$$z_j = \frac{1}{1 + e^{-y_j}} \quad (2.1)$$

$$z_j = \frac{e^{2y_j} - 1}{e^{2y_j} + 1} \quad (2.2)$$

Активационната функция има основна роля за нормиране на изходния сигнал. Тази нормализация е необходима тъй като за трансферната функция при различните неврони постъпват различен брой сигнали и без нормализация това би направило изходните сигнали несъизмерими. При градиентните точни числени

методи има изискване активационната функция да бъде диференцируема, нещо което не е необходимо при евристичните алгоритми.

$$y_j = \sum_{i=1} x_i * w_{ij} \quad (2.3)$$

Често използваната линейна трансферна функция (2.3) има нужда от използване на допълнително събираемо наречено „отместване“ (bias). От формална гледна точка, отместването може да се интерпретира като тегло свързващо неврона с друг неврон, който емитира единичен сигнал. В практиката за всеки слой е прието да се отделя неврон емитиращ единичен сигнал. Само в изходния слой е безсмислено такъв неврон да има, тъй като той не получава входни сигнали, а емитирането на постоянна единица в изхода не носи смислена информация за функционирането на мрежата.

От математическа гледна точка класическите невронни мрежи могат да се представят с вектор (стойностите на невроните) и матрица (стойностите на теглата между невроните). Разпространението на сигналите от входа към изхода в този случай би бил умножение на вектор с число. В настоящото учебно помагало на класическа трислойна мрежа ще се подават мащабираните стойности от изминалите времеви периоди, а на изхода ще се очаква прогнозна стойност за бъдещи времеви интервали. Тази концепция е малко по-сложна от идеята за проста квалификация от вида повишение/понижение, но е значително по-информативна, защото се очаква да дава представа за един по-дълъг времеви интервал в бъдещето. След успешно прогнозиране в коя посока ще се промени цената от съществено значение става и въпросът колко дълго ще продължи този спад/повишение.

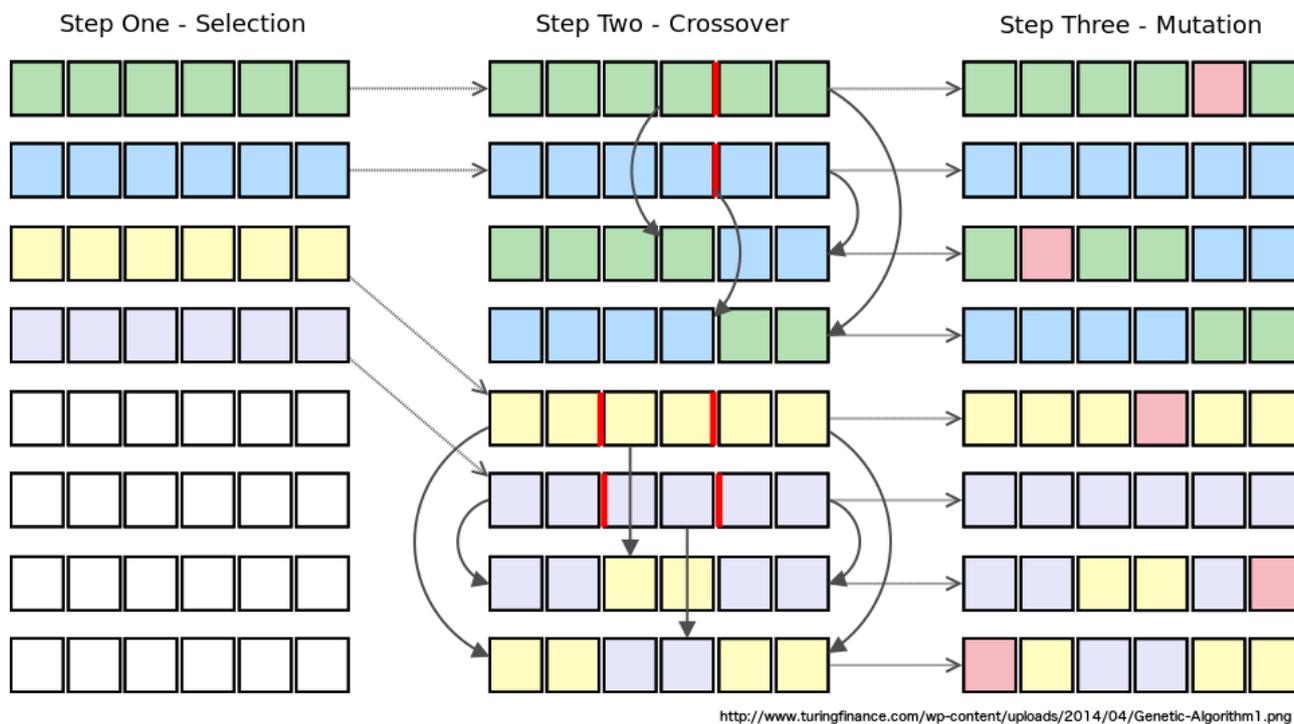
## 2.2 Генетични алгоритми

Генетичните алгоритми са глобална оптимизационна евристика вдъхновена от идеите за биологичната еволюция. Генетичните алгоритми са подмножество на класовете популационни алгоритми и еволюционни алгоритми. Основното си приложение генетичните алгоритми намират при задачи с голямо по размерност пространство на решенията. Често при такива задачи класическите точни числени методи не могат да предложат решение в приемливо време. За да се приложи генетичен алгоритъм, решението на съответната задача трябва да се представи под формата на хромозома (индивид) в обща популация от решения. След това, чрез прилагане на основните операции по селекция, кръстосване и мутация (Фиг. 2.2), отделните индивиди (решения) следва да бъдат подобрявани.

При класическия процес оптимизацията започва от случайно генерирани индивиди. Това не е задължително, особено когато става въпрос за хибридни реализации, в които генетичният алгоритъм е поддържаща оптимизация. При такива ситуации началната популация може да бъде получена в резултат на друга оптимизация или в резултат на човешка подредба. След началната фаза оптимизацията протича итеративно и приключва според предварително определени критерии за край. Най-често се използва предварително дефиниран брой поколения или брой поколения, които не водят до подобрене в намерените решения, но също така е възможно да се дефинира и интервал астрономическо време.

От основна важност за успешната работа с генетичните алгоритми е определянето на целева функция (жизнена функция на индивида), която еднозначно да определя качеството на полученото решение. На база различната жизненост която индивидите в популацията притежават, се взема стохастично решение кои индивиди да участват в създаването на бъдещото поколение и кои не. В множество реализации на генетични алгоритми се прилага правило на елита, така че най-доброто открито решение да достигне края на оптимизационния процес. В същото време правилото на елита крие риск от израждане на популацията, така че всичките решения в нея да клонят към съхранения елит.

Фактът, че генетичните алгоритми са организирани на принципа на популацията от индивиди, ясно подсказва идеалната възможност оптимизационният процес да се организира не в една глобална популация, а в множество различни локални популации, които да съществуват на различни изчислителни машини. Това от своя страна би дало възможност за реализиране на миграционни процеси, точно както



Фигура 2.2: Трите основни операции в генетичните алгоритми

това се наблюдава при естествените биологични видове.

## 2.3 Финансови времеви редове

Времевите редове са серия от замервания извършени в последователен ред във времето. Практически се получава последователност от дискретни стойности. Времевите редове могат да са съставени от стойности на равни интервали или стойности на произволни интервали. Често в практиката се случва да има липсващи замервания, което води до определени усложнения при анализирането. При финансовите времеви редове основно се използват равни интервали на отчитане и рядко има липсващи стойности при отчитането. Всяко едно отчитане при финансовите времеви редове се отличава с група стойности, характеризирани времеви интервал за който се отнася, а именно – начална стойност за интервала, най-висока постигната стойност за интервала, най-ниска стойност постигната за интервала и крайна стойност за интервала.

На Фиг. 2.3 тази информация се обозначава с вертикални черти, като долният край на вертикалната черта символизира най-ниското ниво, горният край най-високото ниво, а двете странични чертичките маркират нивото на отваряне (отляво) и нивото на затваряне (от дясно). Анализирането на времевите редове може да предостави съществена информация за лицата вземащи решения. Когато става въпрос за времеви редове във финансовата област основна цел на анализа е предлагането на прогноза. Една от най-често използваните форми на анализ е напасването на крива (curve fitting). От математическа гледна точка задачата представлява построяване на крива която да премине максимално близо до предварително определени точки. От математиката е добре известно, че през краен брой точки могат да се построят безкрайно на брой преминаващи криви. За да има смисъл от напасването кривата трябва да притежава определени свойства като изглаждане, добра интерполация и екстраполация.

Условното разделяне на финансовия времеви ред на две половини (минало и бъдеще) дава изключително добра възможност за представяне на задачата по прогнозиране в термините на изкуствените



Фигура 2.3: Отношението евро/долар в рамките на един час

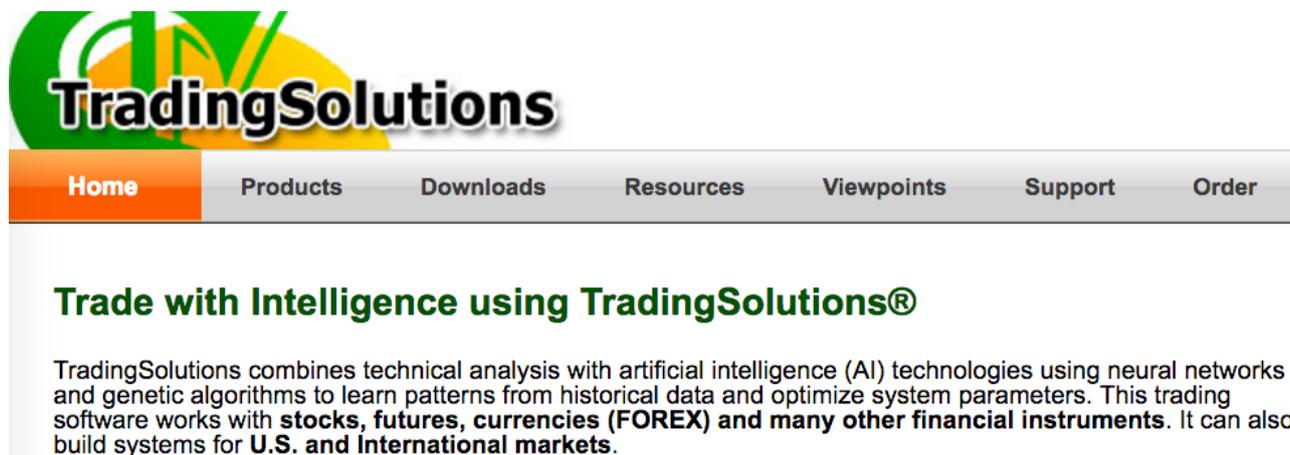
невронни мрежи. Мащабираната информация от миналия период (зелено на графиката) се подава към входния слой на изкуствената невронна мрежа, а прогнозата се получава в изходния слой на изкуствената невронна мрежа и се подлага на обратно мащабиране (червеното на графиката). Така декомпозиран времеви ред се използва при фазата за обучение на изкуствената невронна мрежа. За работната фаза на мрежата за прогноза на изкуствената невронна мрежа се подават последните измерени стойности, без да има яснота какви са бъдещите измервания.

## 2.4 Прогнозиране в разпределена среда

Гъвкавите възможности на изкуствените невронни мрежи да изградят функционална зависимост между входни и изходни данни ги прави идеален кандидат за прогнозираща система. Ако се приеме, че измерените стойности във времеви ред са точки в двуизмерно пространство, то задачата за прогнозиране може да се представи като задача за прекарване на крива през  $N$  точки (curve fitting). Ако образно се оприличи изкуствената невронна мрежа на полином, то теглата ѝ биха представлявали коефициенти в полинома. Стойностите които изкуствената невронна мрежа генерира на изхода си за бъдещи моменти от времето, са своеобразна екстраполация според апроксимираната крива. Тъй като класическите многослойни невронни мрежи работят с входни сигнали между 0.0 и 1.0 или -1.0 и +1.0, то информацията от времеви ред трябва да бъде мащабирана в съответния работен интервал на мрежата. Стойностите на времеви ред условно се разделят на минали (зелените Фиг. 2.3) и бъдещи (червените Фиг. 2.3). Изходната (прогнозна) информация на изкуствената невронна мрежа след това се мащабира обратно към оригиналните интервали на времеви ред.

От чисто практическа гледна точка, използването на изкуствени невронни мрежи и генетични алгоритми се е доказало като удачен подхода за прогнозиране на финансови инструменти, което ясно се вижда в продуктовата линия на TradingSolutions (Фиг. 2.4). Почти петнадесет години продуктовата линия на TradingSolutions доставяше системи за подпомагане вземането на решения. По настоящем, тази продуктова линия е придобита от компанията nDimensional, Inc.

Възможността да се прогнозира цените на финансови инструменти винаги е привличала не само индустрията, но и научните среди. Едно от най-забележителните постижения в тази насока е проектът MoneyVee (Фиг. 2.5). Макар и вече да не съществува, този проект предлагаше възможност за изчисляване



Фигура 2.4: Продуктовата линия TradingSolutions използваща изкуствени невронни мрежи и генетични алгоритми за прогнозиране на Forex финансови инструменти

на прогнози с помощта на дарена изчислителна мощност в разпределена среда [3]. Същинските прогнози се пресмятаха, с помощта на изкуствени невронни мрежи, върху изчислителните машини на потребителите в периоди когато натоварването на машините е ниско и се активира програмата за защита на монитора (screensaver). Преди масовото навлизане на мобилните устройства честа практика при проектите за дарена изчислителна мощност с цел пресмятане в разпределена среда, основен подход бе извършване на пресмятанятията в специално създадена за целта програма за предпазване на монитора. След навлизането на катодно лъчевите тръби при настолните компютри се появява ефект от увреждане на монитора, ако върху него продължително се визуализира статична картина. Решаването на този проблем се оказва най-удачно с помощта на операционната система, която да установи период в който потребителят не използва изчислителната машина и да активира софтуерна програма за предпазване на монитора. Развитието на мониторите с течни кристали постепенно изведе от употреба мониторите с катодно-лъчеви тръби и използването на програми за предпазване на монитора изгуби своето първоначално предназначение. Въпреки това този вид софтуерни решения останаха в употреба и основно служат за повишаване на информационната сигурност, като не само дават естетическа визуализация, но и привеждат работната сесия на потребителя в заключено състояние, което възпрепятства използването на компютърната система от неоторизирани потребители. Наличието на изчислителни ресурси, които са неефективно използвани, дава основание на множество учени да разработят системи за отдалечено пресмятане в разпределена среда точно под формата на програми за предпазване на монитора, които да се възползват от дарените потребителски изчислителни ресурси.

## 2.5 Фонови пресмятания върху мобилни устройства

Съществува концептуална разлика в начина, по който потребителите използват настолните си компютри и мобилните устройства. На първо място, настолният компютър бива пускан и спиран според нуждите на потребителя, докато най-често мобилните устройства са в непрекъснат режим на употреба. Това води до основната разлика, че мобилните устройства не изпадат в режим на занижена употреба, но също така имат режими за употреба при изключителна важност (примерно телефонно обаждане с висок приоритет). Втората фундаментална разлика се състои във факта, че основен похват за пестене на електрическа енергия, доставяна предимно от батерии при мобилните устройства, е динамичното изгасване на екрана. Тази стратегия за пестене на енергия кардинално отменя концепцията за програма, предпазваща монитора. За да се реализира ефективно система за разпределени пресмятания върху мобилни устройства, е много по-удачно да се използва технологията за активен тапет, отколкото да се залага на идеята за програма, предпазваща монитора. Точно тази идея е развита в настоящото учебно помагало.



<https://upload.wikimedia.org/wikipedia/commons/8/88/Moneyb2.gif>

Фигура 2.5: Системата MoneyBee за финансово прогнозиране в разпределена среда

## Глава 3

# Софтуерна архитектура

Разработката на софтуер е в областта на инженерните науки, тъй като продуктът се създава по принципите за изграждане на конструкции (в случая софтуерни). При стартирането на нов софтуерен проект трябва да се вземат редица решения според заданието на потребителя. Цел на настоящото помагало е софтуерно решение, което извършва изчисления от страната на клиентски мобилни устройства. Задачите за пресмятане се възлагат от сървър и получените пресметнати резултати се получават обратно на същата машина. От страната на клиента се получават стойности за цена на валути под формата на времеви ред. Сървърът изпраща на клиента също информацията за топология на изкуствена невронна мрежа. Клиентът, от своя страна, използва котировките на валутите и информацията, за изкуствената невронна мрежа за да извършва пресмятанията, необходими за обучението на изкуствената невронна мрежа. Процесът по обучението на изкуствената невронна мрежа се извършва с помощта на генетични алгоритми, които имат за цел търсене на възможно по-близки до оптималните стойности за теглата на мрежата. Клиентското приложение също поема отговорностите за онагледяване на процеса по обучение и представяне на постигнатите прогнози. Така представенаq системата съвсем естествено води към избора на софтуерна архитектура „клиент-сървър“.

### 3.1 Избор на развойни средства

В съвременната софтуерна индустрия има голям избор от развойни средства за различните нужди на софтуерните разработчици. Една част от развойните средства са комерсиални, докато друга част са инструменти с отворен код. В настоящото помагало, акцентът основно пада върху развойни средства с отворен код, тъй като минимизирането на разходите за производство е основен стремеж с цел постигане на икономическа ефективност. Изборът на развойни инструменти е задача от областта на многокритериалния анализ и основно се характеризира с наличието на множество критерии, които често са противоречиви.

#### 3.1.1 От страна на сървъра

За уеб базирани сървър решения най-популярни са технологиите JSP, ASP, PHP и Node.js. Тъй като ASP е комерсиална технология на фирмата Microsoft тя не представлява интерес за настоящото помагало. JSP е технология на фирмата Oracle която е с отворен код и дава възможност за изграждане на стабилни корпоративни решения. Недостатък на JSP е нуждата от по-сериозни софтуерни и хардуерни ресурси по отношение на хостинга. Node.js е технология, която набира все по-голяма популярност, но все още не е достигнала достатъчно ниво на „зрялост“ като същевременно също изисква повече софтуерни и хардуерни ресурси от страна на хостинга. По отношение на PHP, технологията е с тясно предназначение и има едно от най-високите нива на „зрялост“. В същото време, разходите за хостинг при PHP са едни от най-ниските, което прави изборът на тази технология изключително икономически ефективен. Задачата на

уеб базираната сървър технология е да служи като посредник между мрежата и системата за управление на бази от данните. От страна на сървъра най-рационално е данните да се съхраняват в реляционна база данни. Съществуват множество решения, които да бъдат приложени в тази част на системата като най-популярните са: Oracle, MS SQL Server, PostgreSQL и MySQL. Oracle намира своето приложение в корпоративния сегмент и е свързан със значителни финансови разходи, което го прави неприемлив за настоящото помагало. MS SQL Server е алтернативна система на Oracle в корпоративния сегмент и също е свързана със значителни финансови разходи. PostgreSQL е система с отворен код, която е съизмерима с технологичните възможности на Oracle и е потенциално добър кандидат за ниско бюджетни разработки. В настоящото помагало PostgreSQL е избегнат поради своята ненужна сложност, при едно относително просто софтуерно решение. Изборът пада върху MySQL, тъй като системата е максимално опростена добре наложена сред потребителите и се поддържа от фирмата Oracle. Освен всичко изброено, MySQL има добра поддръжка при хостинг доставчиците и е икономически най-ефективният избор.

### 3.1.2 От страна на клиента

При „умните“ мобилни устройства най-разпространените операционни системи са Android, iOS и Windows Phone. По настоящем фирмата Microsoft преустанови развитието на своята операционна система Windows Phone, което моментално води до отхвърлянето ѝ за настоящото разглеждане. Инвестицията за разработка на приложения под iOS на фирмата Apple води до отпадането на тази платформа за нуждите на настоящото помагало. Към разходите за разработка може да се добави и фактът, че програмирането за iOS се извършва на два езика Objective-C и Swift, които имат относително малка популярност в Източна Европа. Най-много „умни“ мобилни устройства в световен мащаб се използват с операционната система Android. Android е с отворен код, поддържа се основно от компанията Google и позволява разработка със значително по-ниски финансови разходи, спрямо конкурентите си. Приложенията за Android основно се разработват на езика Java, който по настоящем е един от най-широко използваните програмни езици и се характеризира с много висока степен на „зрялост“.

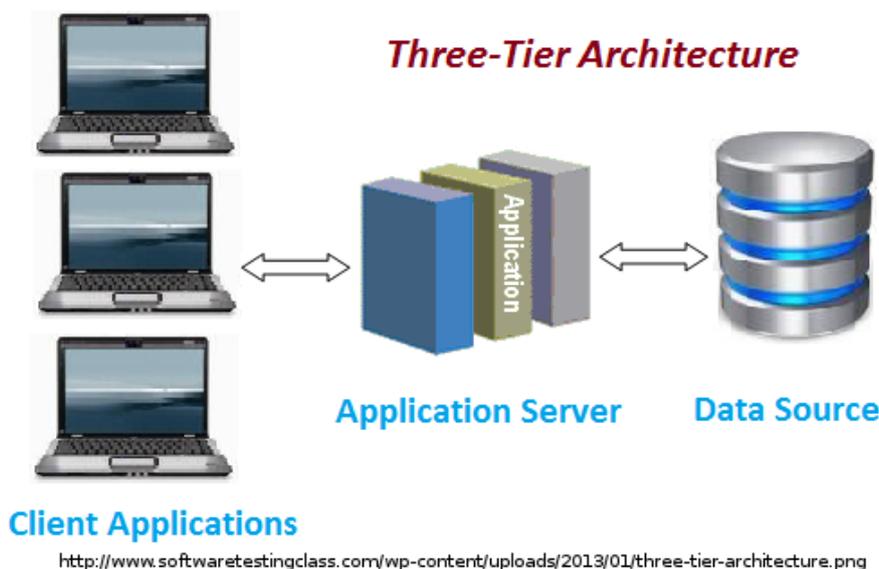
### 3.1.3 За комуникация между сървъра и клиента

Съществуват множество възможности за изграждането на комуникацията между сървъра и клиента. В най-суров вид информацията може да се предава като серия байтове (plane text), което води до множество затруднения при получаването и последващата ѝ обработка. Широко използвана алтернатива е тагиращият език XML. При този вариант информацията бива „пакетирана“ в серия от тагове, които дават определена структура и семантика. Основният замисъл при проектирането на XML е бил създаването на структурирани документи както от хора, така и от машини. Поради тази причина XML има една по-голяма експресивност в сравнение с неговата алтернатива JSON. JSON е максимално опростен тагиращ език за структурирано представяне на информацията, който води своето начало от обектите в програмния език JavaScript. За разлика от XML, JSON има основно предназначение за обмяна на структурирана информация между машини, а не толкова между хора и машини. Поради всичко изброено, в настоящото помагало изборът пада върху JSON като основа за изграждането на комуникационния протокол между сървъра и неговите клиенти. Тъй като от страната на сървъра се предвижда уеб базирано решение, то JSON базираният протокол ще протича в комуникационни сесии на HTTP протокола. В зората на уеб страниците е разработен протоколът HTTP, стъпващ на TCP/IP, за ефективно предаване на информация между уеб сървърите и уеб браузърите. HTTP протоколът е добре наложен и с добра поддръжка в световен мащаб. Една важна негова характеристика е, че при този протокол комуникацията е разделена на заявки и отговори без да се поддържа постоянна комуникационна линия.

## 3.2 Компоненти на системата

След направения кратък обзор на технологии и развойни средства изборът е за създаването на „клиент-сървър“ система. От страна на сървъра се разполагат модули за съхранение на данните (MySQL система

за управление на бази от данни) и за комуникация (PHP уеб скриптове). Уеб сървърът комуникира с клиентите на база JSON/HTTP комуникационен протокол. Android мобилни устройства правят уеб заявки, извършват изчисленията и връщат резултата до сървъра. Тъй като се разчита на дарена изчислителна мощност, е нужно да се избере подходящ начин за използване на мобилното устройство без това да нарушава основните му функции и без да пречи на потребителите. Със своята технология Active Wallpaper, Android предлага идеална възможност за целите на настоящото помагало. Активният тапет представлява изображение, което се изрисува зад всички основни графични компоненти от графичния потребителски интерфейс на Android. По-същественото е, че активният тапет е Java приложение, което работи в постоянен фонов режим и може да изпълнява определени кратки задачи, когато устройството не е високо натоварено.

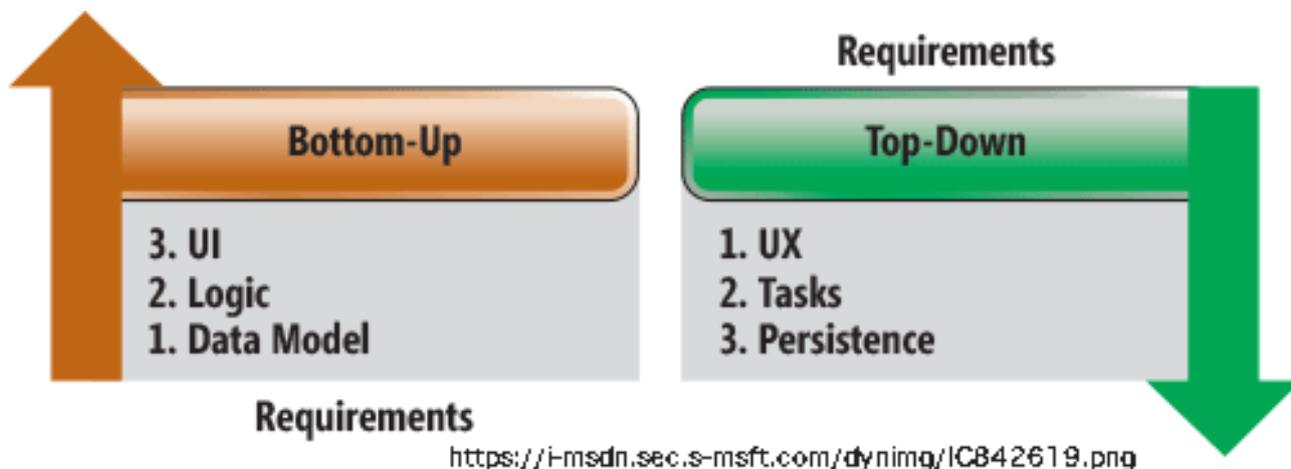


Фигура 3.1: Уеб базирана трислойна софтуерна архитектура

При реализацията на настоящата система за изчисления в разпределена среда, изборът пада върху класическа трислойна софтуерна архитектура. Същият подход за три слоя се прилага и при реализацията на мобилното приложение, където SQLite локално съхранява данните над които се работи, Java обектно-ориентиран код извършва изчисленията, а Android базиран графичен потребителски интерфейс поема отговорността за визуализацията на процеса пред потребителя.

### 3.2.1 Подход за разработка

Ако приемем, че визуализацията е най-горният слой, а базата данни най-долният слой на една трислойна софтуерна архитектура, то има два основни подхода за изработване на софтуерната система (Фиг. 3.2). При първия подход първоначално се разработва визуалния интерфейс, след това слоя на работната логика и накрая базата данни. Този подход се нарича „top-down“ и е полезен, когато се анализира софтуерно задание, при което вече има в употреба множество хартиени документи (първоизточници на работните екрани). Този подход също е удобен при проекти с малък размер, където базата данни е относително опростена. Вторият много популярен подход е когато се започне с проектирането на базата данни, след това работната логика, която обработва данните и едва накрая екраните, визуализиращи информацията. Този подход се нарича „bottom-up“ и е най-подходящ, когато се разработва сложно софтуерно решение, което се очаква да работи с големи обеми данни и множество различни структури на информацията. Акцентът в настоящото помагало е върху изчисленията които мобилните устройства извършват, а не толкова към създаването на голям масив от данни. Поради тази причина изборът в случая пада върху подхода „top-down“.



Фигура 3.2: Подходи за софтуерна разработка

### 3.3 Лиценз и хранилище за проекта

При съвременните софтуерни проекти с отворен код са от значение две неща – юридическият лиценз под който съществува проектът и публичното хранилище, в което е разположен програмният текст.

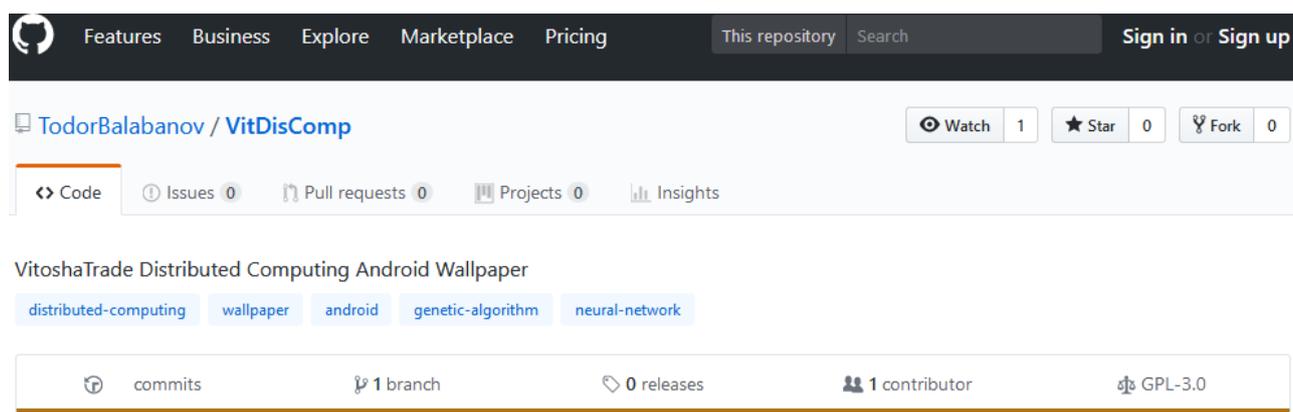
#### 3.3.1 Лиценз

От съществено значение е изборът на правилен софтуерен лиценз, когато се разработва софтуерен проект предвиден да бъде публично достъпен. Съществуват множество възможности като някои от най-популярните са - BSD License, MIT license, Mozilla Public License и GNU General Public License. За нуждите на настоящото помагало е предпочетен GNU General Public License v3, защото този лиценз е най-предпазващ за създателя на софтуерния продукт. В най-общи линии GPL3 позволява - комерсиална употреба, модификации, разпространение, включване в патенти и употреба за лични нужди. Лицензът съпровожда изключително ограничена отговорност за създателите на продукта и абсолютно никаква гаранция за употребата му от страна на потребителите. Лицензът също налага и серия ограничения – задължително включване на текст за авторските права на създателите, списък на извършените промени, не позволява закриване на кода и задължава всяко надграждане на продукта да бъде под същия лиценз. Със своята протекционистка природа GPL е един от лицензите дал най-силен тласък в развитието на продукти с отворен код, което е достатъчна причина да бъде избран за разработки без ясно изразена комерсиална насоченост.

#### 3.3.2 Хранилище за програмен код

Прието е всеки проект да има название, което особено важи в света на софтуера с отворен код. За настоящото помагало е избрано името VitDisComp. Това название е избрано, тъй като разработката ще се възползва от наличното сървър решение в проекта VIToshatrade [4] и по своята същност проектът е DIStributed COMPuting решение.

Съществуват различни възможности за публикуване на програмния код, но една от най-популярните алтернативи е облачната услуга GitHub. Услугата GitHub (Фиг. 3.3) е базирана на системата за контрол на версиите Git и дава една от най-широките възможности за популяризиране на програмен код с отворен лиценз.



Фигура 3.3: Публикуване на проект в GitHub

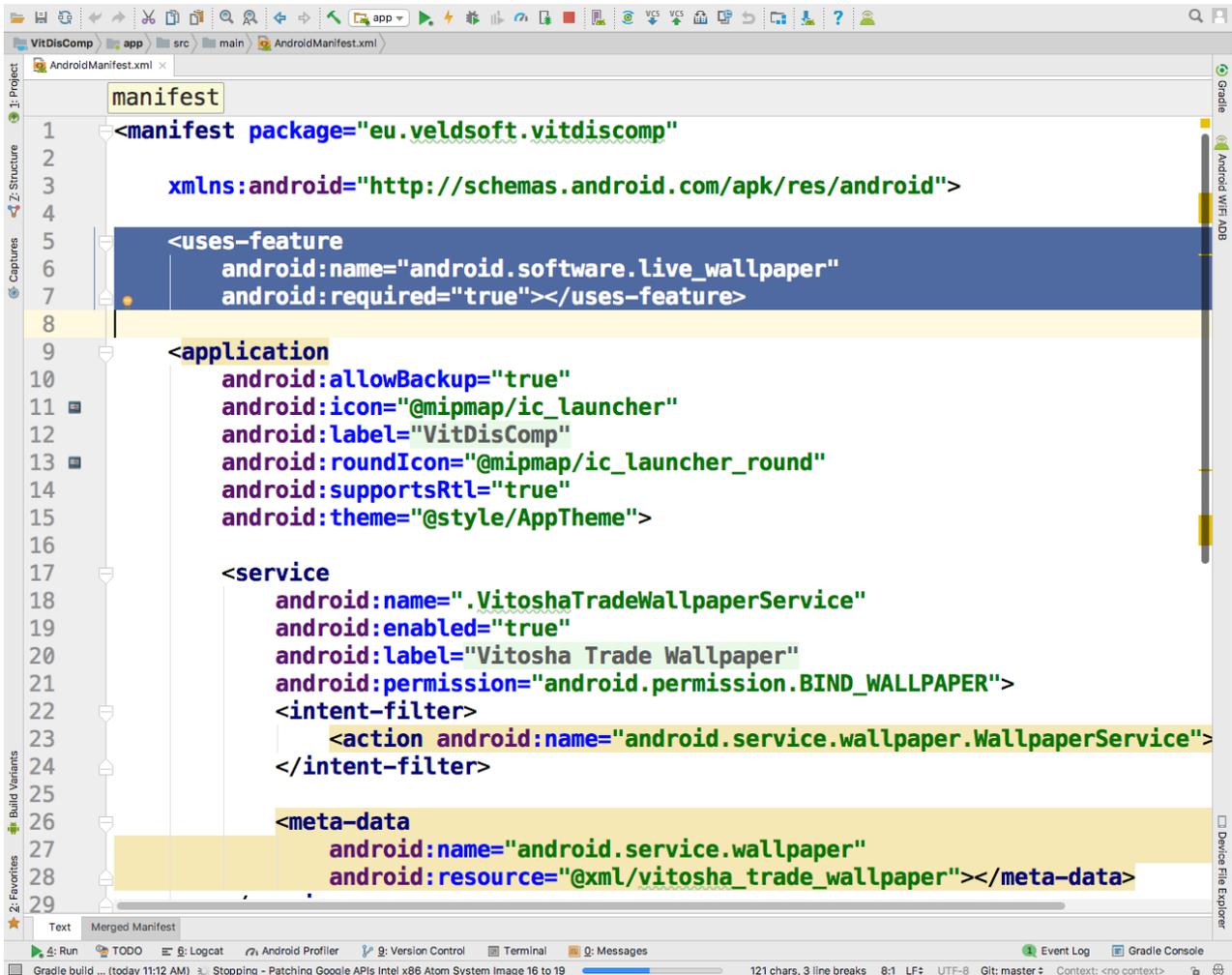
## Глава 4

# Android активен тапет

Технологията Android Live Wallpaper предоставя анимирани възможности за визуално представяне под формата на виртуален тапет. По своята същина този вид приложения не се различават драстично от останалите Android приложения и дори могат да изпълняват почти всички техни възможности. Тъй като виртуалният тапет е постоянно активен той е идеален кандидат за реализацията на пресмятания във фонов режим. За създаването на активен тапет са необходими следните компоненти: 1. XML файл който описва компонентите на тапета; 2. Фонов модул (Android Service); 3. Подходящи флагове за достъп до ресурсите на устройството.

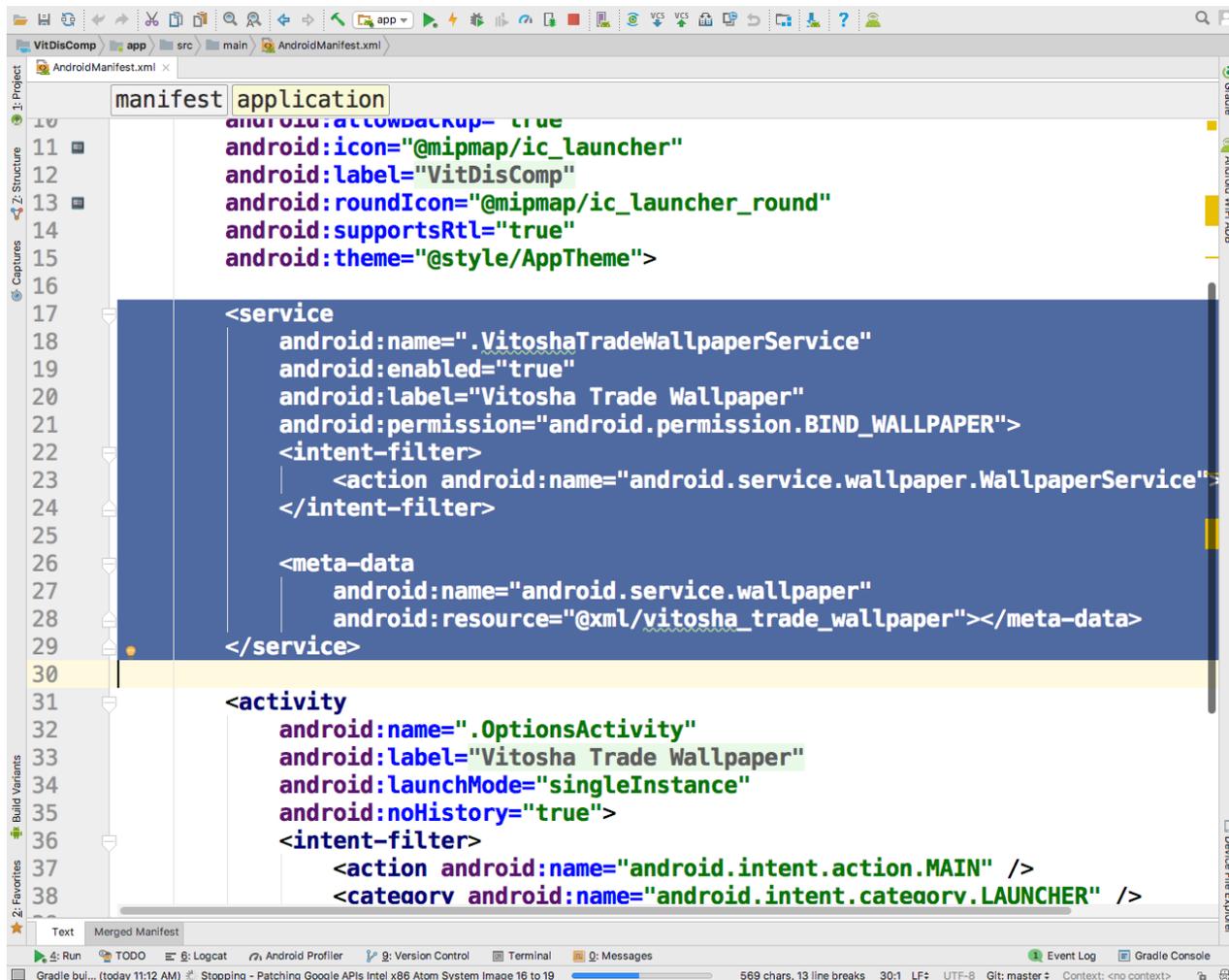
### 4.1 Манифест файл на мобилното приложение

При създаването на приложения за операционната система Android е възприет подходът компонентите на мобилното приложение да бъдат описани в манифест файл (AndroidManifest.xml) с използването на XML синтаксис.



Фигура 4.1: Определяне на приложението като активен тапет

За да се определи приложението като приложение от тип активен тапет, е необходимо това изрично да се маркира в манифест файла (Фиг. 4.1). Най-съществената полза от тази дефиниция е, че тя предотвратява възможността приложението да бъде инсталирано на устройства, които не поддържат възможностите за визуално представяне на активен тапет.



Фигура 4.2: Модул от тип “услуга”

Когато в Android се използват много продължителни пресмятания, които не са удачни за извършване, в нишка е прието тези изчисления да се изнасят в модули без графичен потребителски интерфейс, наречени „услуги“ (Service). Работата на активния тапет се извършва точно в такъв модул и поради тази причина в проекта е добавен един (Фиг. 4.2).

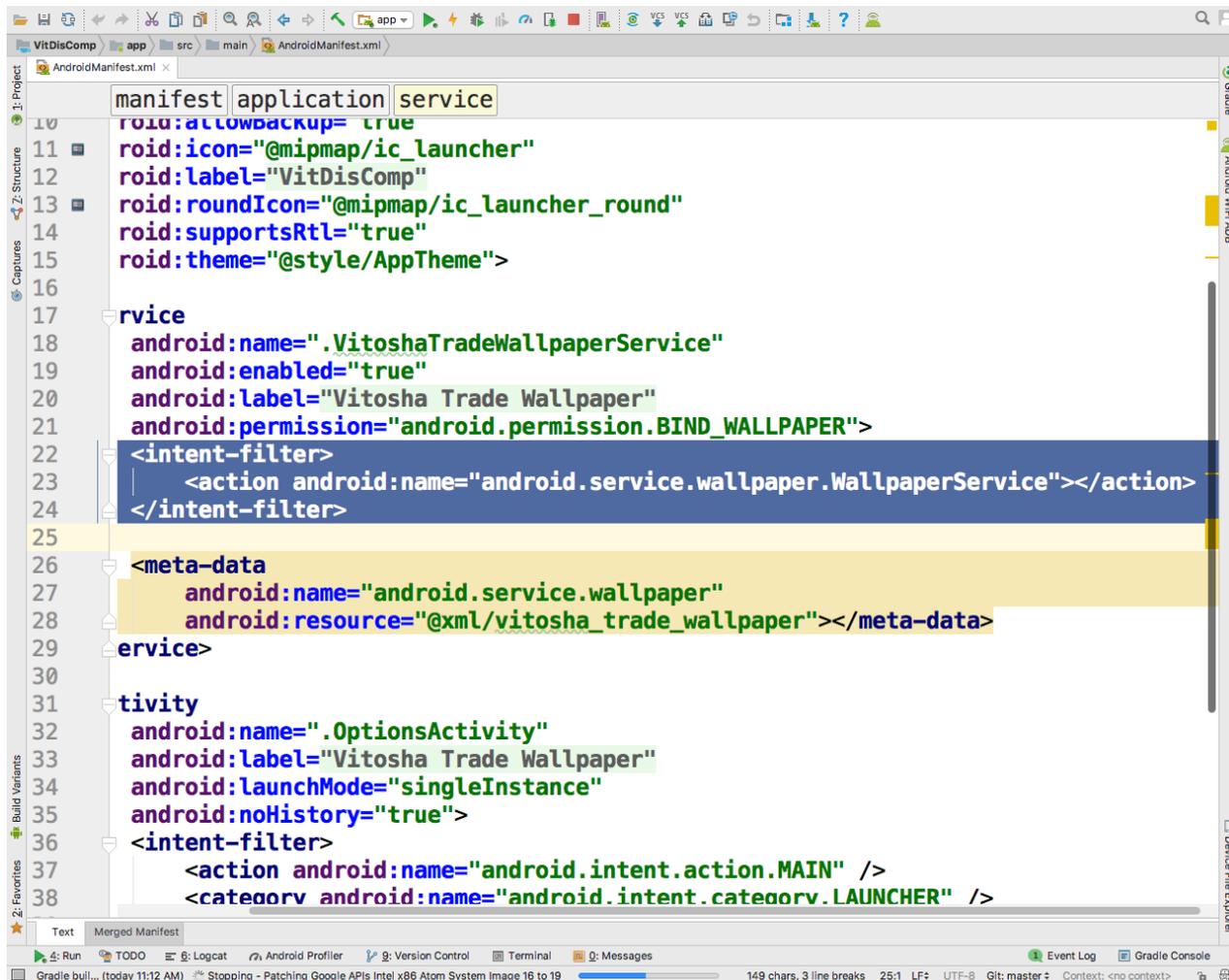
```

10  manifest application service
11  android:allowBackup="true"
12  android:icon="@mipmap/ic_launcher"
13  android:label="VitDisComp"
14  android:roundIcon="@mipmap/ic_launcher_round"
15  android:supportRtl="true"
16  android:theme="@style/AppTheme">
17
18  <service
19      android:name=".VitoshaTradeWallpaperService"
20      android:enabled="true"
21      android:label="Vitosha Trade Wallpaper"
22      android:permission="android.permission.BIND_WALLPAPER">
23      <intent-filter>
24          <action android:name="android.service.wallpaper.WallpaperService">
25      </intent-filter>
26
27      <meta-data
28          android:name="android.service.wallpaper"
29          android:resource="@xml/vitosha_trade_wallpaper"></meta-data>
30  </service>
31
32  <activity
33      android:name=".OptionsActivity"
34      android:label="Vitosha Trade Wallpaper"
35      android:launchMode="singleInstance"
36      android:noHistory="true">
37      <intent-filter>
38          <action android:name="android.intent.action.MAIN" />
39          <category android:name="android.intent.category.LAUNCHER" />

```

Фигура 4.3: Флагове за достъп до ресурса активен тапет

Моделът за сигурност изисква за всяко по-особено действие да се получи изричното съгласие на потребителя. В случая с активния тапет е необходимо добавянето на `android.permission.BIND_WALLPAPER` разрешение (Фиг. 4.3).



Фигура 4.4: Регистрация на услугата за прослушване на съобщения от операционната система.

Освен нуждата от разрешения за ползване на ресурсите за активен тапет, е нужно услугата да се абонира за прослушване на съобщения от операционната система (Фиг. 4.4).

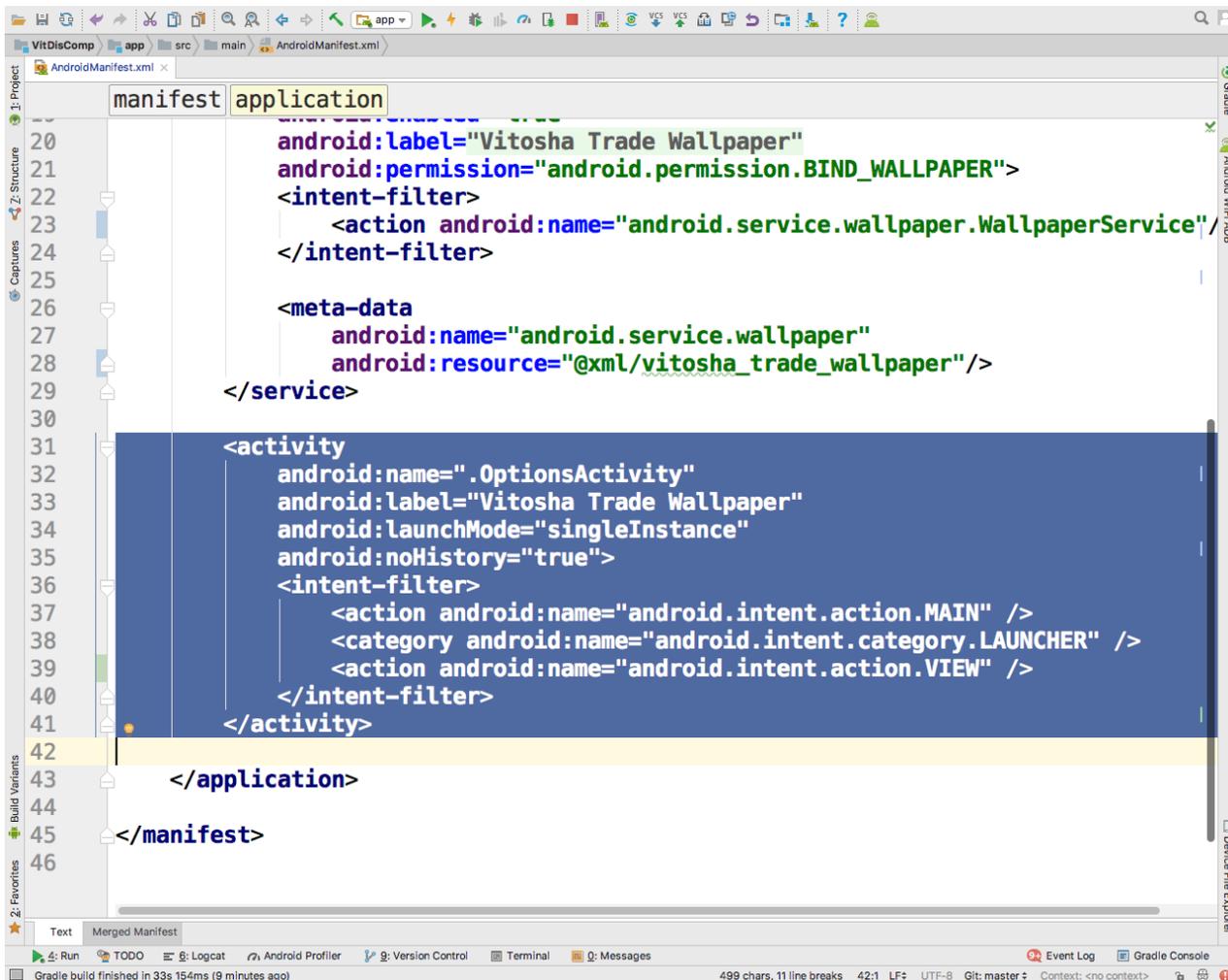
```

11  android:icon="@mipmap/ic_launcher"
12  android:label="VitDisComp"
13  android:roundIcon="@mipmap/ic_launcher_round"
14  android:supportsRtl="true"
15  android:theme="@style/AppTheme">
16
17  <service
18      android:name=".VitoshaTradeWallpaperService"
19      android:enabled="true"
20      android:label="Vitosha Trade Wallpaper"
21      android:permission="android.permission.BIND_WALLPAPER">
22      <intent-filter>
23          <action android:name="android.service.wallpaper.WallpaperService">
24      </intent-filter>
25
26      <meta-data
27          android:name="android.service.wallpaper"
28          android:resource="@xml/vitosha_trade_wallpaper"></meta-data>
29  </service>
30
31  <activity
32      android:name=".OptionsActivity"
33      android:label="Vitosha Trade Wallpaper"
34      android:launchMode="singleInstance"
35      android:noHistory="true">
36      <intent-filter>
37          <action android:name="android.intent.action.MAIN" />
38          <category android:name="android.intent.category.LAUNCHER" />
39  </activity>

```

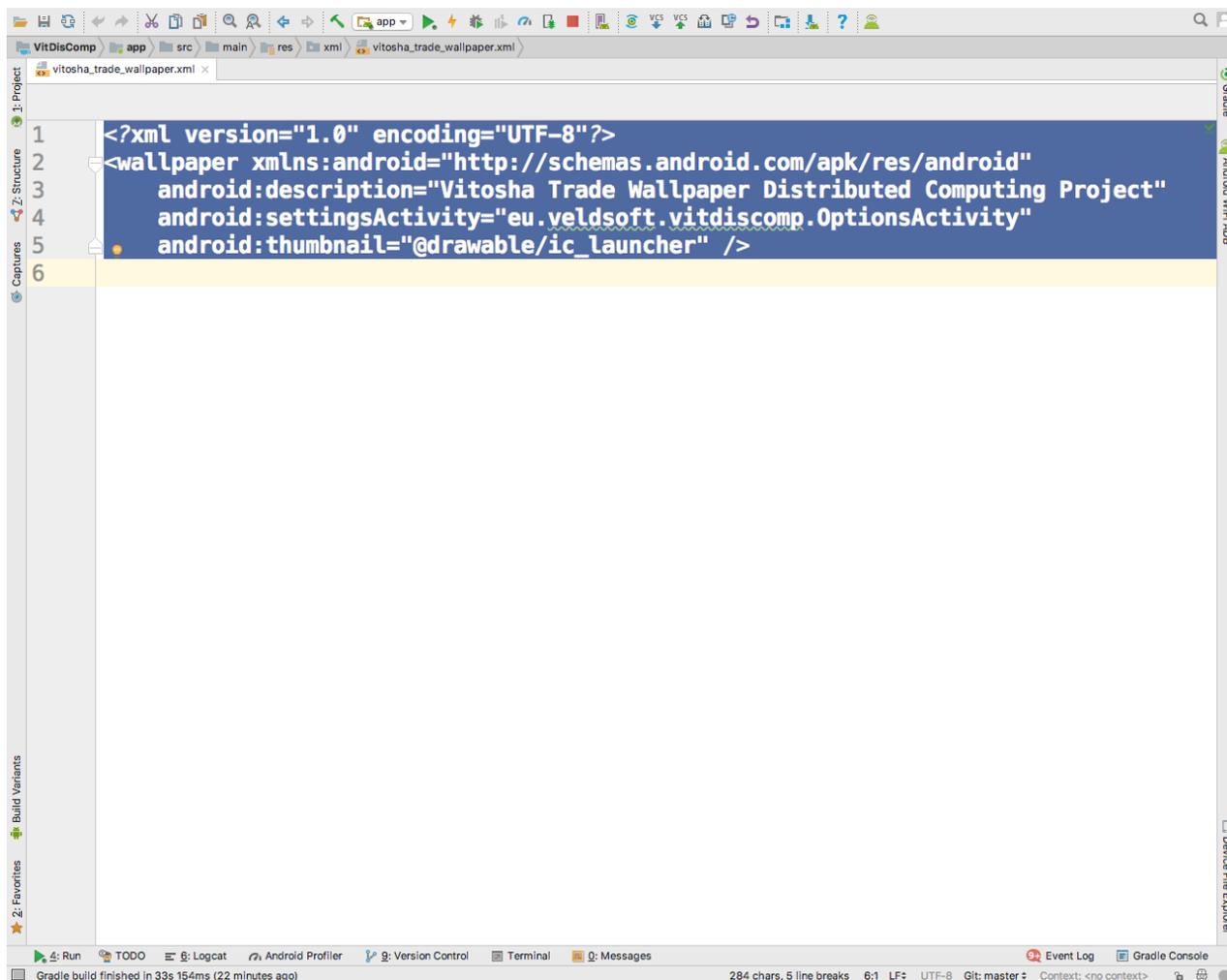
Фигура 4.5: Референция към описателния файл на активния тапет

Описанието на самия активен тапет се съдържа в отделен XML файл, референция към който се посочва в манифеста (Фиг. 4.5).



Фигура 4.6: Прозорец за установяване на активния тапет

Последният компонент в такъв тип приложение е прозорец (Android Activity), който да се стартира от операционната система и да служи за установяване на активния тапет (Фиг. 4.6). В случая се използва възможността този стартов прозорец да съчетава и функциите на прозорец с опции (Preference Activity).

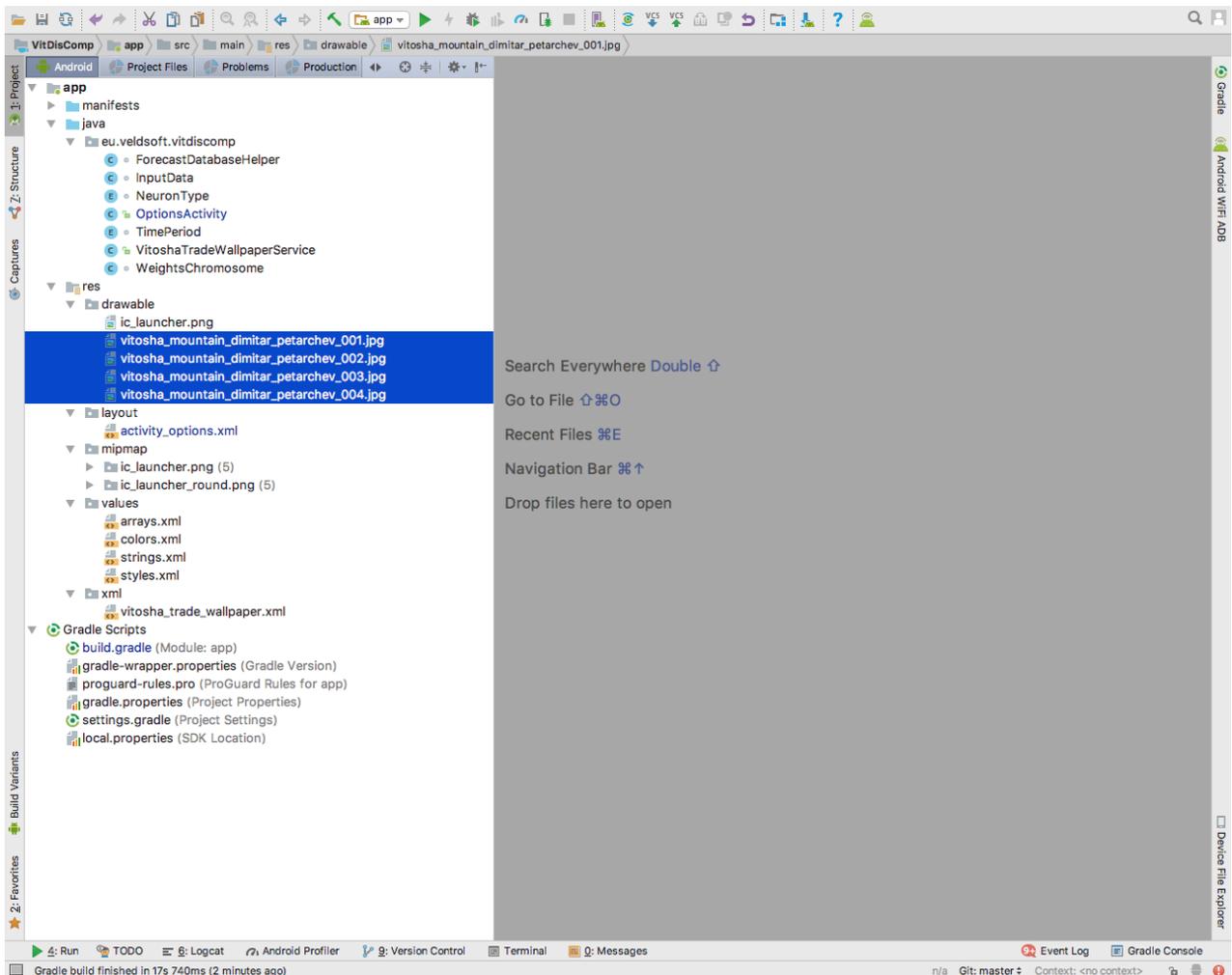


Фигура 4.7: XML описателен файл на тапета

Както вече бе споменато, активният тапет се описва в отделен XML файл, който съдържа кратка анотация на приложението, предварителен изглед, умалена икона и името на прозореца за настройки (Фиг. 4.7).

## 4.2 Екран с настройки

Тъй като активният тапет ще има и вторична задача да визуализира прогреса на пресмятанията, то е разумно към него да бъде създаден и прозорец с настройки.



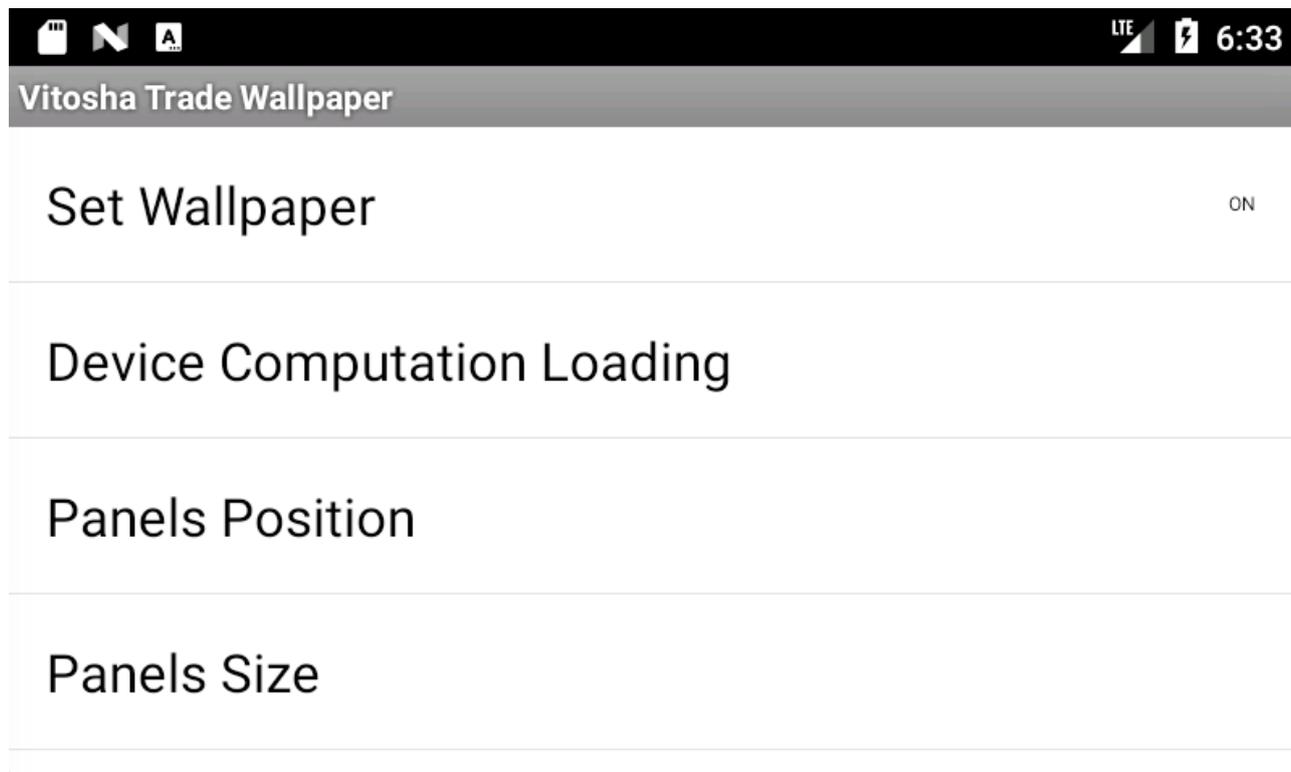
Фигура 4.8: Графични файлове, съдържащи фотографии на планината Витоша до град София, България

Като визуално представяне е избран възможно най-опростен вариант. Няколко фотографии на планината Витоша се визуализират под формата на отрязъци с размерите на екрана който мобилното устройство има (Фиг. 4.8). В три полупрозрачни области се визуализира информация за финансовия времеви ред (код и времеви период), стълб-диаграма за входните и изходните данни и текущо състояние на изкуствената невронна мрежа (Фиг. 4.9).



Фигура 4.9: Висуално представяне на информация от изчисленията

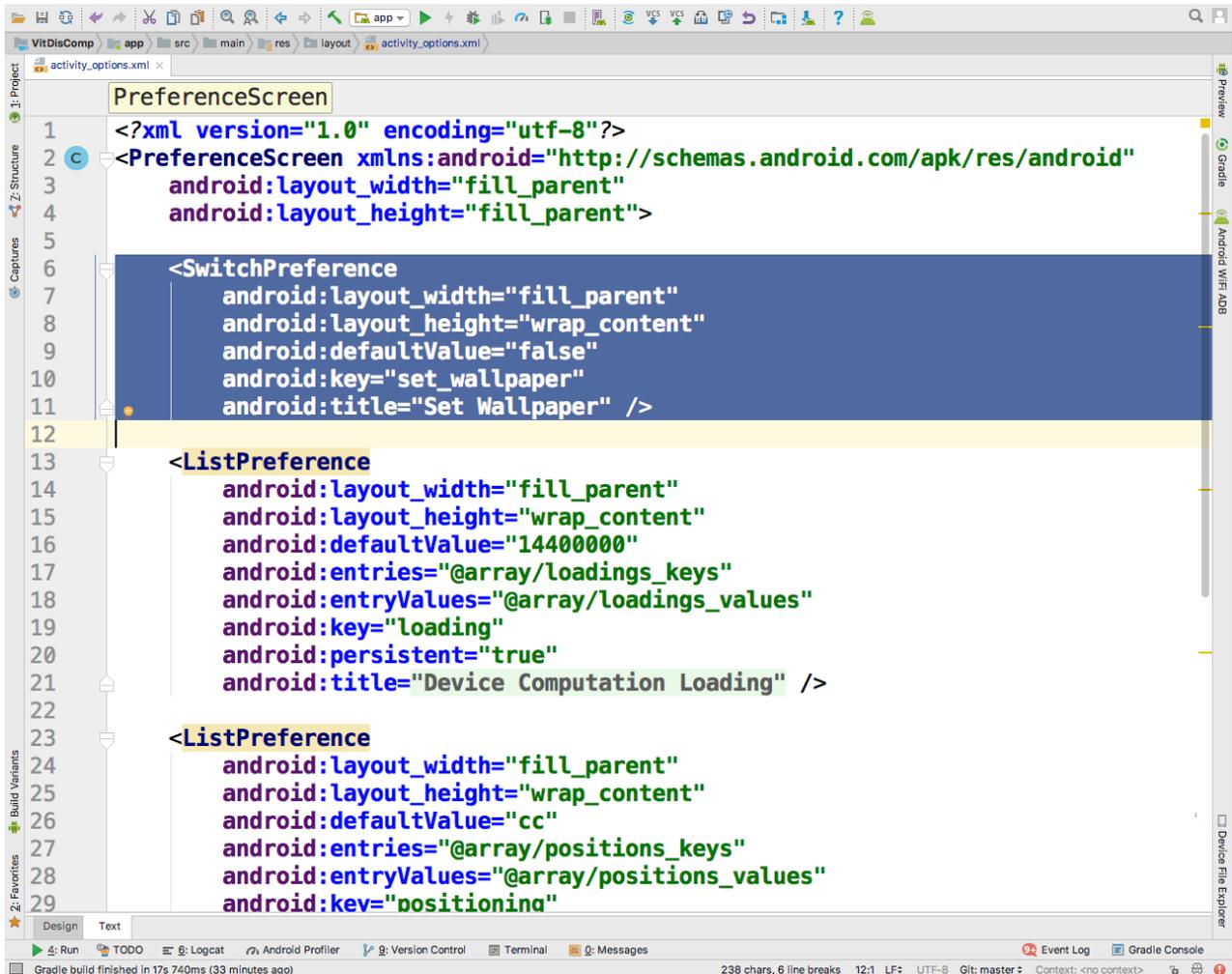
В настройките за активния тапет са предоставени възможности за управление на позицията и размера за трите визуални области. Също така са добавени първоначални настройки за натоварване на устройството и дали активният тапет да бъде включен (Фиг. 4.10).



Фигура 4.10: Първоначален набор от настройки

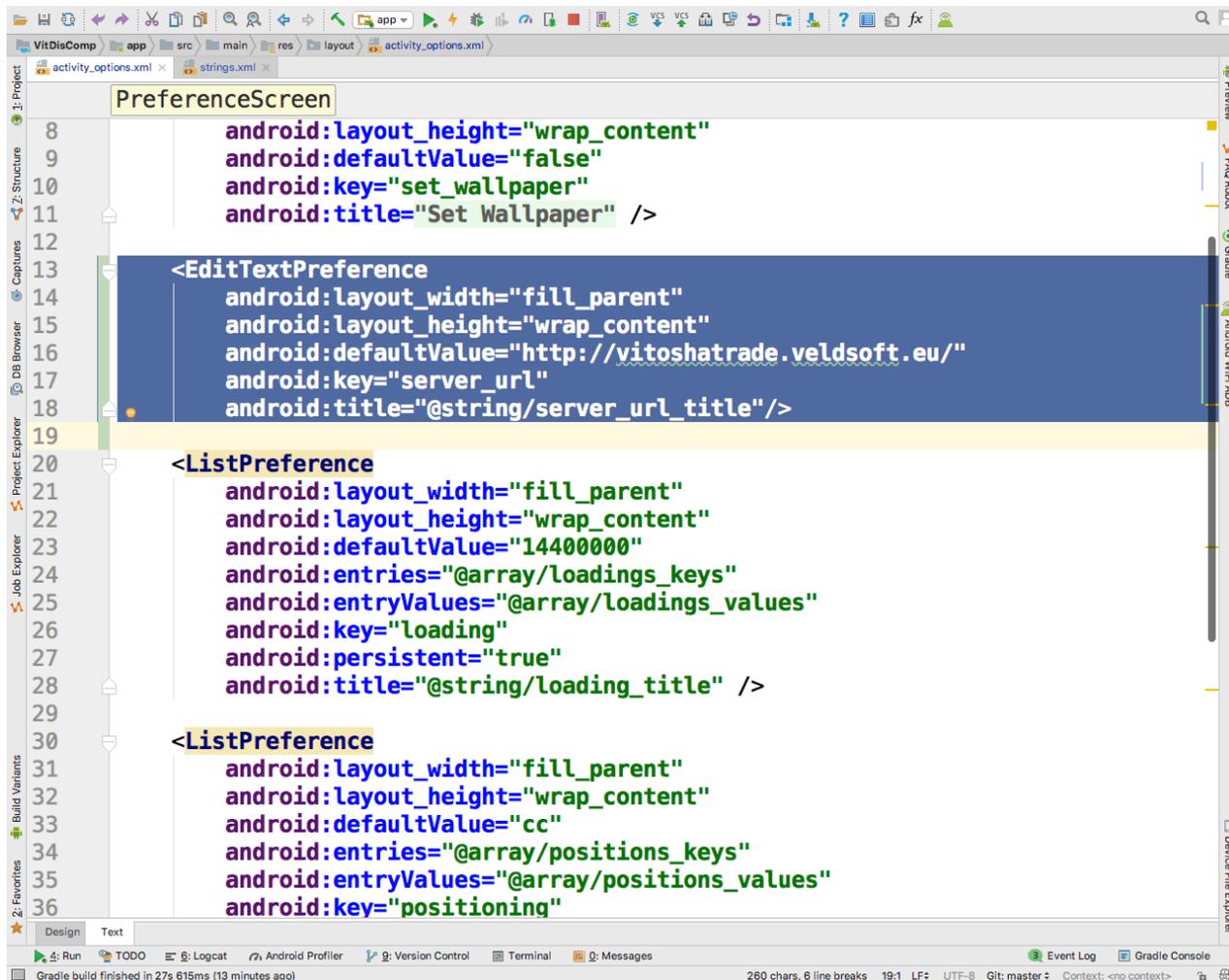
Всеки прозорец в Android се описва със свой файл за разполагане (layout) и файл с Java програмен код. Описателният файл за графичен потребителски интерфейс използва XML и много наподобява съставянето на уеб страница. Когато се изготвя екран за настройки един от най-полезните инструменти в операционната система Android са споделените предпочитания (Shared Preferences). Те позволяват състоянието на визуалните компоненти директно да бъде запаметено в устройството под формата на ключ-стойност двойки и след това програмно тази информация да бъде използвана.

#### 4.2.1 Описание на потребителския интерфейс под формата на XML файлове



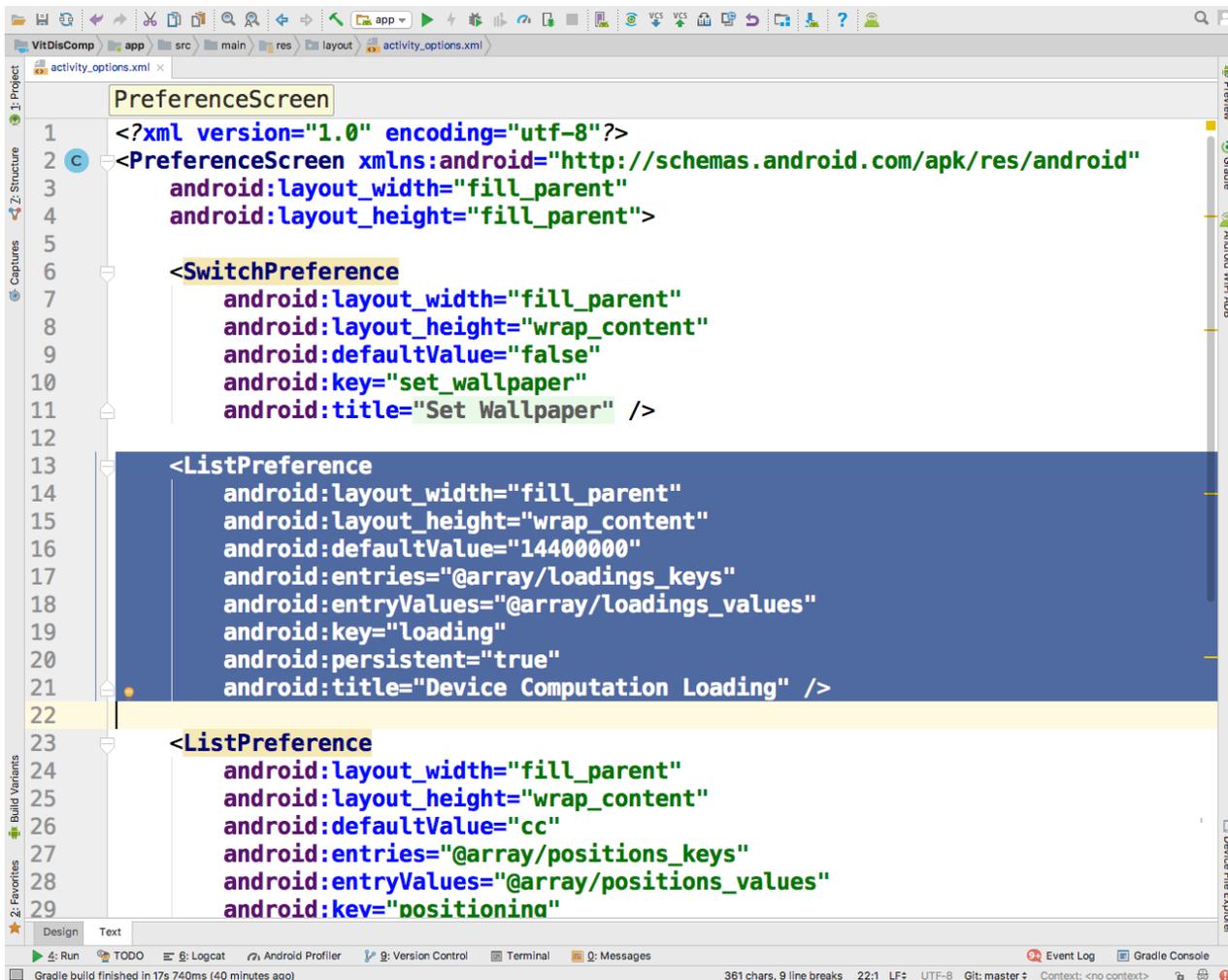
Фигура 4.11: Визуален компонент за включване и изключване на тапета

На първо място е разположен визуален компонент за включване и изключване на активния тапет. Когато ключът е в състояние ON активният тапет бива стартиран, а когато е в позиция OFF, активният тапет бива деактивиран (Фиг. 4.11).



Фигура 4.12: Визуален компонент за посочване на сървър URL

Тъй като мобилното приложение тегли информацията за времевите редове от отдалечен веб сървър, то е подходящо потребителят да има възможност да настройва URL адреса на сървъра, с който се работи (Фиг. 4.12). Тази опция позволява пренасочване на трафика към други сървъри, след като мобилното приложение вече е инсталирано на потребителските мобилни устройства.



Фигура 4.13: Визуален компонент за регулиране на натоварването

За натоварването на устройството при извършването на фоновите пресмятания се използва списък с предварително дефинирани стойности (Фиг. 4.13).

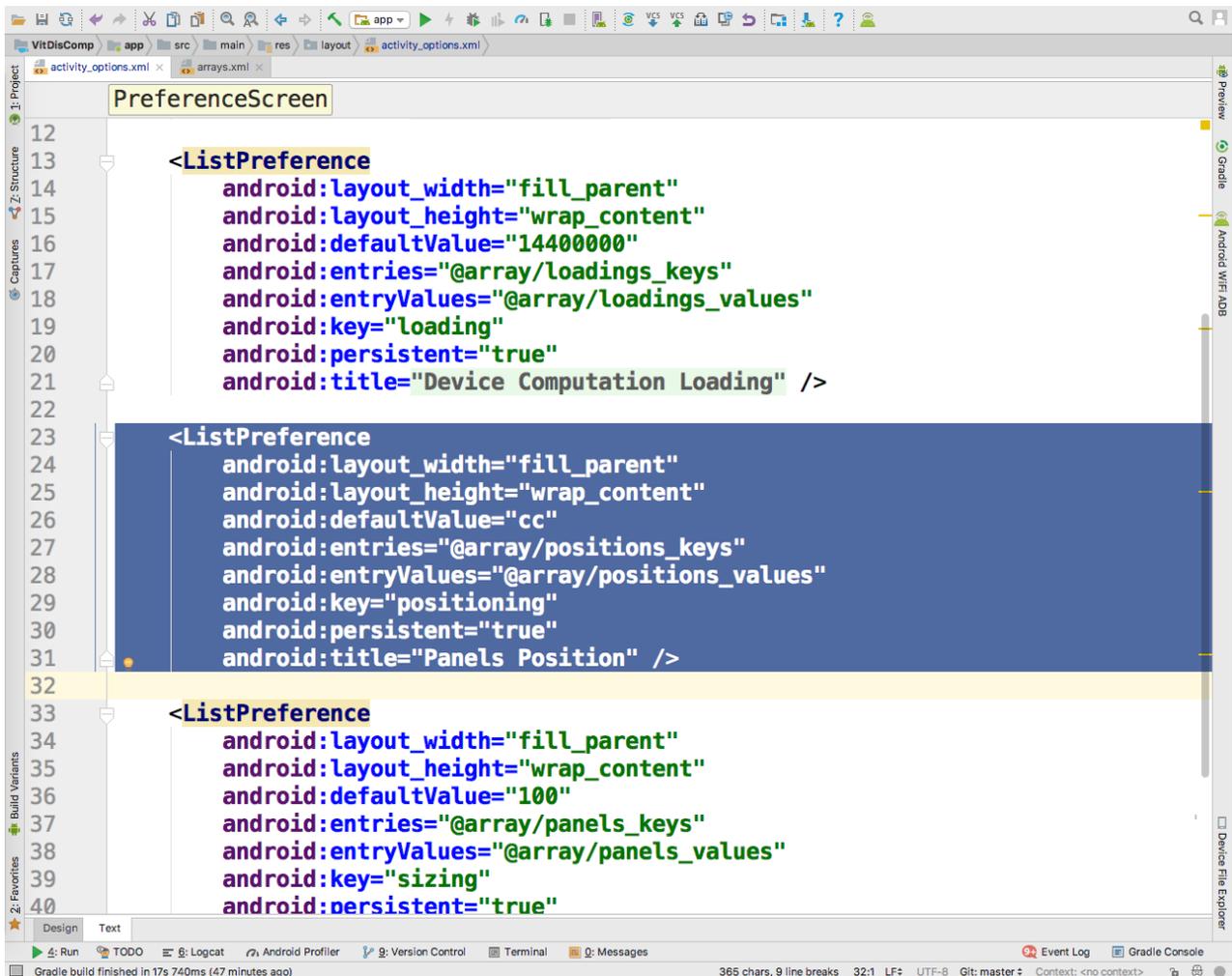
```

resources
1  <?xml version="1.0" encoding="utf-8"?>
2  <resources>
3
4  <string-array name="loadings_keys">
5      <item name="very_low">Very Low</item>
6      <item name="low">Low</item>
7      <item name="normal">Normal</item>
8      <item name="high">High</item>
9      <item name="very_high">Very High</item>
10 </string-array>
11 <string-array name="loadings_values">
12     <item name="very_low">1440000</item>
13     <item name="low">720000</item>
14     <item name="normal">360000</item>
15     <item name="high">180000</item>
16     <item name="very_high">60000</item>
17 </string-array>
18 <string-array name="positions_keys">
19     <item name="left_top">Left-Top</item>
20     <item name="center_top">Center-Top</item>
21     <item name="righth_top">Righth-Top</item>
22     <item name="left_center">Left-Center</item>
23     <item name="center_center">Center-Center</item>
24     <item name="righth_center">Righth-Center</item>
25     <item name="left_bottom">Left-Bottom</item>
26     <item name="center_bottom">Center-Bottom</item>
27     <item name="righth_bottom">Righth-Bottom</item>
28 </string-array>
29 <string-array name="positions_values">
30     <item name="left_top">lt</item>

```

Фигура 4.14: Стойности за натоварване на системата

При проектирането на системата Android, една от основните цели е била максимално разделяне на графичния интерфейс от данните. Точно поради тази причина стойностите за натоварване са изнесени в отделен ресурс (Фиг. 4.14).



Фигура 4.15: Позициониране на областите за визуално представяне

Позиционирането на областите за визуално представяне също се настройва с избор на стойности от списък (Фиг. 4.15).

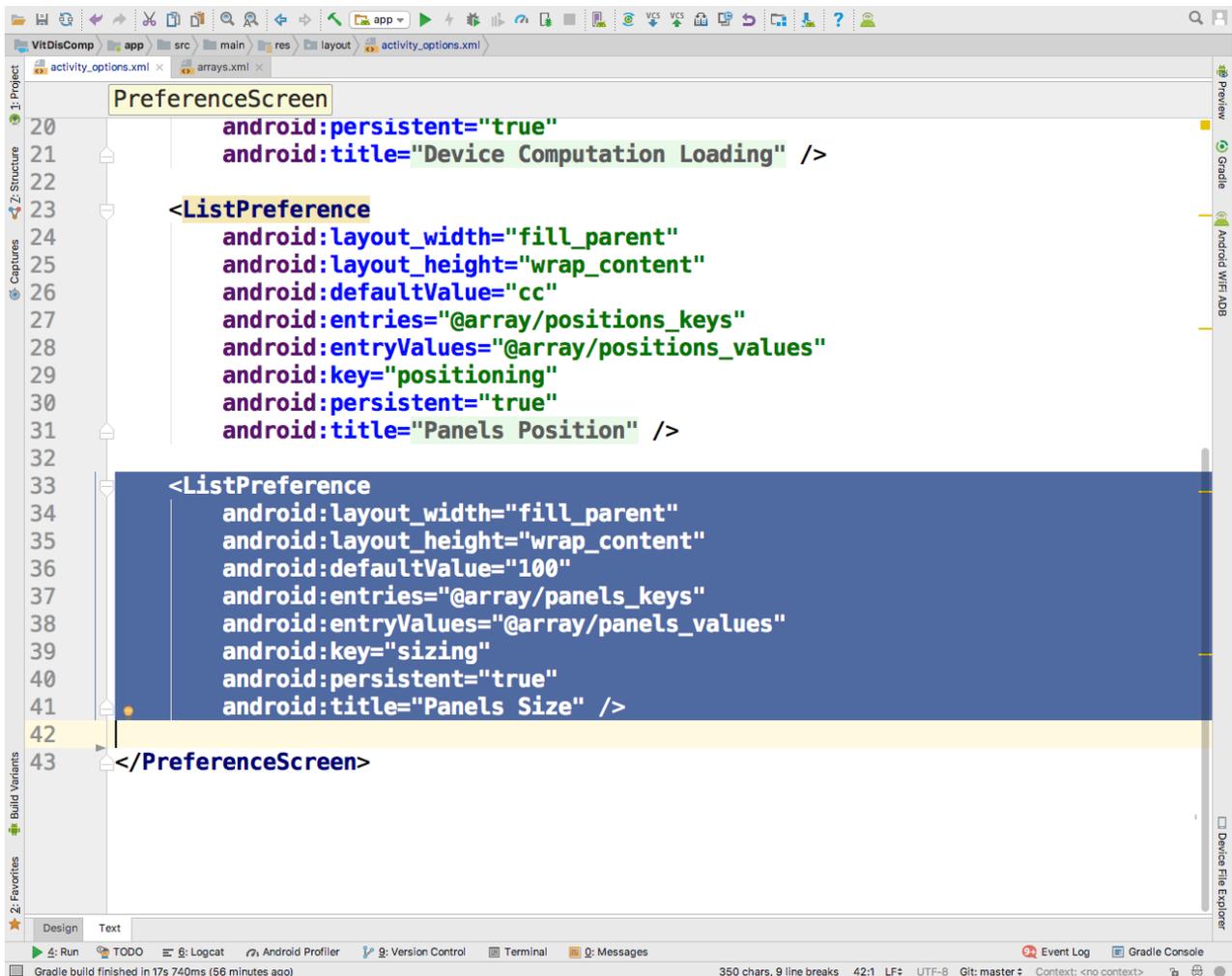
```

resources
12 <item name="very_low">1440000</item>
13 <item name="low">720000</item>
14 <item name="normal">360000</item>
15 <item name="high">180000</item>
16 <item name="very_high">60000</item>
17 </string-array>
18 <string-array name="positions_keys">
19 <item name="left_top">Left-Top</item>
20 <item name="center_top">Center-Top</item>
21 <item name="righth_top">Righth-Top</item>
22 <item name="left_center">Left-Center</item>
23 <item name="center_center">Center-Center</item>
24 <item name="righth_center">Righth-Center</item>
25 <item name="left_bottom">Left-Bottom</item>
26 <item name="center_bottom">Center-Bottom</item>
27 <item name="righth_bottom">Righth-Bottom</item>
28 </string-array>
29 <string-array name="positions_values">
30 <item name="left_top">lt</item>
31 <item name="center_top">ct</item>
32 <item name="righth_top">rt</item>
33 <item name="left_center">lc</item>
34 <item name="center_center">cc</item>
35 <item name="righth_center">rc</item>
36 <item name="left_bottom">lb</item>
37 <item name="center_bottom">cb</item>
38 <item name="righth_bottom">rb</item>
39 </string-array>
40 <string-array name="panels_keys">
41 <item name="small">Small</item>

```

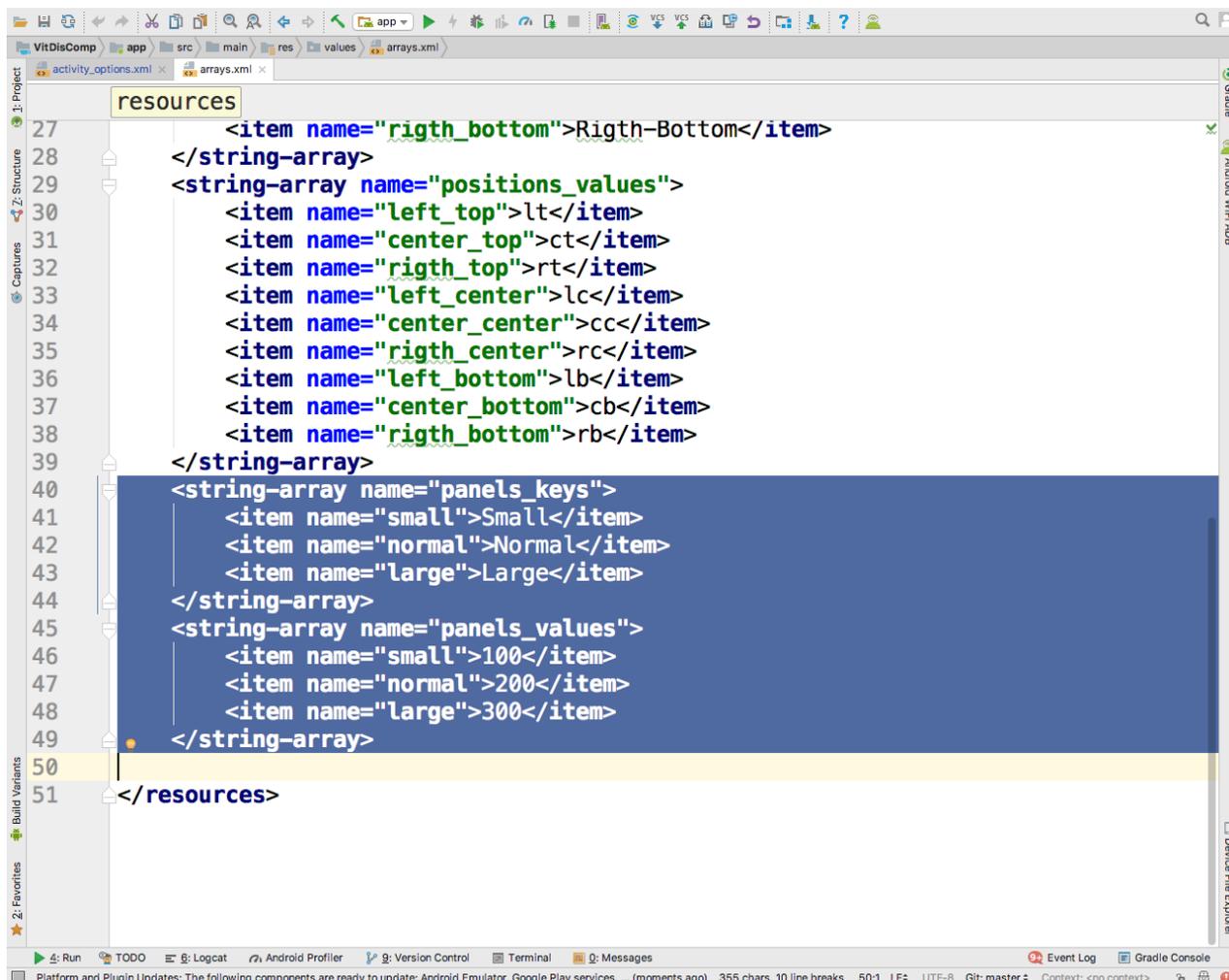
Фигура 4.16: Стойности за позициониране на областите за визуално представяне

Аналогично на стойностите за натоварване, списъкът с възможните позиции на областите за визуално представяне е изнесен в отделен ресурсен файл (Фиг. 4.16).



Фигура 4.17: Размер на областите за визуално представяне

От първоначалните характеристики, последна е размерът на областите за визуално представяне на информацията (Фиг. 4.17).

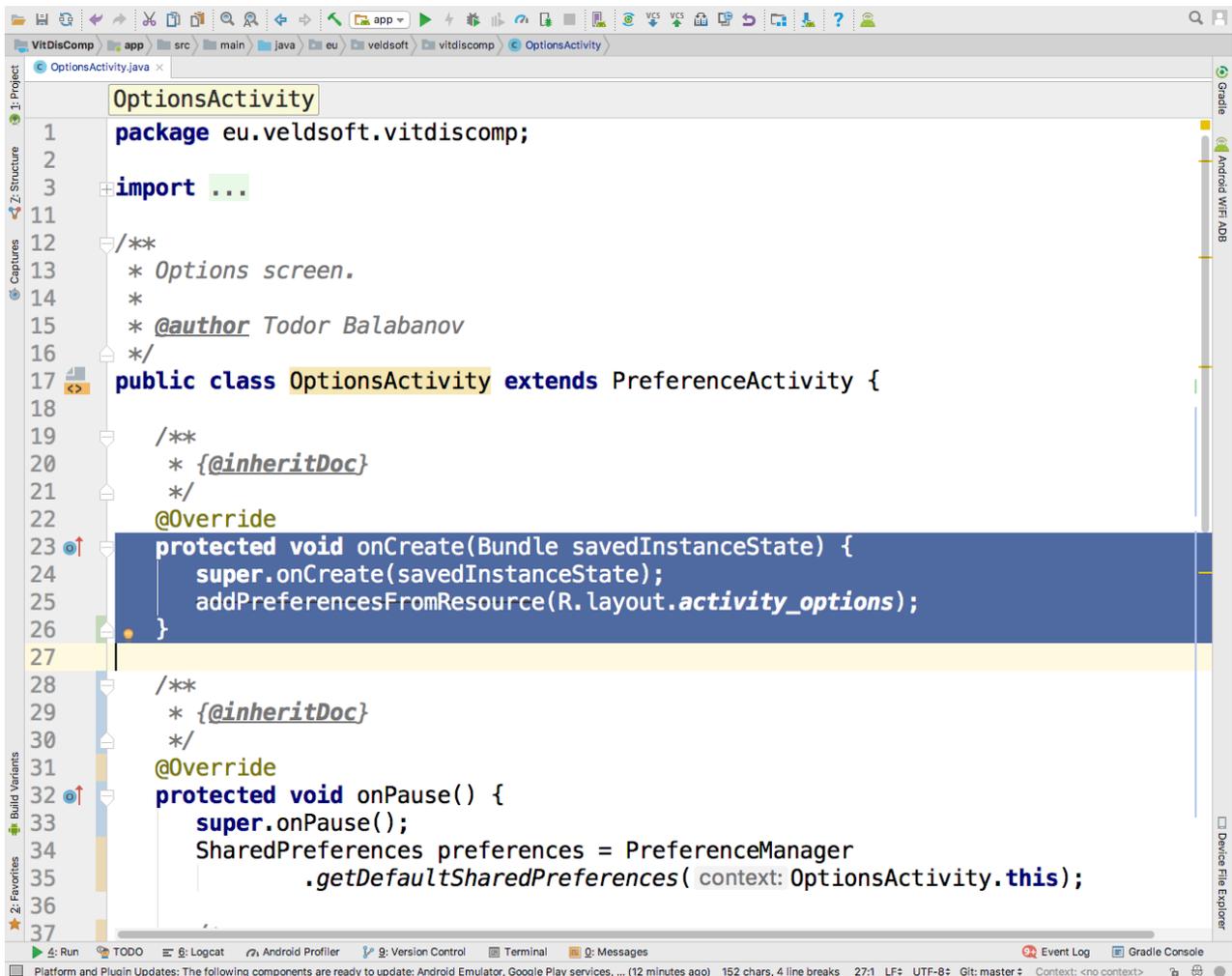


Фигура 4.18: Стойности за размер на областите за визуално представяне

Областите за визуално представяне на информацията се предлагат в три размера – малък, среден и голям (Фиг. 4.18).

## 4.2.2 Програмен код за управление на интерфейса

Събитията, предизвикани от графичния програмен интерфейс, биват прихващани в специално написани за целта Java функции, така че при активирането им да бъдат извършени необходимите програмни действия. За екрана с настройки се прихващат само две събития – създаване и паузиране.



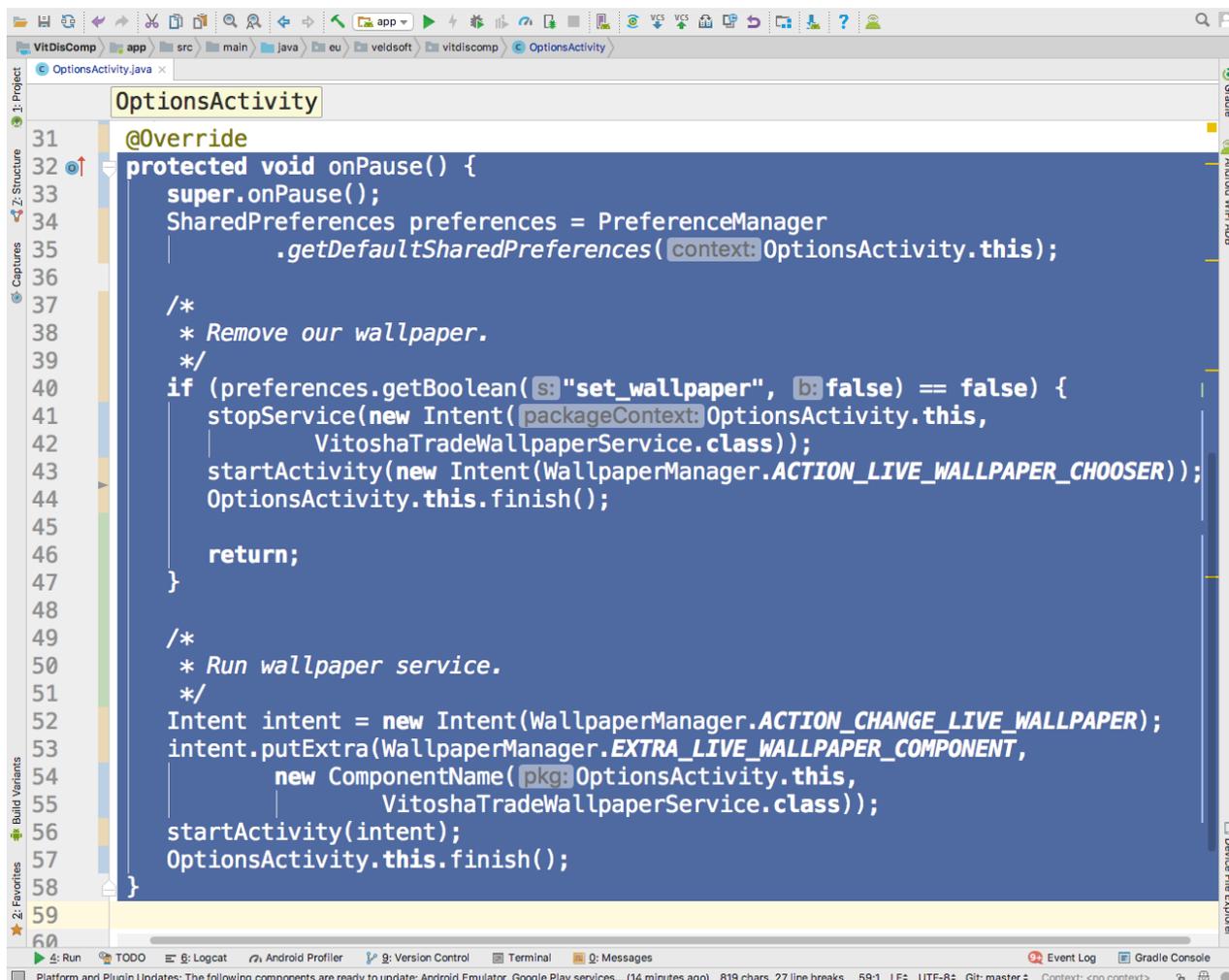
```

1  package eu.veldsoft.vitdiscomp;
2
3  import ...
11
12  /**
13   * Options screen.
14   *
15   * @author Todor Balabanov
16   */
17  public class OptionsActivity extends PreferenceActivity {
18
19     /**
20     * {@inheritDoc}
21     */
22     @Override
23     protected void onCreate(Bundle savedInstanceState) {
24         super.onCreate(savedInstanceState);
25         addPreferencesFromResource(R.layout.activity_options);
26     }
27
28     /**
29     * {@inheritDoc}
30     */
31     @Override
32     protected void onPause() {
33         super.onPause();
34         SharedPreferences preferences = PreferenceManager
35             .getDefaultSharedPreferences(context: OptionsActivity.this);
36
37

```

Фигура 4.19: Събитие за създаване на прозореца

Събитието за създаване има за цел да трансформира XML описанието на интерфейса до визуални компоненти, видими за потребителя (Фиг. 4.19).



Фигура 4.20: Събитие за пауза на прозореца

При събитието за пауза единствено се взема решение дали активният тапет да бъде стартиран или да бъде скрит (Фиг. 4.20).

### 4.3 Пресмятане на фонов режим

Продължителните пресмятания, които не изискват графичен потребителски интерфейс се осъществяват в модули, наречени „услуги“ (Service). Когато става въпрос за активен тапет, е необходимо да се напише собствен клас, който е наследник на класа WallpaperService (Фиг. 4.21).

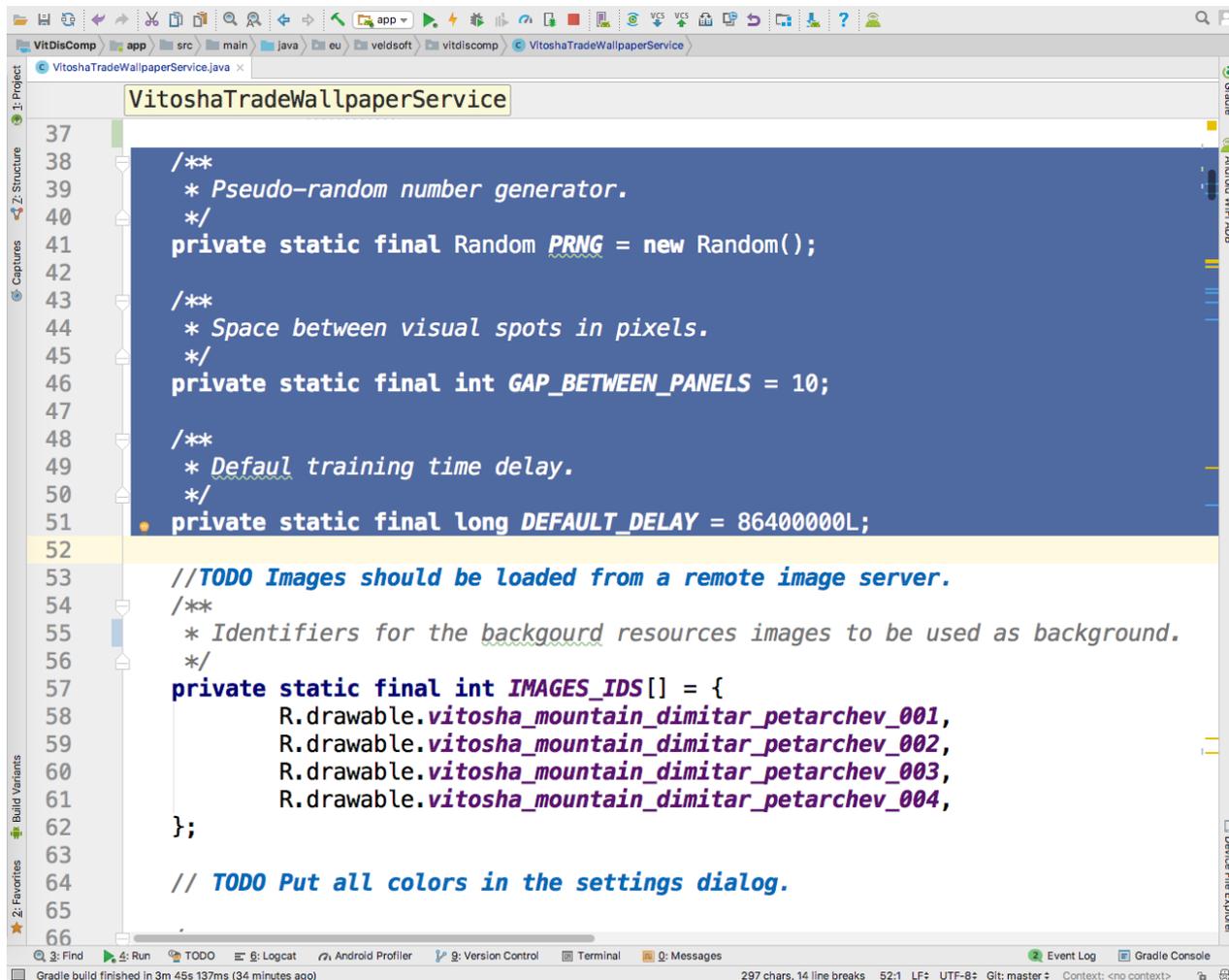
```

30
31 /**
32  * Background calculation unit.
33  *
34  * @author Todor Balabanov
35  */
36 public class VitoshaTradeWallpaperService extends WallpaperService {
37
38     /**
39     * Pseudo-random number generator.
40     */
41     private static final Random PRNG = new Random();
42
43     /**
44     * Space between visual spots in pixels.
45     */
46     private static final int GAP_BETWEEN_PANELS = 10;
47
48     /**
49     * Defaul training time delay.
50     */
51     private static final long DEFAULT_DELAY = 86400000L;
52
53     //TODO Images should be loaded from a remote image server.
54     /**
55     * Identifiers for the backgourd resources images to be used as backgroud.
56     */
57     private static final int IMAGES_IDS[] = {
58         R.drawable.vitosha_mountain_dimitar_petarchev_001,
59

```

Фигура 4.21: Наследяване на WallpaperService

Група от константи спомагат за генерирането на случайни числа, определяне на разстояние между зоните за визуално представяне и подразбираща се стойност за времето между две отделни стартирания на тренировъчния алгоритъм (Фиг. 4.22).



Фигура 4.22: Спомагателни константи

Цветовете, използвани за визуалното представяне, също са зададени с група от константи, но в последствие ще бъдат превърнати в настройки със стойности от прозореца за настройки (Фиг. 4.23).

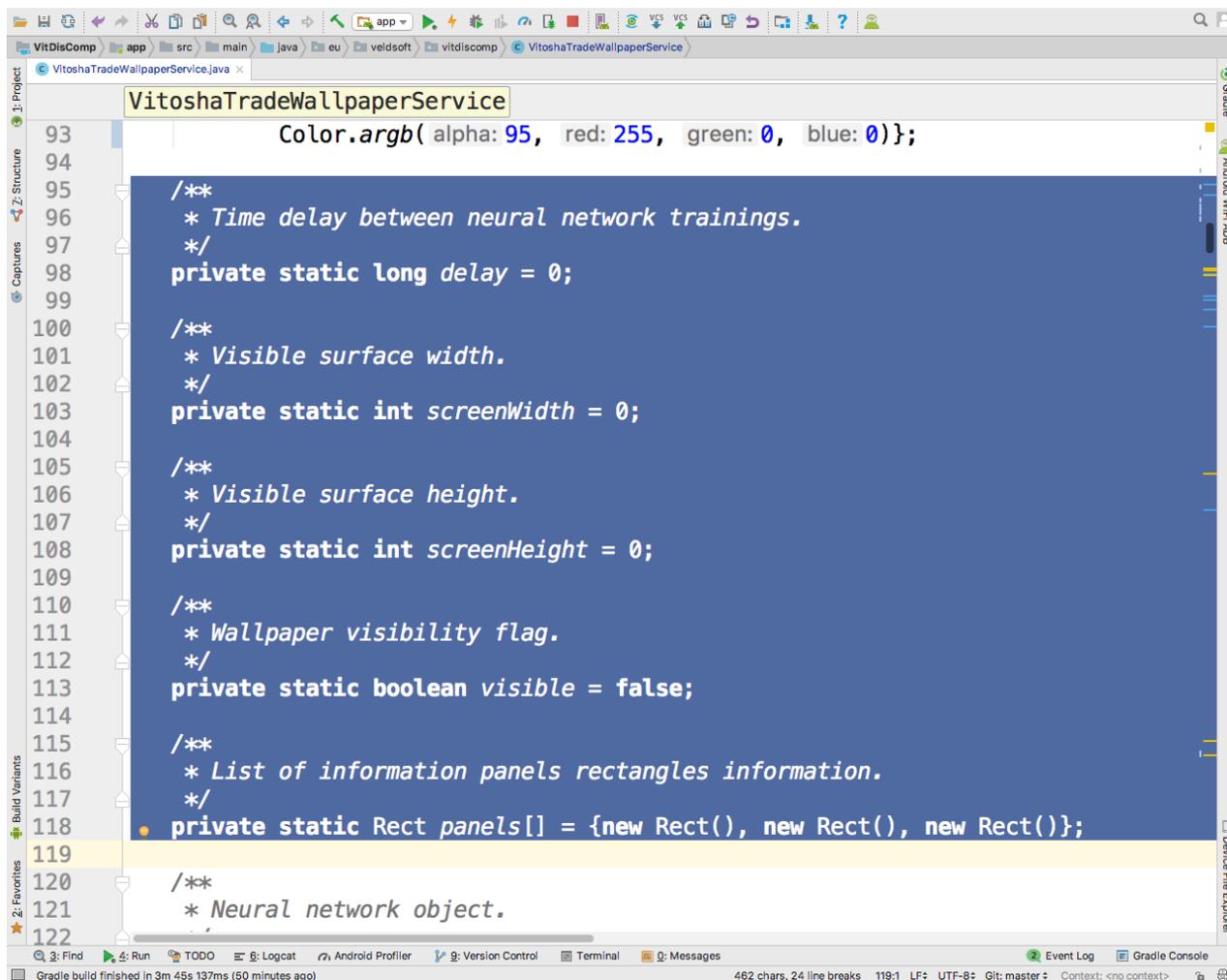
```

65
66 /**
67  * Panel background color in order to be a part transparent from the real backg
68  */
69 private static final int PANEL_BACKGROUND_COLOR =
70     Color.argb(alpha:63, red:0, green:0, blue:0);
71
72 /**
73  * Text color to be used in panels.
74  */
75 private static final int PANEL_TEXT_COLOR =
76     Color.argb(alpha:95, red:255, green:255, blue:255);
77
78 /**
79  * Colors used in the charts.
80  */
81 private static final int CHART_COLORS[] = {
82     Color.argb(alpha:95, red:0, green:255, blue:0),
83     Color.argb(alpha:95, red:255, green:0, blue:0)};
84
85 /**
86  * Colors used to visualize neural networks.
87  */
88 private static final int ANN_COLORS[] = {
89     Color.argb(alpha:95, red:0, green:255, blue:0),
90     Color.argb(alpha:95, red:255, green:255, blue:255),
91     Color.argb(alpha:95, red:0, green:0, blue:255),
92     Color.argb(alpha:95, red:255, green:255, blue:255),
93     Color.argb(alpha:95, red:255, green:0, blue:0)};
94

```

Фигура 4.23: Константи за цветовете

Група променливи отговарят за състоянието на активния тапет. Това включва – размери на екрана, време между отделните обучения на изкуствената невронна мрежа, дали активният тапет е включен или изключен и точната позиция на зоните за визуално представяне на информацията от процеса по обучение на изкуствената невронна мрежа (Фиг. 4.24).



Фигура 4.24: Променливи отразяващи състоянието на активния тапет

Друга група променливи (референции към обекти) поемат отговорността за управлението на изкуствената невронна мрежа, тренировъчните примери, входно-изходните данни към мрежата и правилото за нейното обучение (Фиг. 4.25).

```

118 private static Rect panels[] = {new Rect(), new Rect(), new Rect()};
119
120 /**
121  * Neural network object.
122  */
123 private static BasicNetwork network = new BasicNetwork();
124
125 /**
126  * Training examples data set.
127  */
128 private static MLDataSet examples = null;
129
130 /**
131  * Forecasted data.
132  */
133 private static MLData forecast = null;
134
135 /**
136  * Calculated output data.
137  */
138 private static MLData output = null;
139
140 /**
141  * Training rule object.
142  */
143 private static ResilientPropagation train = null;
144
145 /**
146  * Initialize static class members.
147

```

Фигура 4.25: Променливи отговарящи за изкуствената невронна мрежа

При комерсиалното производство на софтуер много често използван похват е създаването на „софтуерни тапи“. Това са парчета код, които служат за комуникация между обекти и части от системата, които още не са създадени. В настоящия случай точно такава софтуерна тапа представя липсата на сървър и източник с реални данни (Фиг. 4.26).



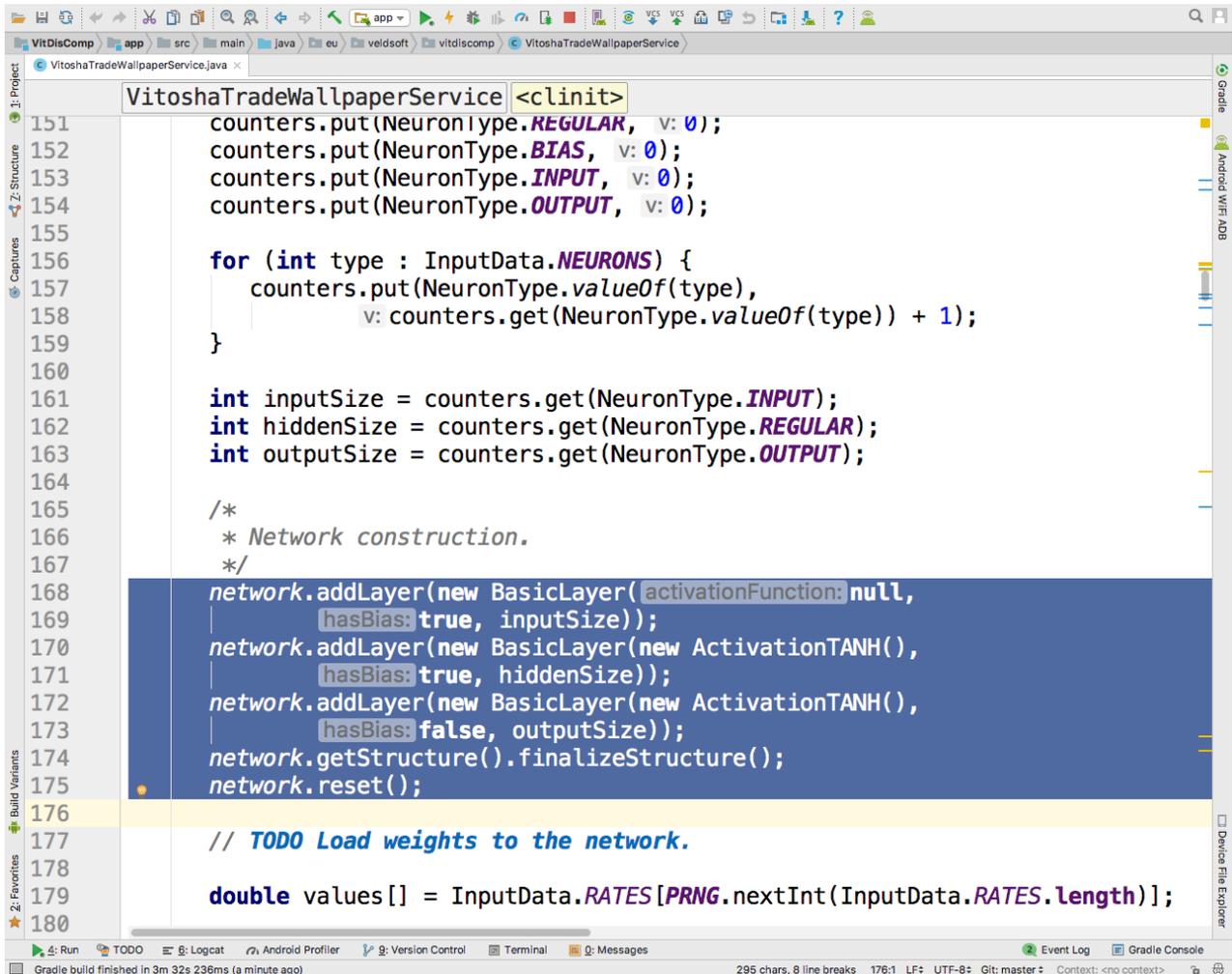
```

143 private static ResilientPropagation train = null;
144
145 /**
146  * Initialize static class members.
147  */
148 static {
149     // TODO Load ANN structure from the remote server.
150     Map<NeuronType, Integer> counters = new HashMap<NeuronType, Integer>();
151     counters.put(NeuronType.REGULAR, 0);
152     counters.put(NeuronType.BIAS, 0);
153     counters.put(NeuronType.INPUT, 0);
154     counters.put(NeuronType.OUTPUT, 0);
155
156     for (int type : InputData.NEURONS) {
157         counters.put(NeuronType.valueOf(type),
158             counters.get(NeuronType.valueOf(type)) + 1);
159     }
160
161     int inputSize = counters.get(NeuronType.INPUT);
162     int hiddenSize = counters.get(NeuronType.REGULAR);
163     int outputSize = counters.get(NeuronType.OUTPUT);
164
165     /*
166     * Network construction.
167     */
168     network.addLayer(new BasicLayer(activationFunction: null,
169         hasBias: true, inputSize));
170     network.addLayer(new BasicLayer(new ActivationTANH(),
171         hasBias: true, hiddenSize));
172
173

```

Фигура 4.27: Определяне на броя и типовете неврони

Чрез сортиране с броеве, ефективно се определя броят на невроните и техните типове (Фиг. 4.27).



Фигура 4.28: Структура на трислойна изкуствена невронна мрежа

За да се илюстрират възможностите за прогнозиране на изкуствените невронни мрежи, е избрана една от най-популярните структури, а именно трислойна невронна мрежа, която се обучава с обратно разпространение на грешката (Фиг. 4.28). Входният слой има единствената задача да получи сигналите от външната среда и поради тази причина не се задава активационна функция. За скрития и изходния слой е избрана функцията  $\tanh(x)$  (хиперболичен тангенс), тъй като тя е асимптотично сходима в безкрайностите по абсцисата и в същото време е симетрична по същата тази ос. Обучението протича малко по-бързо с хиперболичен тангенс, тъй като стойностите на сигналите в скритите слоеве по-лесно се „отклоняват“ в положителна или отрицателна посока. За сравнение, при сигмоидната функция сигналите са само положителни и при нужда от отрицателни стойности тази компенсация трябва да се получава единствено през отрицателни тегла на връзките. Входният и скритият слоеве имат отместващ неврон (bias), който не е нужен в изходния слоеве, тъй като отместващият неврон винаги емитира високото ниво на сигнализация.

```

151 counters.put(NeuronType.REGULAR, v: 0);
152 counters.put(NeuronType.BIAS, v: 0);
153 counters.put(NeuronType.INPUT, v: 0);
154 counters.put(NeuronType.OUTPUT, v: 0);
155
156 for (int type : InputData.NEURONS) {
157     counters.put(NeuronType.valueOf(type),
158                 v: counters.get(NeuronType.valueOf(type)) + 1);
159 }
160
161 int inputSize = counters.get(NeuronType.INPUT);
162 int hiddenSize = counters.get(NeuronType.REGULAR);
163 int outputSize = counters.get(NeuronType.OUTPUT);
164
165 /*
166  * Network construction.
167  */
168 network.addLayer(new BasicLayer( activationFunction: null,
169                                 hasBias: true, inputSize));
170 network.addLayer(new BasicLayer(new ActivationTANH(),
171                                 hasBias: true, hiddenSize));
172 network.addLayer(new BasicLayer(new ActivationTANH(),
173                                 hasBias: false, outputSize));
174 network.getStructure().finalizeStructure();
175 network.reset();
176
177 // TODO Load weights to the network.
178
179 double values[] = InputData.RATES[PRNG.nextInt(InputData.RATES.length)];
180

```

Фигура 4.29: Избор на стойности за прогнозиране

От описанието на паралелните масиви при финансовите времеви редове ясно се вижда, че има четири възможни редици числа, които да се подават за прогнозиране към невронната мрежа (отваряне, най-ниско, най-високо, затваряне).

```

250
251     /*
252     * Normalize data.
253     */
254     double min = values[0];
255     double max = values[0];
256     for (double value : values) {
257         if (value < min) {
258             min = value;
259         }
260         if (value > max) {
261             max = value;
262         }
263     }
264
265     /*
266     * At the first index is the low value. At the second index is the high
267     * value.
268     *
269     * There is a problem with this approach, because some activation
270     * functions are zero if the argument is infinity.
271     *
272     * The fist layer has no activation function.
273     */
274     double range[] = findLowAndHigh(network.getActivation( layer: 2));
275
276     /*
277     * Prepare training set.
278     */
279

```

Фигура 4.30: Определяне на границите във времевия ред

В практиката най-често се използва поредицата за стойности на затваряне, а целта е да се прогнозира нивото на отваряне в следващия времеви интервал. Целите на настоящото изложение не са реално постигнати на финансови прогнози и поради тази причина на случаен принцип се избира коя последователност да бъде използвана при различните стартирания на обучението/прогнозирането (Фиг. 4.29).

```

246 // data.addDescription(description);
247 // data.load(new Date(InputData.TIME[0]), new
248 // Date(InputData.TIME[InputData.TIME.length - 1]));
249 // data.generate();
250
251 /*
252  * Normalize data.
253  */
254 double min = values[0];
255 double max = values[0];
256 for (double value : values) {
257     if (value < min) {
258         min = value;
259     }
260     if (value > max) {
261         max = value;
262     }
263 }
264
265 /*
266  * At the first index is the low value. At the second index is the high
267  * value.
268  *
269  * There is a problem with this approach, because some activation
270  * functions are zero if the argument is infinity.
271  *
272  * The first layer has no activation function.
273  */
274 double range[] = findLowAndHigh(network.getActivation(layer: 2));
275

```

Фигура 4.31: Диапазон на активационната функция в изходния слой

Тъй като целта е към системата за прогнозиране да се подават различни финансови времеви редове, то неизбежно се налага входната информация в системата да бъде предварително нормализирана, а първата стъпка към това е да се определят най-големата и най-малката стойности във времевия ред (Фиг. 4.30). В същото време данните трябва да бъдат съпоставени с диапазона, в който работят активационните функции на изкуствената невронна мрежа. В настоящото помагало е прието, че на входа ще се подадат сигнали в диапазона на функцията, използвана в изходния слой (Фиг. 4.31). Това решение се аргументира с факта, че информацията, подадена от невронната мрежа към външната среда, ще претърпи обратен процес на преоразмеряване, така че да има смисъл в конкретния финансов времеви ред.

```

VitoshaTradeWallpaperService <clinit>
274 double range[] = findLowAndHigh(network.getActivation( layer: 2));
275
276 /*
277  * Prepare training set.
278  */
279 double input[][] = new double[values.length -
280     (inputSize + outputSize)][inputSize];
281 double target[][] = new double[values.length -
282     (inputSize + outputSize)][outputSize];
283 for (int i = 0; i < values.length - (inputSize + outputSize); i++) {
284     for (int j = 0; j < inputSize; j++) {
285         input[i][j] = range[0] + (range[1] - range[0]) *
286             (values[i + j] - min) / (max - min);
287     }
288     for (int j = 0; j < outputSize; j++) {
289         target[i][j] = range[0] + (range[1] - range[0]) *
290             (values[i + inputSize + j] - min) / (max - min);
291     }
292 }
293 examples = new BasicMLDataSet(input, target);
294
295 /*
296  * Prepare forecast set.
297  */
298 input = new double[1][inputSize];
299 for (int j = 0; j < inputSize; j++) {
300     input[0][j] = range[0] + (range[1] - range[0]) *
301         (values[values.length - inputSize + j] - min) / (max - min);
302 }

```

Фигура 4.32: Тренировъчни примери

Нормализираните входни данни условно разделяме на две множества (измервания в миналото и измервания в бъдещето). Разделянето е условно, тъй като практически данните са от реални измервания в миналото (Фиг. 4.32).

```

277     * Prepare training set.
278     */
279     double input[][] = new double[values.length -
280         (inputSize + outputSize)][inputSize];
281     double target[][] = new double[values.length -
282         (inputSize + outputSize)][outputSize];
283     for (int i = 0; i < values.length - (inputSize + outputSize); i++) {
284         for (int j = 0; j < inputSize; j++) {
285             input[i][j] = range[0] + (range[1] - range[0]) *
286                 (values[i + j] - min) / (max - min);
287         }
288         for (int j = 0; j < outputSize; j++) {
289             target[i][j] = range[0] + (range[1] - range[0]) *
290                 (values[i + inputSize + j] - min) / (max - min);
291         }
292     }
293     examples = new BasicMLDataSet(input, target);
294
295     /*
296     * Prepare forecast set.
297     */
298     input = new double[1][inputSize];
299     for (int j = 0; j < inputSize; j++) {
300         input[0][j] = range[0] + (range[1] - range[0]) *
301             (values[values.length - inputSize + j] - min) / (max - min);
302     }
303     forecast = new BasicMLData(input[0]);
304 }
305
306

```

Фигура 4.33: Входни данни към изкуствената невронна мрежа за получаване на прогноза

Тъй като изкуствената невронна мрежа се използва и в двата й режима (обучение и прогнозиране), то се изготвя и вектор входни данни, спрямо които да бъде направена прогноза (Фиг. 4.33).

```

310     * @return Array with two values - lowest in the first index and highest in the last
311     */
312     @private static double[] findLowAndHigh(ActivationFunction activation) {
313         /*
314         * Use range of double values.
315         */
316         double check[] = {
317             Double.MIN_VALUE, -0.000001, -0.00001, -0.0001,
318             -0.001, -0.01, -0.1, -1, -10, -100, -1000,
319             -10000, -100000, -1000000, 0, 0.000001, 0.00001,
320             0.0001, 0.001, 0.01, 0.1, 1, 10, 100, 1000, 10000,
321             100000, 1000000, Double.MAX_VALUE};
322
323         /*
324         * Map the range of double values to activation function output.
325         */
326         activation.activationFunction(check, 0, check.length);
327
328         /*
329         * Sort the result of the activation function output.
330         */
331         Arrays.sort(check);
332
333         /*
334         * Return minimum and maximum values of the activation function output.
335         */
336         return new double[]{check[0], check[check.length - 1]};
337     }
338
339

```

Фигура 4.34: Изследване на група от стойности за определяне на диапазона

В общия случай активационните функции имат асимптотична сходимост и са монотонно нарастващи (Фиг. 4.35), което позволява техните крайни диапазони да бъдат установявани, чрез проверка на група от стойности (Фиг. 4.34).

Name	Plot	Equation	Derivative (with respect to $x$ )	Range
Identity		$f(x) = x$	$f'(x) = 1$	$(-\infty, \infty)$
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}$	$\{0, 1\}$
Logistic (a.k.a. Soft step)		$f(x) = \frac{1}{1 + e^{-x}}$	$f'(x) = f(x)(1 - f(x))$	$(0, 1)$
TanH		$f(x) = \tanh(x) = \frac{2}{1 + e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$	$(-1, 1)$
ArcTan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2 + 1}$	$(-\frac{\pi}{2}, \frac{\pi}{2})$
Softsign [7][8]		$f(x) = \frac{x}{1 +  x }$	$f'(x) = \frac{1}{(1 +  x )^2}$	$(-1, 1)$
Rectified linear unit (ReLU) <sup>[9]</sup>		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$[0, \infty)$
Leaky rectified linear unit (Leaky ReLU) <sup>[10]</sup>		$f(x) = \begin{cases} 0.01x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0.01 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$(-\infty, \infty)$
Parameteric rectified linear unit (PReLU) <sup>[11]</sup>		$f(\alpha, x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(\alpha, x) = \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$(-\infty, \infty)$
Randomized leaky rectified linear unit (RRReLU) <sup>[12]</sup>		$f(\alpha, x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$ [1]	$f'(\alpha, x) = \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$(-\infty, \infty)$
Exponential linear unit (ELU) <sup>[13]</sup>		$f(\alpha, x) = \begin{cases} \alpha(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(\alpha, x) = \begin{cases} f(\alpha, x) + \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$(-\alpha, \infty)$
Scaled exponential linear unit (SELU) <sup>[14]</sup>		$f(\alpha, x) = \lambda \begin{cases} \alpha(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$ with $\lambda = 1.0507$ and $\alpha = 1.67326$	$f'(\alpha, x) = \lambda \begin{cases} \alpha e^x & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$(-\lambda\alpha, \infty)$
S-shaped rectified linear activation unit (SReLU) <sup>[15]</sup>		$f_{t_l, a_l, t_r, a_r}(x) = \begin{cases} t_l + a_l(x - t_l) & \text{for } x \leq t_l \\ x & \text{for } t_l < x < t_r \\ t_r + a_r(x - t_r) & \text{for } x \geq t_r \end{cases}$ $t_l, a_l, t_r, a_r$ are parameters.	$f'_{t_l, a_l, t_r, a_r}(x) = \begin{cases} a_l & \text{for } x \leq t_l \\ 1 & \text{for } t_l < x < t_r \\ a_r & \text{for } x \geq t_r \end{cases}$	$(-\infty, \infty)$
Adaptive piecewise linear (APL) <sup>[16]</sup>		$f(x) = \max(0, x) + \sum_{i=1}^S a_i^+ \max(0, -x + b_i^+)$	$f'(x) = H(x) - \sum_{i=1}^S a_i^+ H(-x + b_i^+)$ [2]	$(-\infty, \infty)$
SoftPlus <sup>[17]</sup>		$f(x) = \ln(1 + e^x)$	$f'(x) = \frac{1}{1 + e^{-x}}$	$(0, \infty)$
Bent identity		$f(x) = \frac{\sqrt{x^2 + 1} - 1}{2} + x$	$f'(x) = \frac{x}{2\sqrt{x^2 + 1}} + 1$	$(-\infty, \infty)$
SoftExponential <sup>[18]</sup>		$f(\alpha, x) = \begin{cases} -\frac{\ln(1 - \alpha(x + \alpha))}{\alpha} & \text{for } \alpha < 0 \\ x & \text{for } \alpha = 0 \\ \frac{e^{\alpha x} - 1}{\alpha} + \alpha & \text{for } \alpha > 0 \end{cases}$	$f'(\alpha, x) = \begin{cases} \frac{1}{1 - \alpha(x + \alpha)} & \text{for } \alpha < 0 \\ e^{\alpha x} & \text{for } \alpha \geq 0 \end{cases}$	$(-\infty, \infty)$
Sinusoid <sup>[19]</sup>		$f(x) = \sin(x)$	$f'(x) = \cos(x)$	$[-1, 1]$
Sinc		$f(x) = \begin{cases} 1 & \text{for } x = 0 \\ \frac{\sin(x)}{x} & \text{for } x \neq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x = 0 \\ \frac{\cos(x)}{x} - \frac{\sin(x)}{x^2} & \text{for } x \neq 0 \end{cases}$	$[\approx -0.217234, 1]$
Gaussian		$f(x) = e^{-x^2}$	$f'(x) = -2xe^{-x^2}$	$(0, 1]$

Фигура 4.35: Активационни функции [5]

Както е добре видно на Фиг. 4.35, за някои от активационните функции такова определяне на диапазона може да се окаже силно подвеждащо (примерно при синус функцията).

```

332
333
334     /*
335     * Return minimum and maximum values of the activation function output.
336     */
337     return new double[]{check[0], check[check.length - 1]};
338 }
339
340 /**
341 * {@inheritDoc}
342 */
343 @Override
344 public Engine onCreateEngine() {
345     return new WallpaperEngine();
346 }
347
348 /**
349 * Wallpaper engine class.
350 *
351 * @author Todor Balabanov
352 */
353 private class WallpaperEngine extends Engine {
354     /**
355     * Thread handler.
356     */
357     private final Handler handler = new Handler();
358
359     /**
360     * Paint object.
361     */

```

Фигура 4.36: Собствена реализация на наследения Engine клас и създаване на обект

Събитието за създаване на engine обект се предефинира, така че да се връща обект от нашия собствен клас за engine (Фиг. 4.36).

## 4.4 Двигател на услугата

Същинската работа за фоновото пресмятане в услугата се извършва от обект, описан с частен вътрешен клас от тип „двигател“ (Engine).

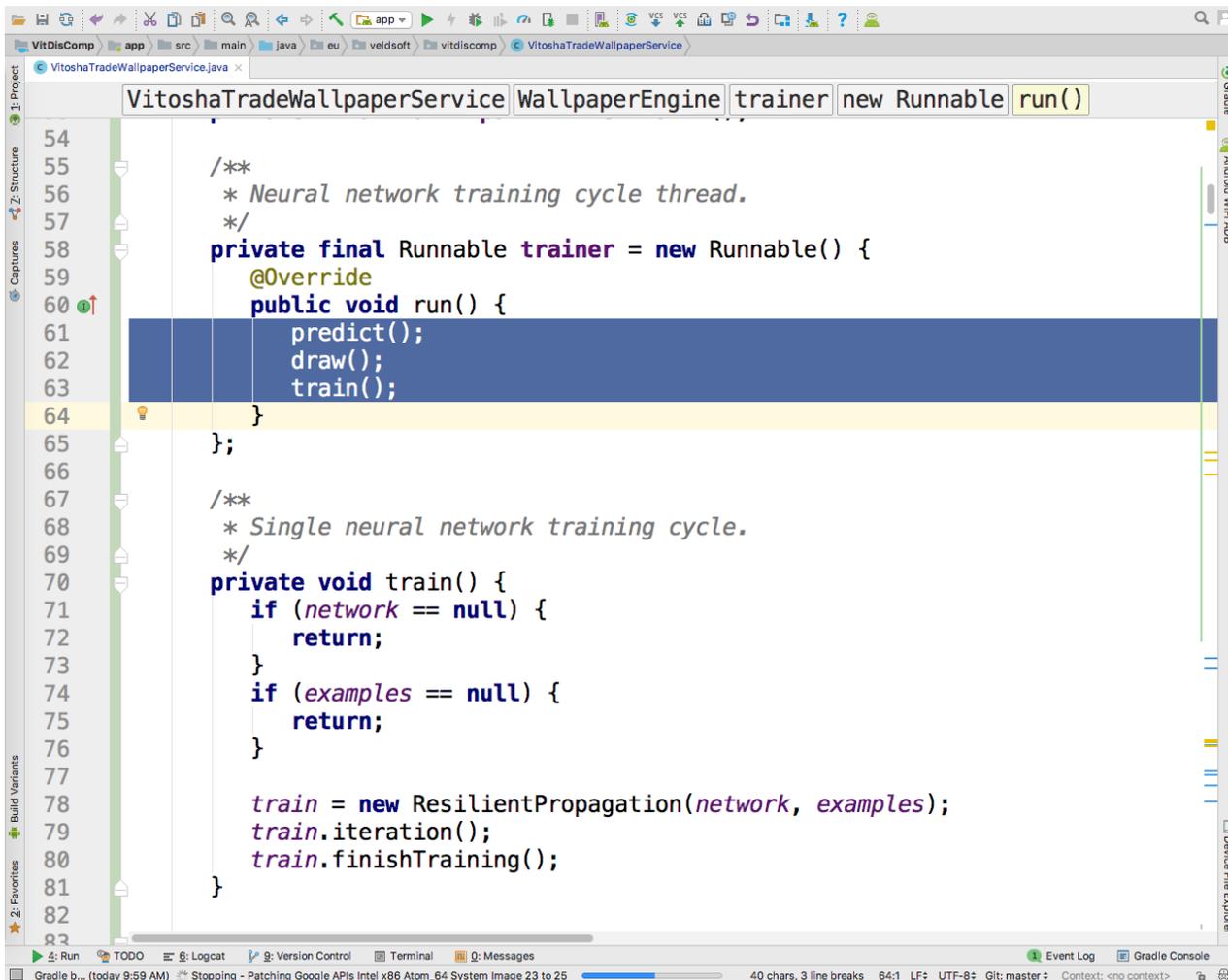
```

38  /**
39  * Wallpaper engine class.
40  *
41  * @author Todor Balabanov
42  */
43  private class WallpaperEngine extends Engine {
44
45      /**
46       * Thread handler.
47       */
48      private final Handler handler = new Handler();
49
50      /**
51       * Paint object.
52       */
53      private final Paint paint = new Paint();
54
55      /**
56       * Neural network training cycle thread.
57       */
58      private final Runnable trainer = new Runnable() {
59          @Override
60          public void run() {
61              predict();
62              draw();
63              train();
64          }
65      };
66
67

```

Фигура 4.37: Вътрешни променливи за двигателя на услугата

В този двигател се използват три вътрешни променливи (Фиг. 4.37). Обектът от тип „четка“ (Paint) е само спомагателен и служи за еднократно определяне на характеристиките за изрисване. Ползата е, че обектът бива създаден веднъж и след това се ползва през целия жизнен цикъл на услугата.



Фигура 4.38: Единичен цикъл във фонов режим

За реалната работа на двигателя се дефинира отделна нишка (`trainer`), която да извършва операциите по генериране на прогноза, изчертаване на активния тапет и извършването на един цикъл от обучаващия процес на изкуствената невронна мрежа (Фиг. 4.38). Ефективното управление на нишки в Android се извършва с помощта на обекти „държател“ (Handler). Обектът държател поема отговорността по стартирането на нишката, спирането на нишката и реализацията на интервала за изчакване преди да бъде осъществено следващото събуждане на нишката.

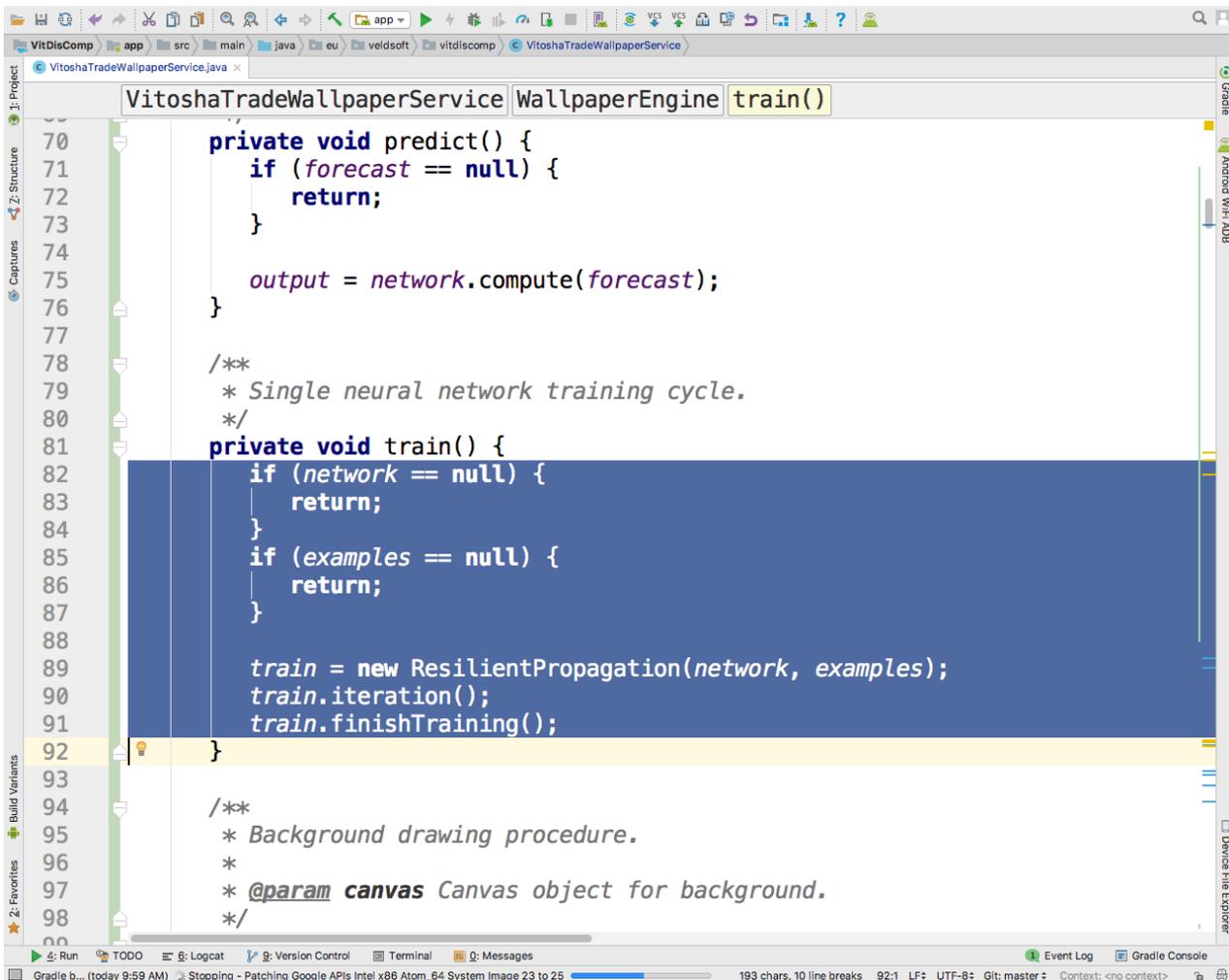
```

57     */
58     private final Runnable trainer = new Runnable() {
59         @Override
60         public void run() {
61             predict();
62             draw();
63             train();
64         }
65     };
66
67     /**
68      * Neural network prediction getter.
69     */
70     private void predict() {
71         if (forecast == null) {
72             return;
73         }
74
75         output = network.compute(forecast);
76     }
77
78     /**
79      * Single neural network training cycle.
80     */
81     private void train() {
82         if (network == null) {
83             return;
84         }
85         if (examples == null) {
86

```

Фигура 4.39: Изчисляване на прогноза

За да се изчисли прогноза, според текущото ниво на обученост на изкуствената невронна мрежа, е достатъчно мрежата да се активира в работен режим с подходящи данни към входния слой (Фиг. 4.39).



```

VitoshaTradeWallpaperService WallpaperEngine train()
70 private void predict() {
71     if (forecast == null) {
72         return;
73     }
74
75     output = network.compute(forecast);
76 }
77
78 /**
79  * Single neural network training cycle.
80  */
81 private void train() {
82     if (network == null) {
83         return;
84     }
85     if (examples == null) {
86         return;
87     }
88
89     train = new ResilientPropagation(network, examples);
90     train.iteration();
91     train.finishTraining();
92 }
93
94 /**
95  * Background drawing procedure.
96  *
97  * @param canvas Canvas object for background.
98  */

```

Фигура 4.40: Тренировъчен цикъл на изкуствената невронна мрежа

За да се осъществи един тренировъчен цикъл на изкуствената невронна мрежа, е достатъчно да се създаде тренировъчен обект (в случая от еластично обратно разпространение на грешката), към който се подават мрежата и тренировъчните примери (Фиг. 4.40).

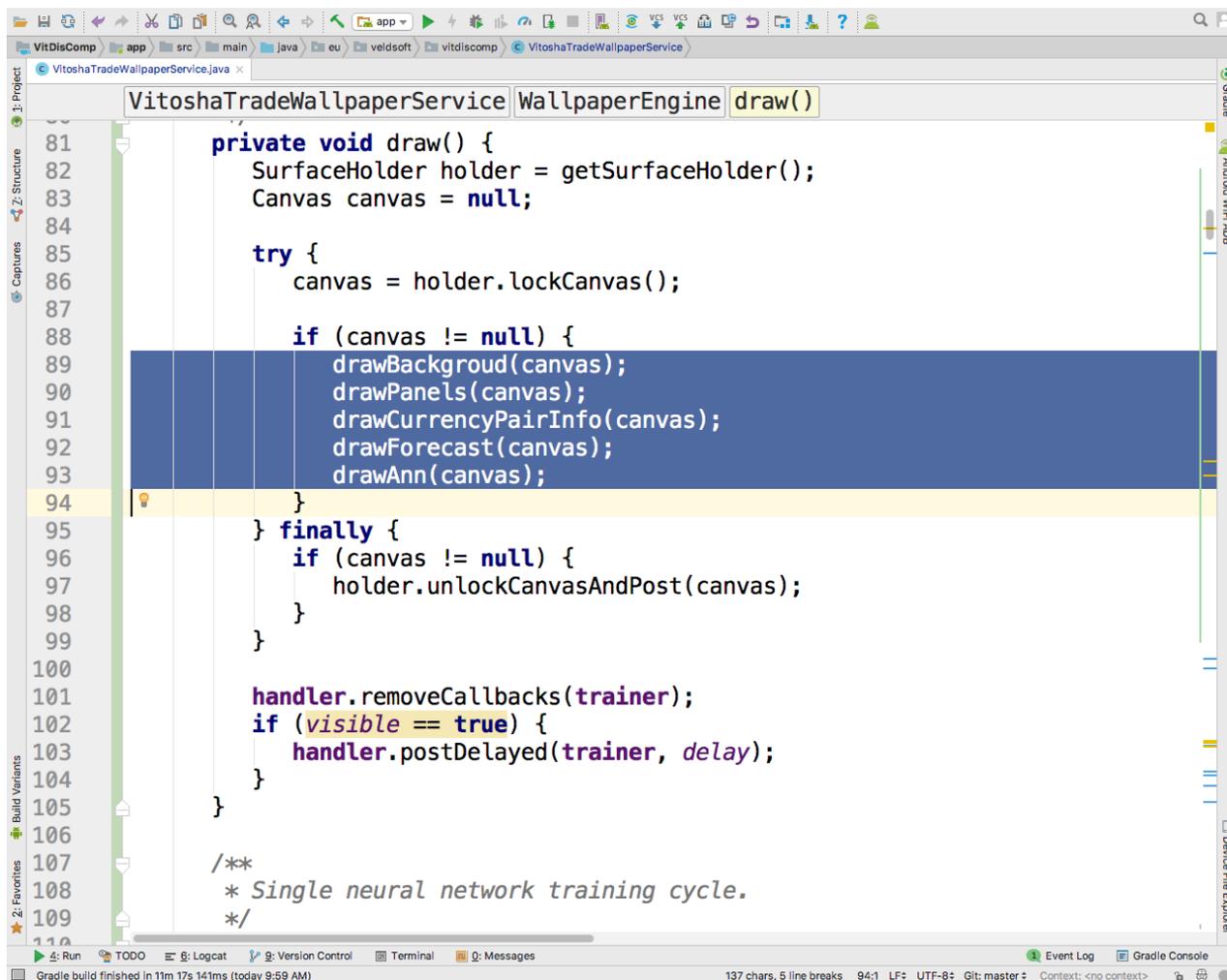
```

77
78 /**
79  * Common drawing procedure.
80  */
81 private void draw() {
82     SurfaceHolder holder = getSurfaceHolder();
83     Canvas canvas = null;
84
85     try {
86         canvas = holder.lockCanvas();
87
88         if (canvas != null) {
89             drawBackground(canvas);
90             drawPanels(canvas);
91             drawCurrencyPairInfo(canvas);
92             drawForecast(canvas);
93             drawAnn(canvas);
94         }
95     } finally {
96         if (canvas != null) {
97             holder.unlockCanvasAndPost(canvas);
98         }
99     }
100
101     handler.removeCallbacks(trainer);
102     if (visible == true) {
103         handler.postDelayed(trainer, delay);
104     }
105 }
106

```

Фигура 4.41: Основна процедура по изрисване на информацията от тренировъчния процес

Изрисването на информацията от тренировъчния процес е организирано в отделна функция (Фиг. 4.41). За да бъде осъществена визуалното представяне, се осигурява достъп до повърхността и платното, което тя съдържа. В самия край на функцията за визуално представяне се взема решение дали ще се изпълни следващ цикъл на обучение или изпълнението ще бъде преустановено.



Фигура 4.42: Разбиване на задачата за визуално представяне

Добрите практики за писане на програмен код включват разбиване на една сложна задача в група от множество по-прости задачи. Този принцип е приложен по отношение на визуалното представяне като са създадени пет помощни функции (Фиг. 4.42). Първата функция изрисува фон, втората очертава полупрозрачни области за панелите, третата запълва панела с информация за валутната двойка, четвъртата показва информация в панела за текущо актуалната прогноза, а петата стилизиран модел на изкуствената невронна мрежа.

```

123  /**
124   * Background drawing procedure.
125   *
126   * @param canvas Canvas object for background.
127   */
128  private void drawBackground(Canvas canvas) {
129      // TODO Images should be loaded from an image server.
130      /*
131       * Change picture according the day in the year.
132       */
133      Bitmap image = BitmapFactory.decodeResource(
134          VitoshTradeWallpaperService.this.getResources(),
135          IMAGES_IDS[Calendar.getInstance().
136              get(Calendar.DAY_OF_YEAR) % IMAGES_IDS.length]);
137
138      /*
139       * Select random top-left corner for image clip.
140       */
141      int left = PRNG.nextInt(n: image.getWidth() - screenWidth);
142      int top = PRNG.nextInt(n: image.getHeight() - screenHeight);
143
144      /*
145       * Clip part of the image.
146       */
147      canvas.drawBitmap(image, new Rect(left, top,
148          right: left + screenWidth - 1,
149          bottom: top + screenHeight - 1),
150          new Rect(left: 0, top: 0, right: screenWidth - 1,
151          bottom: screenHeight - 1), paint: null);
152  }
    
```

Фигура 4.43: Изрисуване на фона

За да бъде изрисуван фонът, се зарежда предварително подготвено изображение и от него се отрязва такова парче, че да пасва на размерите на екрана (Фиг. 4.43). Коя част от изображението да бъде използвана се определя на случаен принцип.

```

149         bottom: top + screenHeight - 1),
150         new Rect( left: 0, top: 0, right: screenWidth - 1,
151                 bottom: screenHeight - 1), paint: null);
152     }
153
154     /**
155     * Panels drawing procedure.
156     *
157     * @param canvas Canvas object for panels drawing.
158     */
159     private void drawPanels(Canvas canvas) {
160         /*
161         * Panels.
162         */
163         paint.setColor(PANEL_BACKGROUND_COLOR);
164         for (Rect rectangle : panels) {
165             canvas.drawRect(rectangle, paint);
166         }
167     }
168
169     /**
170     * Currency pair info drawing procedure.
171     *
172     * @param canvas Canvas object for currency pair info drawing.
173     */
174     private void drawCurrencyPairInfo(Canvas canvas) {
175         /*
176         * Time series info.
177         */
178

```

Фигура 4.44: Изрисуване на областите за служебна информация

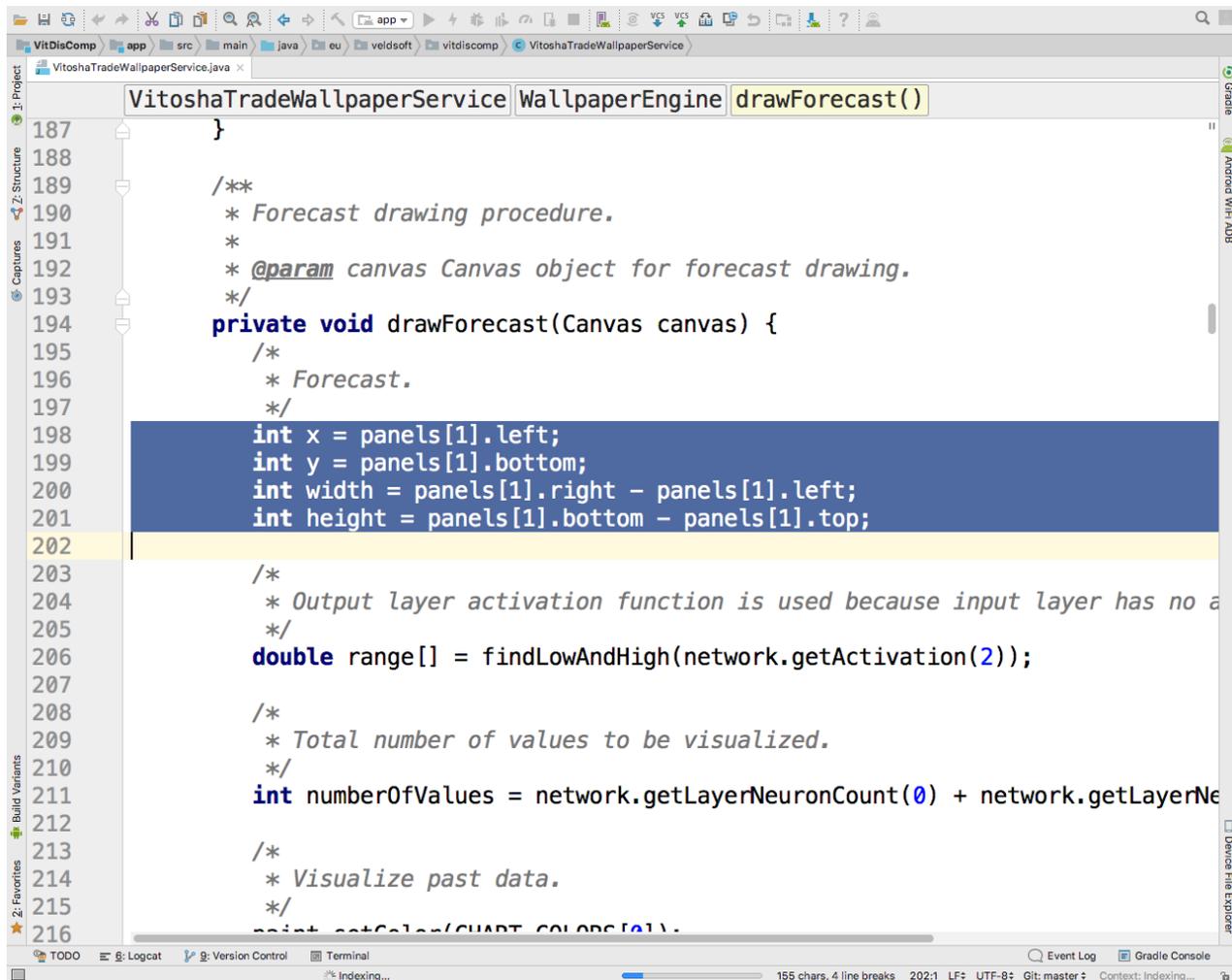
Веднага след изрисуването на фона се изрисуват и полупрозрачните петна за визуално представяне на информацията от процеса по обучението (Фиг. 4.44). Похватът за полупрозрачност е нужен, за да се избегне страничният ефект от сливане на цветове между служебната информация и фона.

```

163     paint.setColor(PANEL_BACKGROUND_COLOR);
164     for (Rect rectangle : panels) {
165         canvas.drawRect(rectangle, paint);
166     }
167 }
168
169 /**
170  * Currency pair info drawing procedure.
171  *
172  * @param canvas Canvas object for currency pair info drawing.
173  */
174 private void drawCurrencyPairInfo(Canvas canvas) {
175     /*
176     * Time series info.
177     */
178     int textSize = (panels[0].bottom - panels[0].top) / 5;
179     paint.setTextSize(textSize);
180     paint.setColor(PANEL_TEXT_COLOR);
181     canvas.drawText(text: "" + InputData.SYMBOL,
182                   x: GAP_BETWEEN_PANELS + panels[0].left,
183                   y: GAP_BETWEEN_PANELS + panels[0].top + textSize, paint);
184     canvas.drawText(text: "" + TimePeriod.value(InputData.PERIOD),
185                   x: GAP_BETWEEN_PANELS + panels[0].left,
186                   y: GAP_BETWEEN_PANELS + panels[0].top + 2 * textSize, paint);
187 }
188
189 /**
190  * Forecast drawing procedure.
191  *
192  */
    
```

Фигура 4.45: Изрисуване на информацията за валутната двойка

Информацията за валутната двойка се състои от название и времеви интервал на финансовия времеви ред (Фиг. 4.45).



Фигура 4.46: Позиция и размери за визуално представяне на прогнозата

Визуалното представяне на прогнозата изисква малко по-сложни пресмятания, така че информацията от входа и изхода да се поместят в рамките на определеното петно. За да бъде изпълнена тази задача, се вземат координатите и размерите на определената област (Фиг. 4.46).

```

193     */
194     private void drawForecast(Canvas canvas) {
195         /*
196          * Forecast.
197          */
198         int x = panels[1].left;
199         int y = panels[1].bottom;
200         int width = panels[1].right - panels[1].left;
201         int height = panels[1].bottom - panels[1].top;
202
203         /*
204          * Output layer activation function is used because input layer
205          * has no activation function.
206          */
207         double range[] = findLowAndHigh(network.getActivation(layer:2));
208
209         /*
210          * Total number of values to be visualized.
211          */
212         int numberOfValues = network.getLayerNeuronCount(0) +
213             network.getLayerNeuronCount(2);
214
215         /*
216          * Visualize past data.
217          */
218         paint.setColor(CHART_COLORS[0]);
219         for (int i = 0; forecast.getData() != null &&
220             i < forecast.getData().length; i++) {
221             int offset = (int) (height * (forecast.getData()[i] - range[0]) /
222
    
```

Фигура 4.47: Обхват на данните по ширина и височина

За да бъдат визуализирани данните, е от съществено значение да се определи минималната и максималната стойности за вход-изхода, както и броят стойности, които ще бъдат визуализирани (Фиг. 4.47). В случая броят стойности за визуално представяне съвпада със сумата от броя входни въздействия и броя изходни сигнали.

```

214
215     /*
216     * Visualize past data.
217     */
218     paint.setColor(CHART_COLORS[0]);
219     for (int i = 0; forecast.getData() != null &&
220         i < forecast.getData().length; i++) {
221         int offset = (int) (height * (forecast.getData()[i] - range[0]) /
222             (range[1] - range[0]));
223         for (int dx = 0; dx < width / numberOfValues; dx++) {
224             canvas.drawLine(x, y, x, stopY: y - offset, paint);
225             x++;
226         }
227     }
228
229     /*
230     * Visualize future data.
231     */
232     paint.setColor(CHART_COLORS[1]);
233     for (int i = 0; output.getData() != null &&
234         i < output.getData().length; i++) {
235         int offset = (int) (height * (output.getData()[i] - range[0]) /
236             (range[1] - range[0]));
237         for (int dx = 0; dx < width / numberOfValues; dx++) {
238             canvas.drawLine(x, y, x, stopY: y - offset, paint);
239             x++;
240         }
241     }
242 }
243

```

Фигура 4.48: Цикли за визуално представяне на стълбовете от времеви реди и прогнозата

Данните за отминалия период от време се показват с един цвят, а данните за прогнозата в друг цвят (Фиг. 4.48). Самата визуално представяне представлява стълбова диаграма (bar-chart) на входните данни и на прогнозата.

```

240     }
241   }
242 }
243
244 /**
245  * Neural networ drawing procedure.
246  *
247  * @param canvas Canvas object for neural network drawing.
248  */
249 private void drawAnn(Canvas canvas) {
250     /*
251     * Artificial neural network.
252     */
253     double topology[][] = {
254         forecast.getData(),
255         new double[network.getLayerNeuronCount(0) *
256             network.getLayerNeuronCount(1)],
257         new double[network.getLayerNeuronCount(1)],
258         new double[network.getLayerNeuronCount(1) *
259             network.getLayerNeuronCount(2)],
260         output.getData()
261     };
262
263     /*
264     * At the first index is the low value. At the second index is the high
265     * value.
266     *
267     * There is a problem with this approach, because some activation
268     * functions are zero if the argument is infinity.
269     */

```

Фигура 4.49: Топология на изкуствената невронна мрежа, която се визуализира

В третата област за визуално представяне се изрисова стилизирана информация за състоянието на изкуствената невронна мрежа. Областта е правоъгълна и бива разделена на пет вертикални зони: 1. Стойности на невроните във входния слой; 2. Стойности на теглата между входния и скрития слой; 3. Стойности на невроните в скрития слой; 4. Стойности на теглата между скрития и изходния слой; 5. Стойности на невроните в изходния слой (Фиг. 4.49).

```

VitoshaTradeWallpaperService WallpaperEngine drawAnn()
261     };
262
263     /*
264     * At the first index is the low value. At the second index is
265     * the high value.
266     *
267     * There is a problem with this approach, because some activation
268     * functions are zero if the argument is infinity.
269     *
270     * The first layer has no activation function.
271     */
272     double range[] = findLowAndHigh(network.getActivation(layer: 2));
273
274     /*
275     * Scale input layer data.
276     */
277     for (int i = 0; i < topology[0].length; i++) {
278         topology[0][i] = (topology[0][i] - range[0]) /
279             (range[1] - range[0]);
280     }
281
282     /*
283     * Scale output layer data.
284     */
285     for (int i = 0; i < topology[4].length; i++) {
286         topology[4][i] = (topology[4][i] - range[0]) /
287             (range[1] - range[0]);
288     }
289
290     for (int i = 0, m = 0, n = 0; i < topology[1].length; i++) {

```

Фигура 4.50: Машабирание на входната и изходната информация

Информацията за входа и изхода се преоразмерява спрямо минималните и максималните стойности с които работят невроните в изходния слой (Фиг. 4.50). Приложена е формулата за MinMax Scaling (4.1).

$$X_{sc} = \frac{X - X_{min}}{X_{max} - X_{min}} \quad (4.1)$$

Невроните във входния слой на практика не би трябвало да имат ограничение от активационна функция, тъй като те само събират информацията от външния свят.

```

286         topology[4][i] = (topology[4][i] - range[0]) /
287             (range[1] - range[0]);
288     }
289
290     for (int i = 0, m = 0, n = 0; i < topology[1].length; i++) {
291         if (n >= network.getLayerNeuronCount(1)) {
292             n = 0;
293             m++;
294         }
295         if (m >= network.getLayerNeuronCount(0)) {
296             m = 0;
297         }
298         topology[1][i] = network.getWeight(0, m, n);
299         n++;
300     }
301
302     for (int i = 0, m = 0, n = 0; i < topology[3].length; i++) {
303         if (n >= network.getLayerNeuronCount(2)) {
304             n = 0;
305             m++;
306         }
307         if (m >= network.getLayerNeuronCount(1)) {
308             m = 0;
309         }
310         topology[3][i] = network.getWeight(1, m, n);
311         n++;
312     }
313
314     /*
315

```

Фигура 4.51: Стойности на теглата между слоевете

Следва определянето на теглата между слоевете (Фиг. 4.51).

```

305         m++;
306     }
307     if (m >= network.getLayerNeuronCount(!:1)) {
308         m = 0;
309     }
310     topology[3][i] = network.getWeight(fromLayer:1, m, n);
311     n++;
312 }
313
314 /*
315  * Hidden layer values. Activation function of the second layer
316  * is used for scaling.
317  */
318 range = findLowAndHigh(network.getActivation(layer:1));
319 for (int i = 0; i < topology[2].length; i++) {
320     topology[2][i] = (network.getLayerOutput(layer:1, i) - range[0]) /
321         (range[1] - range[0]);
322 }
323
324 /*
325  * Normalize weights.
326  */
327 double min = Double.MAX_VALUE;
328 double max = Double.MIN_VALUE;
329 for (double value : topology[1]) {
330     if (value < min) {
331         min = value;
332     }
333     if (value > max) {
334

```

Фигура 4.52: Мащабиране на стойностите в скрития слой

И финално за средната вертикална област се определят стойностите за скрития слой, мащабирани спрямо минималното и максималното ниво на активационната функция, приложена в него (Фиг. 4.52).

```

323
324      /*
325       * Weights normalization.
326       */
327      double min = Double.MAX_VALUE;
328      double max = Double.MIN_VALUE;
329      for (double value : topology[1]) {
330          if (value < min) {
331              min = value;
332          }
333          if (value > max) {
334              max = value;
335          }
336      }
337      for (double value : topology[3]) {
338          if (value < min) {
339              min = value;
340          }
341          if (value > max) {
342              max = value;
343          }
344      }
345      for (int i = 0; i < topology[1].length; i++) {
346          topology[1][i] = (topology[1][i] - min) / (max - min);
347      }
348      for (int i = 0; i < topology[3].length; i++) {
349          topology[3][i] = (topology[3][i] - min) / (max - min);
350      }
351
352

```

Фигура 4.53: Нормализиране на стойностите за теглата

За да бъде визуализирана информацията за теглата, е нужно стойностите им да се нормализират спрямо най-малката и най-голямата стойност измежду тях (Фиг. 4.53).

```

350 }
351
352 /*
353  * Draw topology.
354  */
355 int width = panels[2].right - panels[2].left;
356 int height = panels[2].bottom - panels[2].top;
357 for (int x = panels[2].left, k = 0; k < ANN_COLORS.length;
358      x += width / ANN_COLORS.length, k++) {
359     for (int dx = 0; dx < width / ANN_COLORS.length; dx++) {
360         for (int y = panels[2].top, l = 0; y < panels[2].bottom &&
361              l < topology[k].length; y += height / topology[k].length, l++) {
362             for (int dy = 0; dy < height / topology[k].length; dy++) {
363                 paint.setColor(ANN_COLORS[k]);
364                 paint.setColor(Color.argb(Color.alpha(ANN_COLORS[k]),
365                                           (int) (Color.red(ANN_COLORS[k]) * topology[k][l]),
366                                           (int) (Color.green(ANN_COLORS[k]) * topology[k][l]),
367                                           (int) (Color.blue(ANN_COLORS[k]) * topology[k][l]))));
368                 canvas.drawPoint(x + dx, y + dy, paint);
369             }
370         }
371     }
372 }
373
374
375 /*
376  * Constructor without parameters.
377  */
378 public WallpaperEngine() {
379

```

Фигура 4.54: Изрисуване на топологията

Така подготвените предварително данни с лекота биват визуализирани от група вложени цикли (Фиг. 4.54).

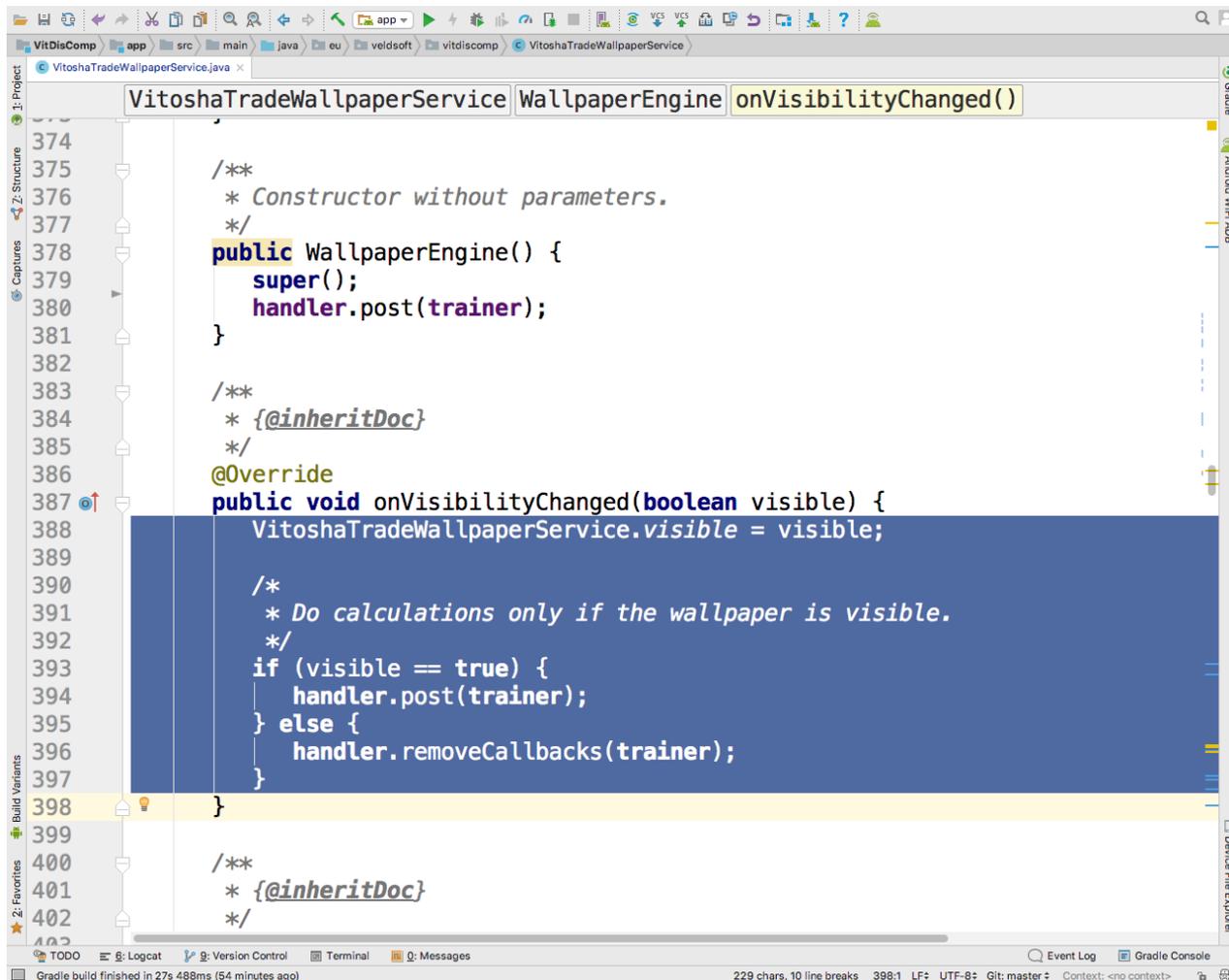
```

365         (int) (Color.red(ANN_COLORS[k]) * topology[k][l]),
366         (int) (Color.green(ANN_COLORS[k]) * topology[k][l]),
367         (int) (Color.blue(ANN_COLORS[k]) * topology[k][l]))
368     canvas.drawPoint(x: x + dx, y: y + dy, paint);
369     }
370 }
371 }
372 }
373 }
374 }
375 /**
376  * Constructor without parameters.
377  */
378 public WallpaperEngine() {
379     super();
380     handler.post(trainer);
381 }
382
383 /**
384  * {@inheritDoc}
385  */
386 @Override
387 public void onVisibilityChanged(boolean visible) {
388     VitoshTradeWallpaperService.visible = visible;
389
390     /**
391      * Do calculations only if the wallpaper is visible.
392      */
393     if (visible == true) {
394

```

Фигура 4.55: Конструктор на двигателя за услугата

Конструкторът на двигателя на услугата има единствената задача да зареди нишката за изпълнение (Фиг. 4.55).



Фигура 4.56: Промяна във видимостта на активния тапет

Когато видимостта на активния тапет бъде променена, се активира събитие `onVisibilityChanged`, в което се определя дали нишката да бъде активирана отново или викането ѝ да бъде преустановено (Фиг. 4.56).

```

381     }
382
383     /**
384     * {@inheritDoc}
385     */
386     @Override
387     public void onVisibilityChanged(boolean visible) {
388         VitoshTradeWallpaperService.visible = visible;
389
390         /*
391          * Do calculations only if the wallpaper is visible.
392          */
393         if (visible == true) {
394             handler.post(trainer);
395         } else {
396             handler.removeCallbacks(trainer);
397         }
398     }
399
400     /**
401     * {@inheritDoc}
402     */
403     @Override
404     public void onSurfaceDestroyed(SurfaceHolder holder) {
405         super.onSurfaceDestroyed(holder);
406         VitoshTradeWallpaperService.visible = false;
407         handler.removeCallbacks(trainer);
408     }
409
410

```

Фигура 4.57: Спиране на пресмятанията при разрушаване на повърхността за изрисване

При събитие за унищожаване на рисуващията площ, се вдига флаг за спиране на изчисленията и нишката отговорна за тях се премахва от опашката за изпълнение (Фиг. 4.57).

```

408     }
409
410     /**
411     * {@inheritDoc}
412     */
413     @Override
414     public void onSurfaceChanged(SurfaceHolder holder,
415                                int format, int width, int height) {
416         super.onSurfaceChanged(holder, format, width, height);
417
418         screenWidth = width;
419         screenHeight = height;
420
421         SharedPreferences preferences = PreferenceManager
422             .getDefaultSharedPreferences(
423                 context: VitoshTradeWallpaperService.this);
424
425         int panelsSideSize = Integer.parseInt(
426             preferences.getString(s: "sizing", s1: "100"));
427
428         switch (preferences.getString(s: "positioning", s1: "0 0")) {
429             case "lt":
430                 panels[0].left = GAP_BETWEEN_PANELS;
431                 panels[0].top = GAP_BETWEEN_PANELS;
432                 panels[0].right = GAP_BETWEEN_PANELS + panelsSideSize;
433                 panels[0].bottom = panelsSideSize + GAP_BETWEEN_PANELS;
434
435                 panels[1].left = GAP_BETWEEN_PANELS;
436                 panels[1].top = 2 * GAP_BETWEEN_PANELS + panelsSideSize;
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```

Фигура 4.58: Размер на зоните за изрисване

Когато настъпи промяна в площта за изрисване на активния тапет, е нужно да се вземат серия мерки по преизчисляване на размерите. Такова събитие примерно настъпва, когато устройството сменя визуалното представяне от портретна в пейзажна или обратното. Първата характеристика която се следи е размерът на петната за визуално представяне (Фиг. 4.58).

```

422     .getDefaultSharedPreferences(
423         context: VitoshTradeWallpaperService.this);
424
425     int panelsSideSize = Integer.parseInt(
426         preferences.getString(s: "sizing", s1: "100"));
427
428     switch (preferences.getString(s: "positioning", s1: "0 0")) {
429         case "lt":
430             panels[0].left = GAP_BETWEEN_PANELS;
431             panels[0].top = GAP_BETWEEN_PANELS;
432             panels[0].right = GAP_BETWEEN_PANELS + panelsSideSize;
433             panels[0].bottom = panelsSideSize + GAP_BETWEEN_PANELS;
434
435             panels[1].left = GAP_BETWEEN_PANELS;
436             panels[1].top = 2 * GAP_BETWEEN_PANELS + panelsSideSize;
437             panels[1].right = GAP_BETWEEN_PANELS + panelsSideSize;
438             panels[1].bottom = 2 * GAP_BETWEEN_PANELS + 2 * panelsSideSize;
439
440             panels[2].left = GAP_BETWEEN_PANELS;
441             panels[2].top = 3 * GAP_BETWEEN_PANELS + 2 * panelsSideSize;
442             panels[2].right = GAP_BETWEEN_PANELS + panelsSideSize;
443             panels[2].bottom = 3 * GAP_BETWEEN_PANELS + 3 * panelsSideSize;
444         break;
445         case "ct":
446             panels[0].left = width / 2 - panelsSideSize / 2;
447             panels[0].top = GAP_BETWEEN_PANELS;
448             panels[0].right = width / 2 + panelsSideSize / 2;
449             panels[0].bottom = panelsSideSize + GAP_BETWEEN_PANELS;
450
451

```

Фигура 4.59: Координати и размери на областите за визуално представяне - горе-ляво

```

437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
    panels[1].right = GAP_BETWEEN_PANELS + panelsSideSize;
    panels[1].bottom = 2 * GAP_BETWEEN_PANELS + 2 * panelsSideSize;

    panels[2].left = GAP_BETWEEN_PANELS;
    panels[2].top = 3 * GAP_BETWEEN_PANELS + 2 * panelsSideSize;
    panels[2].right = GAP_BETWEEN_PANELS + panelsSideSize;
    panels[2].bottom = 3 * GAP_BETWEEN_PANELS + 3 * panelsSideSize;
    break;
    case "ct":
        panels[0].left = width / 2 - panelsSideSize / 2;
        panels[0].top = GAP_BETWEEN_PANELS;
        panels[0].right = width / 2 + panelsSideSize / 2;
        panels[0].bottom = panelsSideSize + GAP_BETWEEN_PANELS;

        panels[1].left = width / 2 - panelsSideSize / 2;
        panels[1].top = 2 * GAP_BETWEEN_PANELS + panelsSideSize;
        panels[1].right = width / 2 + panelsSideSize / 2;
        panels[1].bottom = 2 * GAP_BETWEEN_PANELS + 2 * panelsSideSize;

        panels[2].left = width / 2 - panelsSideSize / 2;
        panels[2].top = 3 * GAP_BETWEEN_PANELS + 2 * panelsSideSize;
        panels[2].right = width / 2 + panelsSideSize / 2;
        panels[2].bottom = 3 * GAP_BETWEEN_PANELS + 3 * panelsSideSize;
    break;
    case "rt":
        panels[0].left = width - panelsSideSize - GAP_BETWEEN_PANELS;
        panels[0].top = GAP_BETWEEN_PANELS;
        panels[0].right = width - GAP_BETWEEN_PANELS;
        panels[0].bottom = panelsSideSize + GAP_BETWEEN_PANELS;
    
```

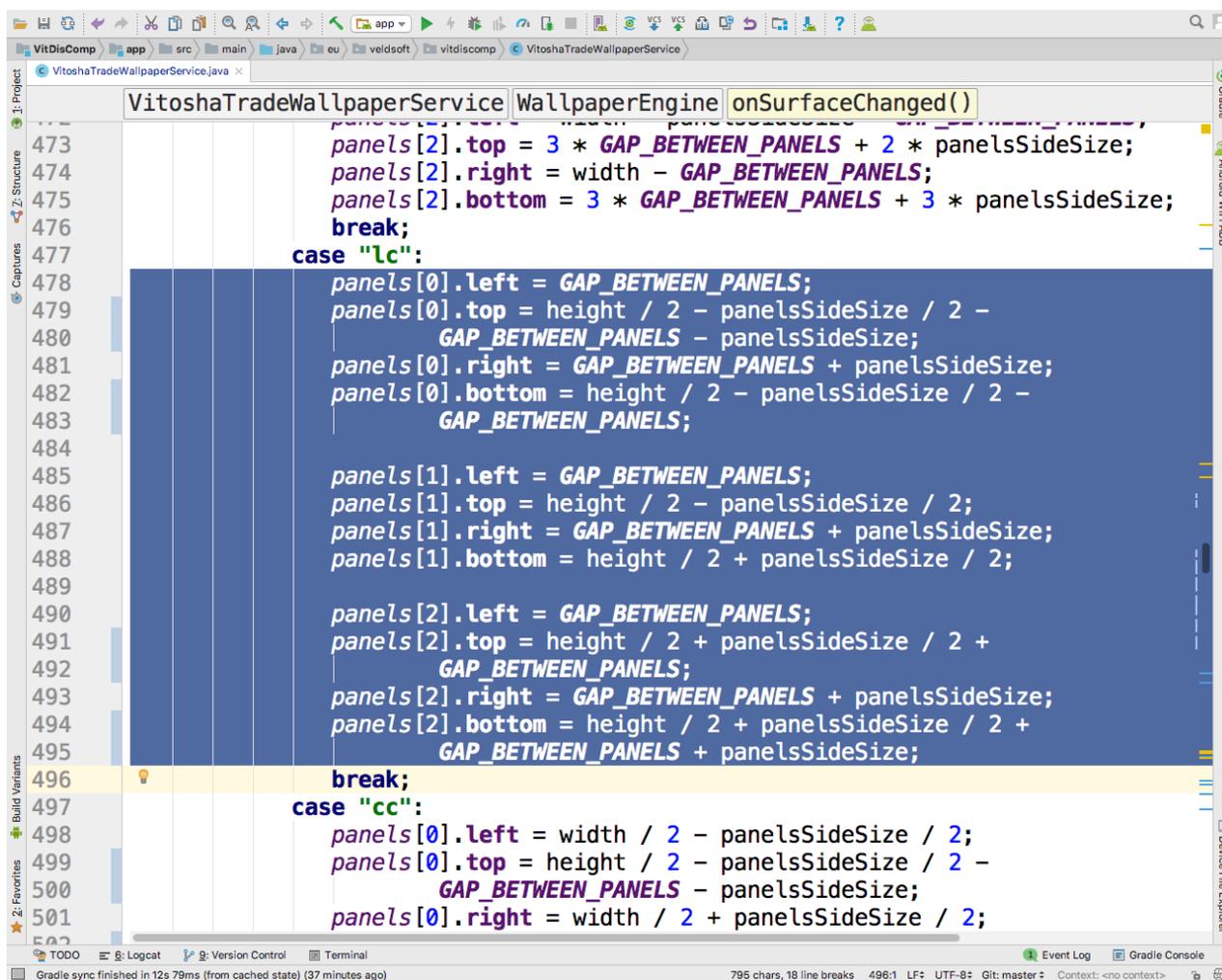
Фигура 4.60: Координати и размери на областите за визуално представяне - горе-център

```

VitoshaTradeWallpaperService WallpaperEngine onSurfaceChanged()
451     panels[1].left = width / 2 - panelsSideSize / 2;
452     panels[1].top = 2 * GAP_BETWEEN_PANELS + panelsSideSize;
453     panels[1].right = width / 2 + panelsSideSize / 2;
454     panels[1].bottom = 2 * GAP_BETWEEN_PANELS + 2 * panelsSideSize;
455
456     panels[2].left = width / 2 - panelsSideSize / 2;
457     panels[2].top = 3 * GAP_BETWEEN_PANELS + 2 * panelsSideSize;
458     panels[2].right = width / 2 + panelsSideSize / 2;
459     panels[2].bottom = 3 * GAP_BETWEEN_PANELS + 3 * panelsSideSize;
460     break;
461     case "rt":
462         panels[0].left = width - panelsSideSize - GAP_BETWEEN_PANELS;
463         panels[0].top = GAP_BETWEEN_PANELS;
464         panels[0].right = width - GAP_BETWEEN_PANELS;
465         panels[0].bottom = panelsSideSize + GAP_BETWEEN_PANELS;
466
467         panels[1].left = width - panelsSideSize - GAP_BETWEEN_PANELS;
468         panels[1].top = 2 * GAP_BETWEEN_PANELS + panelsSideSize;
469         panels[1].right = width - GAP_BETWEEN_PANELS;
470         panels[1].bottom = 2 * GAP_BETWEEN_PANELS + 2 * panelsSideSize;
471
472         panels[2].left = width - panelsSideSize - GAP_BETWEEN_PANELS;
473         panels[2].top = 3 * GAP_BETWEEN_PANELS + 2 * panelsSideSize;
474         panels[2].right = width - GAP_BETWEEN_PANELS;
475         panels[2].bottom = 3 * GAP_BETWEEN_PANELS + 3 * panelsSideSize;
476     break;
477     case "lc":
478         panels[0].left = GAP_BETWEEN_PANELS;
479         panels[0].top = height / 2 - panelsSideSize / 2 -

```

Фигура 4.61: Координати и размери на областите за визуално представяне - горе-дясно



```
VitoshaTradeWallpaperService WallpaperEngine onSurfaceChanged()
panels[2].top = 3 * GAP_BETWEEN_PANELS + 2 * panelsSideSize;
panels[2].right = width - GAP_BETWEEN_PANELS;
panels[2].bottom = 3 * GAP_BETWEEN_PANELS + 3 * panelsSideSize;
break;
case "lc":
panels[0].left = GAP_BETWEEN_PANELS;
panels[0].top = height / 2 - panelsSideSize / 2 -
    GAP_BETWEEN_PANELS - panelsSideSize;
panels[0].right = GAP_BETWEEN_PANELS + panelsSideSize;
panels[0].bottom = height / 2 - panelsSideSize / 2 -
    GAP_BETWEEN_PANELS;

panels[1].left = GAP_BETWEEN_PANELS;
panels[1].top = height / 2 - panelsSideSize / 2;
panels[1].right = GAP_BETWEEN_PANELS + panelsSideSize;
panels[1].bottom = height / 2 + panelsSideSize / 2;

panels[2].left = GAP_BETWEEN_PANELS;
panels[2].top = height / 2 + panelsSideSize / 2 +
    GAP_BETWEEN_PANELS;
panels[2].right = GAP_BETWEEN_PANELS + panelsSideSize;
panels[2].bottom = height / 2 + panelsSideSize / 2 +
    GAP_BETWEEN_PANELS + panelsSideSize;
break;
case "cc":
panels[0].left = width / 2 - panelsSideSize / 2;
panels[0].top = height / 2 - panelsSideSize / 2 -
    GAP_BETWEEN_PANELS - panelsSideSize;
panels[0].right = width / 2 + panelsSideSize / 2;
```

Фигура 4.62: Координати и размери на областите за визуално представяне - център-ляво

Фигура 4.63: Координати и размери на областите за визуално представяне - център-център

```

512         GAP_BETWEEN_PANELS;
513         panels[2].right = width / 2 + panelsSideSize / 2;
514         panels[2].bottom = height / 2 + panelsSideSize / 2 +
515             GAP_BETWEEN_PANELS + panelsSideSize;
516         break;
517         case "rc":
518             panels[0].left = width - panelsSideSize - GAP_BETWEEN_PANELS;
519             panels[0].top = height / 2 - panelsSideSize / 2 -
520                 GAP_BETWEEN_PANELS - panelsSideSize;
521             panels[0].right = width - GAP_BETWEEN_PANELS;
522             panels[0].bottom = height / 2 - panelsSideSize / 2 -
523                 GAP_BETWEEN_PANELS;
524
525             panels[1].left = width - panelsSideSize - GAP_BETWEEN_PANELS;
526             panels[1].top = height / 2 - panelsSideSize / 2;
527             panels[1].right = width - GAP_BETWEEN_PANELS;
528             panels[1].bottom = height / 2 + panelsSideSize / 2;
529
530             panels[2].left = width - panelsSideSize - GAP_BETWEEN_PANELS;
531             panels[2].top = height / 2 + panelsSideSize / 2 +
532                 GAP_BETWEEN_PANELS;
533             panels[2].right = width - GAP_BETWEEN_PANELS;
534             panels[2].bottom = height / 2 + panelsSideSize / 2 +
535                 GAP_BETWEEN_PANELS + panelsSideSize;
536         break;
537         case "lb":
538             panels[0].left = GAP_BETWEEN_PANELS;
539             panels[0].top = height - 3 * GAP_BETWEEN_PANELS - 3 *
540                 panelsSideSize;
541

```

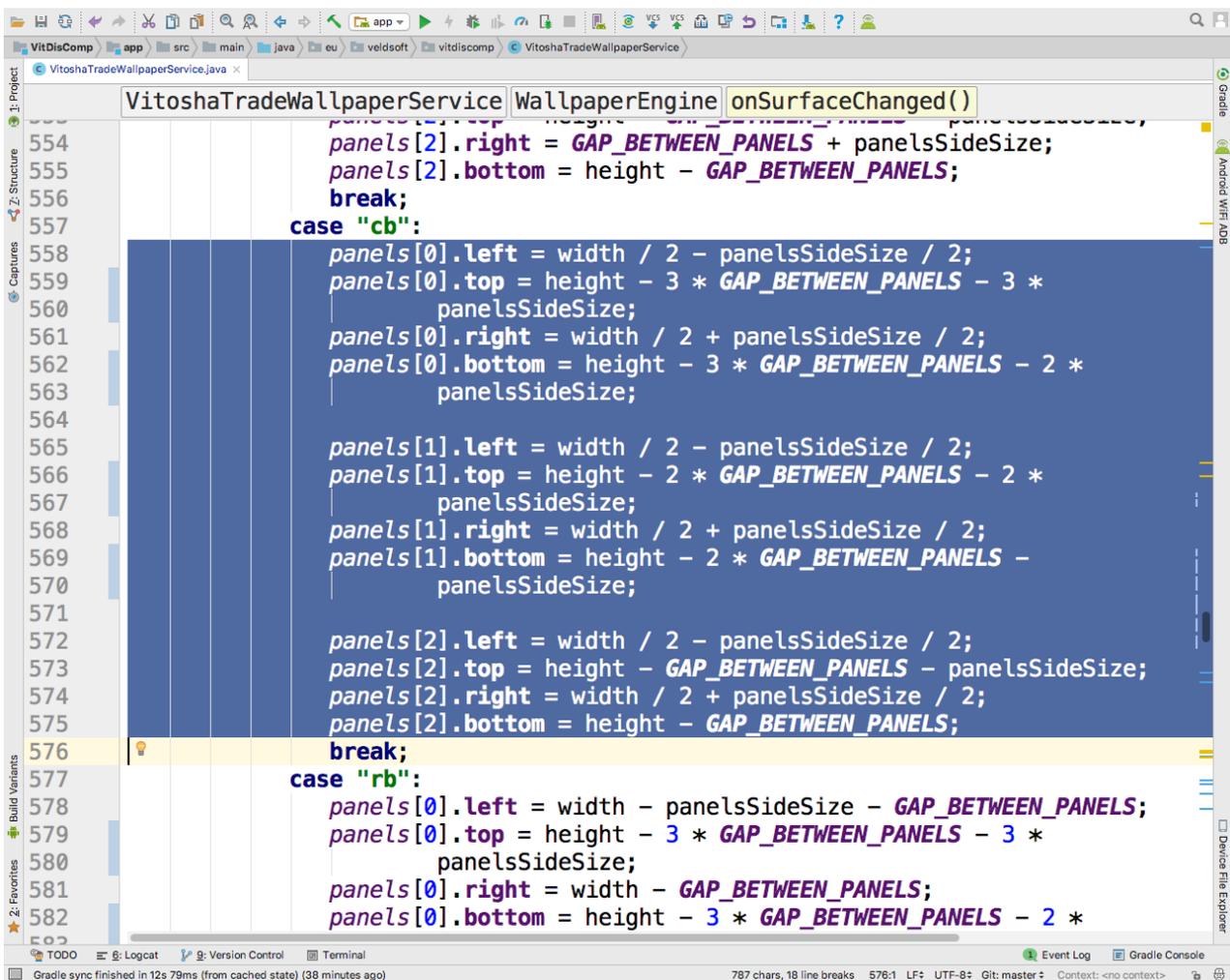
Фигура 4.64: Координати и размери на областите за визуално представяне - център-дясно

```

VitoshaTradeWallpaperService WallpaperEngine onSurfaceChanged()
533     panels[2].right = width - GAP_BETWEEN_PANELS;
534     panels[2].bottom = height / 2 + panelsSideSize / 2 +
535         GAP_BETWEEN_PANELS + panelsSideSize;
536     break;
537     case "lb":
538         panels[0].left = GAP_BETWEEN_PANELS;
539         panels[0].top = height - 3 * GAP_BETWEEN_PANELS - 3 *
540             panelsSideSize;
541         panels[0].right = GAP_BETWEEN_PANELS + panelsSideSize;
542         panels[0].bottom = height - 3 * GAP_BETWEEN_PANELS - 2 *
543             panelsSideSize;
544
545         panels[1].left = GAP_BETWEEN_PANELS;
546         panels[1].top = height - 2 * GAP_BETWEEN_PANELS - 2 *
547             panelsSideSize;
548         panels[1].right = GAP_BETWEEN_PANELS + panelsSideSize;
549         panels[1].bottom = height - 2 * GAP_BETWEEN_PANELS -
550             panelsSideSize;
551
552         panels[2].left = GAP_BETWEEN_PANELS;
553         panels[2].top = height - GAP_BETWEEN_PANELS - panelsSideSize;
554         panels[2].right = GAP_BETWEEN_PANELS + panelsSideSize;
555         panels[2].bottom = height - GAP_BETWEEN_PANELS;
556     break;
557     case "cb":
558         panels[0].left = width / 2 - panelsSideSize / 2;
559         panels[0].top = height - 3 * GAP_BETWEEN_PANELS - 3 *
560             panelsSideSize;
561         panels[0].right = width / 2 + panelsSideSize / 2;
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```

Фигура 4.65: Координати и размери на областите за визуално представяне - долу-ляво



```
VitoshTradeWallpaperService WallpaperEngine onSurfaceChanged()  
554 panels[2].right = GAP_BETWEEN_PANELS + panelsSideSize;  
555 panels[2].bottom = height - GAP_BETWEEN_PANELS;  
556 break;  
557 case "cb":  
558 panels[0].left = width / 2 - panelsSideSize / 2;  
559 panels[0].top = height - 3 * GAP_BETWEEN_PANELS - 3 *  
560 panelsSideSize;  
561 panels[0].right = width / 2 + panelsSideSize / 2;  
562 panels[0].bottom = height - 3 * GAP_BETWEEN_PANELS - 2 *  
563 panelsSideSize;  
564  
565 panels[1].left = width / 2 - panelsSideSize / 2;  
566 panels[1].top = height - 2 * GAP_BETWEEN_PANELS - 2 *  
567 panelsSideSize;  
568 panels[1].right = width / 2 + panelsSideSize / 2;  
569 panels[1].bottom = height - 2 * GAP_BETWEEN_PANELS -  
570 panelsSideSize;  
571  
572 panels[2].left = width / 2 - panelsSideSize / 2;  
573 panels[2].top = height - GAP_BETWEEN_PANELS - panelsSideSize;  
574 panels[2].right = width / 2 + panelsSideSize / 2;  
575 panels[2].bottom = height - GAP_BETWEEN_PANELS;  
576 break;  
577 case "rb":  
578 panels[0].left = width - panelsSideSize - GAP_BETWEEN_PANELS;  
579 panels[0].top = height - 3 * GAP_BETWEEN_PANELS - 3 *  
580 panelsSideSize;  
581 panels[0].right = width - GAP_BETWEEN_PANELS;  
582 panels[0].bottom = height - 3 * GAP_BETWEEN_PANELS - 2 *
```

Фигура 4.66: Координати и размери на областите за визуално представяне - долу-център

```

570         panelsSideSize;
571
572         panels[2].left = width / 2 - panelsSideSize / 2;
573         panels[2].top = height - GAP_BETWEEN_PANELS - panelsSideSize;
574         panels[2].right = width / 2 + panelsSideSize / 2;
575         panels[2].bottom = height - GAP_BETWEEN_PANELS;
576         break;
577     case "rb":
578         panels[0].left = width - panelsSideSize - GAP_BETWEEN_PANELS;
579         panels[0].top = height - 3 * GAP_BETWEEN_PANELS - 3 *
580             panelsSideSize;
581         panels[0].right = width - GAP_BETWEEN_PANELS;
582         panels[0].bottom = height - 3 * GAP_BETWEEN_PANELS - 2 *
583             panelsSideSize;
584
585         panels[1].left = width - panelsSideSize - GAP_BETWEEN_PANELS;
586         panels[1].top = height - 2 * GAP_BETWEEN_PANELS - 2 *
587             panelsSideSize;
588         panels[1].right = width - GAP_BETWEEN_PANELS;
589         panels[1].bottom = height - 2 * GAP_BETWEEN_PANELS -
590             panelsSideSize;
591
592         panels[2].left = width - panelsSideSize - GAP_BETWEEN_PANELS;
593         panels[2].top = height - GAP_BETWEEN_PANELS - panelsSideSize;
594         panels[2].right = width - GAP_BETWEEN_PANELS;
595         panels[2].bottom = height - GAP_BETWEEN_PANELS;
596         break;
597     default:
598         break;
599

```

Фигура 4.67: Координати и размери на областите за визуално представяне - долу-дясно

На потребителя е позволено да избере в коя част на активния тапет да бъдат позиционирани зоните за визуално представяне. Възможностите по хоризонтала са ляво, център и дясно, а възможностите по вертикала са горе, център и долу.

```

580         panelsSideSize;
581         panels[0].right = width - GAP_BETWEEN_PANELS;
582         panels[0].bottom = height - 3 * GAP_BETWEEN_PANELS - 2 *
583             panelsSideSize;
584
585         panels[1].left = width - panelsSideSize - GAP_BETWEEN_PANELS;
586         panels[1].top = height - 2 * GAP_BETWEEN_PANELS - 2 *
587             panelsSideSize;
588         panels[1].right = width - GAP_BETWEEN_PANELS;
589         panels[1].bottom = height - 2 * GAP_BETWEEN_PANELS -
590             panelsSideSize;
591
592         panels[2].left = width - panelsSideSize - GAP_BETWEEN_PANELS;
593         panels[2].top = height - GAP_BETWEEN_PANELS - panelsSideSize;
594         panels[2].right = width - GAP_BETWEEN_PANELS;
595         panels[2].bottom = height - GAP_BETWEEN_PANELS;
596         break;
597     default:
598         break;
599 }
600
601     delay = Long.parseLong(preferences.getString(s: "loading",
602         s1: "" + DEFAULT_DELAY));
603 }
604 }
605
606 /**
607  * Pseudo-random number generator.
608  */
609

```

Фигура 4.68: Натоварване на системата за фонови пресмятания

Последната характеристика, която потребителят има възможност да контролира, е до каква степен мобилното му устройство да бъде натоварвано с фонови пресмятания (Фиг. 4.68).

## 4.5 Представяне на информацията върху локалното устройство

Ефективността от изчисленията значително се повишава, ако локалните изчислителни възли разполагат с възможност за локално съхранение на изходни данни и пресметнати резултати.

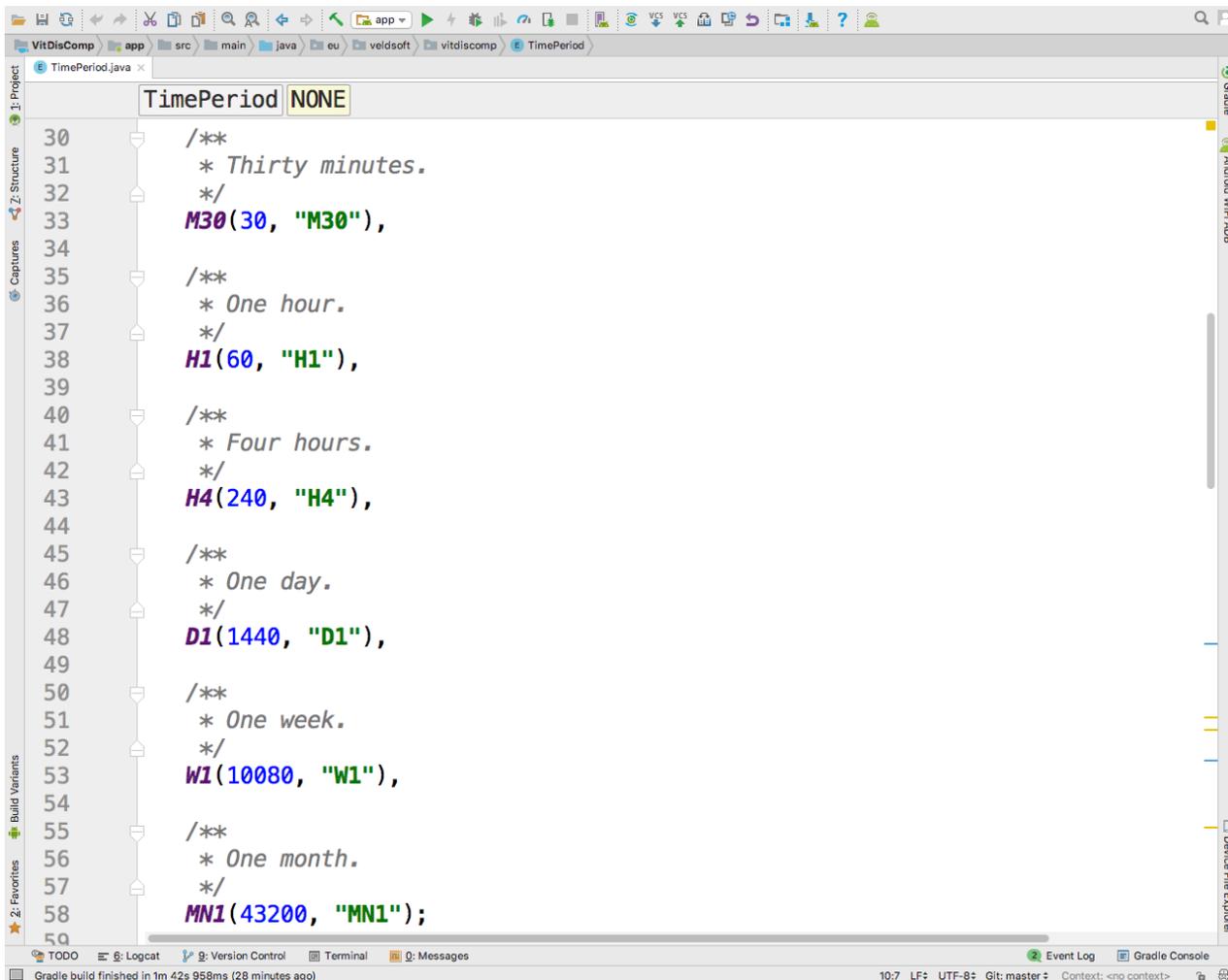
```

1  TimePeriod.java
2
3  /**
4   * Time series fixed time periods.
5   *
6   * @author Todor Balabanov
7   */
8  enum TimePeriod {
9
10     /**
11      * No time period at all.
12      */
13     NONE(0, ""),
14
15     /**
16      * One minute.
17      */
18     M1(1, "M1"),
19
20     /**
21      * Five minutes.
22      */
23     M5(5, "M5"),
24
25     /**
26      * Fifteen minutes.
27      */
28     M15(15, "M15"),
29
30     /**
31      * Thirty minutes

```

Фигура 4.69: Константи за интервалите на времевия ред (0 до 15)

За нуждите на това локално представяне се ползва помощен тип данни от избран тип, който обозначава разстоянието между две отчитания във финансовите времеви редове (Фиг. 4.69, 4.70).



Фигура 4.70: Константи за интервалите на времевия ред (30 до 43200)

Интервалите в използваните времеви редове се определят на база брой минути като най-краткият интервал е една минута. Описването на интервалите става с две променливи - название и брой минути (Фиг. 4.71).

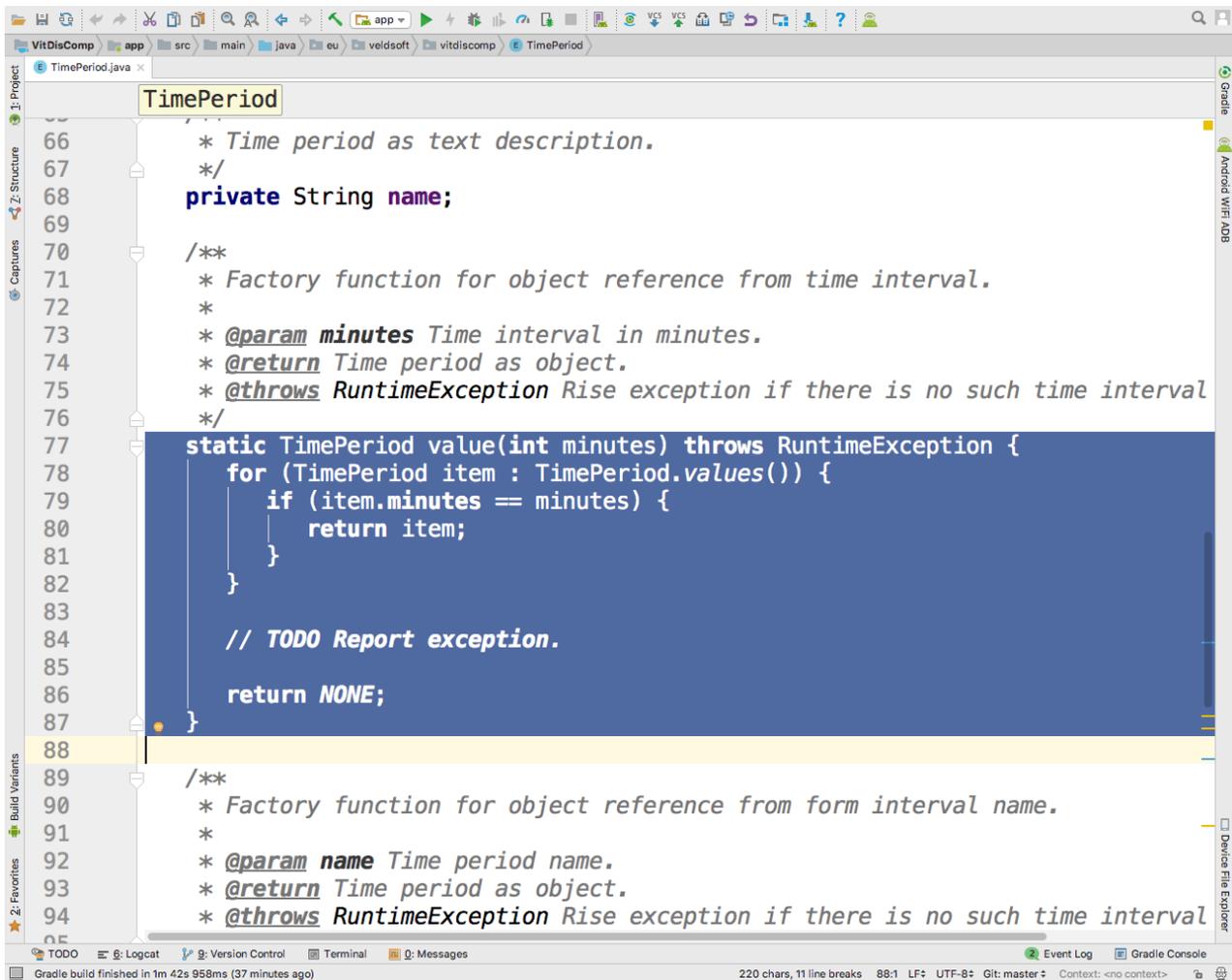
```

56     * One month.
57     */
58     MN1(43200, "MN1");
59
60     /**
61     * Time period as number of minutes.
62     */
63     private int minutes;
64
65     /**
66     * Time period as text description.
67     */
68     private String name;
69
70     /**
71     * Factory function for object reference from time interval.
72     *
73     * @param minutes Time interval in minutes.
74     * @return Time period as object.
75     * @throws RuntimeException Rise exception if there is no such time interval
76     */
77     static TimePeriod value(int minutes) throws RuntimeException {
78         for (TimePeriod item : TimePeriod.values()) {
79             if (item.minutes == minutes) {
80                 return item;
81             }
82         }
83
84         // TODO Report exception.
85

```

Фигура 4.71: Описание на времеви интервал

Използването на изброени константи в Java дава възможност за елегантно използване на статични функции за конструиране на обекти (Factory Method Design Pattern), които по зададено число (брой минути) да връщат съответстващата константа (Фиг. 4.72).



```
TimePeriod
66     * Time period as text description.
67     */
68     private String name;
69
70     /**
71     * Factory function for object reference from time interval.
72     *
73     * @param minutes Time interval in minutes.
74     * @return Time period as object.
75     * @throws RuntimeException Rise exception if there is no such time interval
76     */
77     static TimePeriod value(int minutes) throws RuntimeException {
78         for (TimePeriod item : TimePeriod.values()) {
79             if (item.minutes == minutes) {
80                 return item;
81             }
82         }
83
84         // TODO Report exception.
85
86         return NONE;
87     }
88
89     /**
90     * Factory function for object reference from form interval name.
91     *
92     * @param name Time period name.
93     * @return Time period as object.
94     * @throws RuntimeException Rise exception if there is no such time interval
95     */
```

Фигура 4.72: Определяне на константа за интервал по брой минути

Аналогичен ефект може да се постигне и с използване на текстовото описание на константите (Фиг. 4.73).

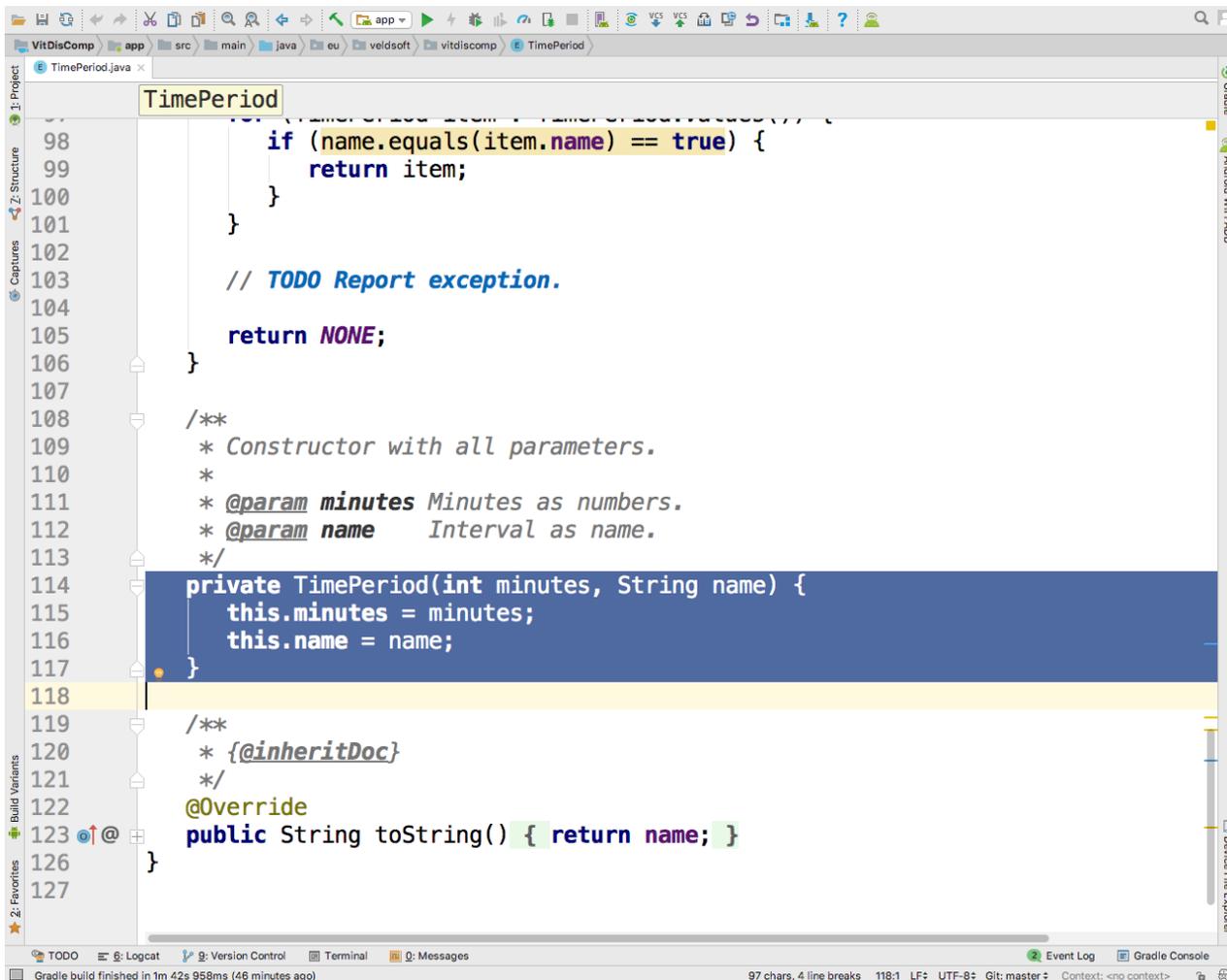
```

86     return NONE;
87 }
88
89 /**
90  * Factory function for object reference from form interval name.
91  *
92  * @param name Time period name.
93  * @return Time period as object.
94  * @throws RuntimeException Rise exception if there is no such time interval
95  */
96 static TimePeriod value(String name) throws RuntimeException {
97     for (TimePeriod item : TimePeriod.values()) {
98         if (name.equals(item.name) == true) {
99             return item;
100        }
101    }
102
103    // TODO Report exception.
104
105    return NONE;
106 }
107
108 /**
109  * Constructor with all parameters.
110  *
111  * @param minutes Minutes as numbers.
112  * @param name Interval as name.
113  */
114 private TimePeriod(int minutes, String name) {
115     this.minutes = minutes;

```

Фигура 4.73: Определяне на константа за интервал по текстово описание

За да се възпрепятства създаването на константи извън изброяения тип, е прието конструкторът да бъде с частно ниво на достъп (Фиг. 4.74).



Фигура 4.74: Конструктор на изброения тип за интервал

Операционната система Android предоставя изключително полезни средства за съхраняване на информация, едно от които е релационната система за управление на бази от данни SQLite.

```

1 package eu.veldsoft.vitdiscomp;
2
3 import ...
4
5
6
7
8 /**
9  * Database helper class.
10 *
11 * @author Todor Balabanov
12 */
13 class ForecastDatabaseHelper extends SQLiteOpenHelper {
14     /**
15     * Rates table columns description class.
16     *
17     * @author Todor Balabanov
18     */
19     public static abstract class RatesColumns implements BaseColumns {
20         public static final String TABLE_NAME = "rates";
21         public static final String COLUMN_NAME_SYMBOL = "symbol";
22         public static final String COLUMN_NAME_PERIOD = "period";
23         public static final String COLUMN_NAME_TIME = "time";
24         public static final String COLUMN_NAME_OPEN = "open";
25         public static final String COLUMN_NAME_LOW = "low";
26         public static final String COLUMN_NAME_HIGH = "high";
27         public static final String COLUMN_NAME_CLOSE = "close";
28         public static final String COLUMN_NAME_VOLUME = "volume";
29     }
30
31     /**
32     * ANN table columns description class.
33

```

Фигура 4.75: Структура на локалната таблица за котировките

Информацията за времевия ред може да се съхранява локално в таблица с подходяща структура, която да отразява наличието на четири стойности за всеки интервал, времето на интервала, търгувания обем, названието на валутната двойка и размера на времевия интервал (Фиг. 4.75).

```

24 public static final String COLUMN_NAME_OPEN = "open";
25 public static final String COLUMN_NAME_LOW = "low";
26 public static final String COLUMN_NAME_HIGH = "high";
27 public static final String COLUMN_NAME_CLOSE = "close";
28 public static final String COLUMN_NAME_VOLUME = "volume";
29 }
30
31 /**
32  * ANN table columns description class.
33  *
34  * @author Todor Balabanov
35  */
36 public static abstract class ANNsColumns implements BaseColumns {
37     public static final String TABLE_NAME = "anns";
38     public static final String COLUMN_NAME_SYMBOL = "symbol";
39     public static final String COLUMN_NAME_PERIOD = "period";
40     public static final String COLUMN_NAME_NEURONS = "";
41     public static final String COLUMN_NAME_ACTIVITIES = "";
42     public static final String COLUMN_NAME_WEIGHTS = "";
43 }
44
45 /**
46  * Database integer version.
47  */
48 public static final int DATABASE_VERSION = 1;
49
50 /**
51  * Database file name.
52  */
53 public static final String DATABASE_NAME = "Forecast.db";

```

Фигура 4.76: Структура на локалната таблица за изкуствените невронни мрежи

Допълнителна таблица поема съхранението на информацията за изкуствените невронни мрежи, което включва названието на валутната двойка, времевия интервал, стойността на невроните, стойността на връзките между невроните и стойността на теглата между невроните (Фиг. 4.76).

```

ForecastDatabaseHelper
40 public static final String COLUMN_NAME_NEURONS = "";
41 public static final String COLUMN_NAME_ACTIVITIES = "";
42 public static final String COLUMN_NAME_WEIGHTS = "";
43 }
44
45 /**
46  * Database integer version.
47  */
48 public static final int DATABASE_VERSION = 1;
49
50 /**
51  * Database file name.
52  */
53 public static final String DATABASE_NAME = "Forecast.db";
54
55 /**
56  * Create rates table SQL patter.
57  */
58 private static final String SQL_CREATE_RATES = "CREATE TABLE " + RatesColumns.
59     + " INTEGER NOT NULL," + RatesColumns.COLUMN_NAME_SYMBOL + " TEXT, " +
60     + " INTEGER, " + RatesColumns.COLUMN_NAME_TIME + " TEXT, " + RatesColu
61     + RatesColumns.COLUMN_NAME_LOW + " TEXT, " + RatesColumns.COLUMN_NAME_
62     + RatesColumns.COLUMN_NAME_CLOSE + " TEXT, " + RatesColumns.COLUMN_NAM
63     + RatesColumns.COLUMN_NAME_SYMBOL + ", " + RatesColumns.COLUMN_NAME_PE
64
65 /**
66  * Create ANNs table SQL patter.
67  */
68 private static final String SQL_CREATE_ANNS = "CREATE TABLE " + ANNsColumns.TA
69

```

Фигура 4.77: Име и версията на базата данни

Android управлява базите от данни под формата на DB ресурс и целочислена версия за структурата (Фиг. 4.77).

```

53 public static final String DATABASE_NAME = "Forecast.db";
54
55 /**
56  * Create rates table SQL patter.
57  */
58 private static final String SQL_CREATE_RATES = "CREATE TABLE "
59     + RatesColumns.TABLE_NAME + " (" + RatesColumns._ID
60     + " INTEGER NOT NULL," + RatesColumns.COLUMN_NAME_SYMBOL
61     + " TEXT, " + RatesColumns.COLUMN_NAME_PERIOD
62     + " INTEGER, " + RatesColumns.COLUMN_NAME_TIME + " TEXT, "
63     + RatesColumns.COLUMN_NAME_OPEN + " TEXT, "
64     + RatesColumns.COLUMN_NAME_LOW + " TEXT, "
65     + RatesColumns.COLUMN_NAME_HIGH + " TEXT, "
66     + RatesColumns.COLUMN_NAME_CLOSE + " TEXT, "
67     + RatesColumns.COLUMN_NAME_VOLUME + " TEXT PRIMARY KEY ("
68     + RatesColumns.COLUMN_NAME_SYMBOL + ", "
69     + RatesColumns.COLUMN_NAME_PERIOD + ")");
70
71 /**
72  * Create ANNs table SQL patter.
73  */
74 private static final String SQL_CREATE_ANNS = "CREATE TABLE "
75     + ANNsColumns.TABLE_NAME + " (" + ANNsColumns._ID
76     + " INTEGER PRIMARY KEY," + ANNsColumns.COLUMN_NAME_SYMBOL
77     + " TEXT, " + ANNsColumns.COLUMN_NAME_PERIOD
78     + " INTEGER, " + ANNsColumns.COLUMN_NAME_NEURONS + " TEXT, "
79     + ANNsColumns.COLUMN_NAME_ACTIVITIES
80     + " TEXT, " + ANNsColumns.COLUMN_NAME_WEIGHTS
81     + " TEXT, FOREIGN KEY (" + ANNsColumns.COLUMN_NAME_SYMBOL

```

Фигура 4.78: Код за създаване на таблицата за котировки

```

63         + RatesColumns.COLUMN_NAME_OPEN + " TEXT, "
64         + RatesColumns.COLUMN_NAME_LOW + " TEXT, "
65         + RatesColumns.COLUMN_NAME_HIGH + " TEXT, "
66         + RatesColumns.COLUMN_NAME_CLOSE + " TEXT, "
67         + RatesColumns.COLUMN_NAME_VOLUME + " TEXT PRIMARY KEY ("
68         + RatesColumns.COLUMN_NAME_SYMBOL + ", "
69         + RatesColumns.COLUMN_NAME_PERIOD + ")");
70
71     /**
72     * Create ANNs table SQL patter.
73     */
74     private static final String SQL_CREATE_ANNS = "CREATE TABLE "
75         + ANNsColumns.TABLE_NAME + " (" + ANNsColumns._ID
76         + " INTEGER PRIMARY KEY," + ANNsColumns.COLUMN_NAME_SYMBOL
77         + " TEXT, " + ANNsColumns.COLUMN_NAME_PERIOD
78         + " INTEGER, " + ANNsColumns.COLUMN_NAME_NEURONS + " TEXT, "
79         + ANNsColumns.COLUMN_NAME_ACTIVITIES
80         + " TEXT, " + ANNsColumns.COLUMN_NAME_WEIGHTS
81         + " TEXT, FOREIGN KEY (" + ANNsColumns.COLUMN_NAME_SYMBOL
82         + ") REFERENCES " + RatesColumns.TABLE_NAME
83         + "(" + RatesColumns.COLUMN_NAME_SYMBOL + "), FOREIGN KEY ("
84         + ANNsColumns.COLUMN_NAME_PERIOD + ") REFERENCES "
85         + RatesColumns.TABLE_NAME + "("
86         + RatesColumns.COLUMN_NAME_PERIOD + ")");
87
88     /**
89     * Drop rates table SQL pattern.
90     */
91     static final String SQL_DELETE_RATES = "DROP TABLE IF EXISTS " + RatesColumns.

```

Фигура 4.79: Код за създаване на таблицата за изкуствените невронни мрежи

Създаването на таблиците се управлява от параметризирани символни константи (Фиг. 4.78, 4.79).

```

82         + ") REFERENCES " + RatesColumns.TABLE_NAME
83         + "(" + RatesColumns.COLUMN_NAME_SYMBOL + "), FOREIGN KEY ("
84         + ANNsColumns.COLUMN_NAME_PERIOD + ") REFERENCES "
85         + RatesColumns.TABLE_NAME + "("
86         + RatesColumns.COLUMN_NAME_PERIOD + "));";
87
88     /**
89     * Drop rates table SQL pattern.
90     */
91     static final String SQL_DELETE_RATES = "DROP TABLE IF EXISTS "
92         + RatesColumns.TABLE_NAME;
93
94     /**
95     * Drop rates table SQL pattern.
96     */
97     static final String SQL_DELETE_ANNS = "DROP TABLE IF EXISTS "
98         + ANNsColumns.TABLE_NAME;
99
100    /**
101    * Constructor.
102    *
103    * @param context Context of database helper usage.
104    */
105    public ForecastDatabaseHelper(Context context) {
106        super(context, DATABASE_NAME, factory: null, DATABASE_VERSION);
107    }
108
109    /**
110    * {@inheritDoc}

```

Фигура 4.80: Код за изтриване на таблиците

Изтриването на таблиците също става с параметризирани символни константи (Фиг. 4.80).

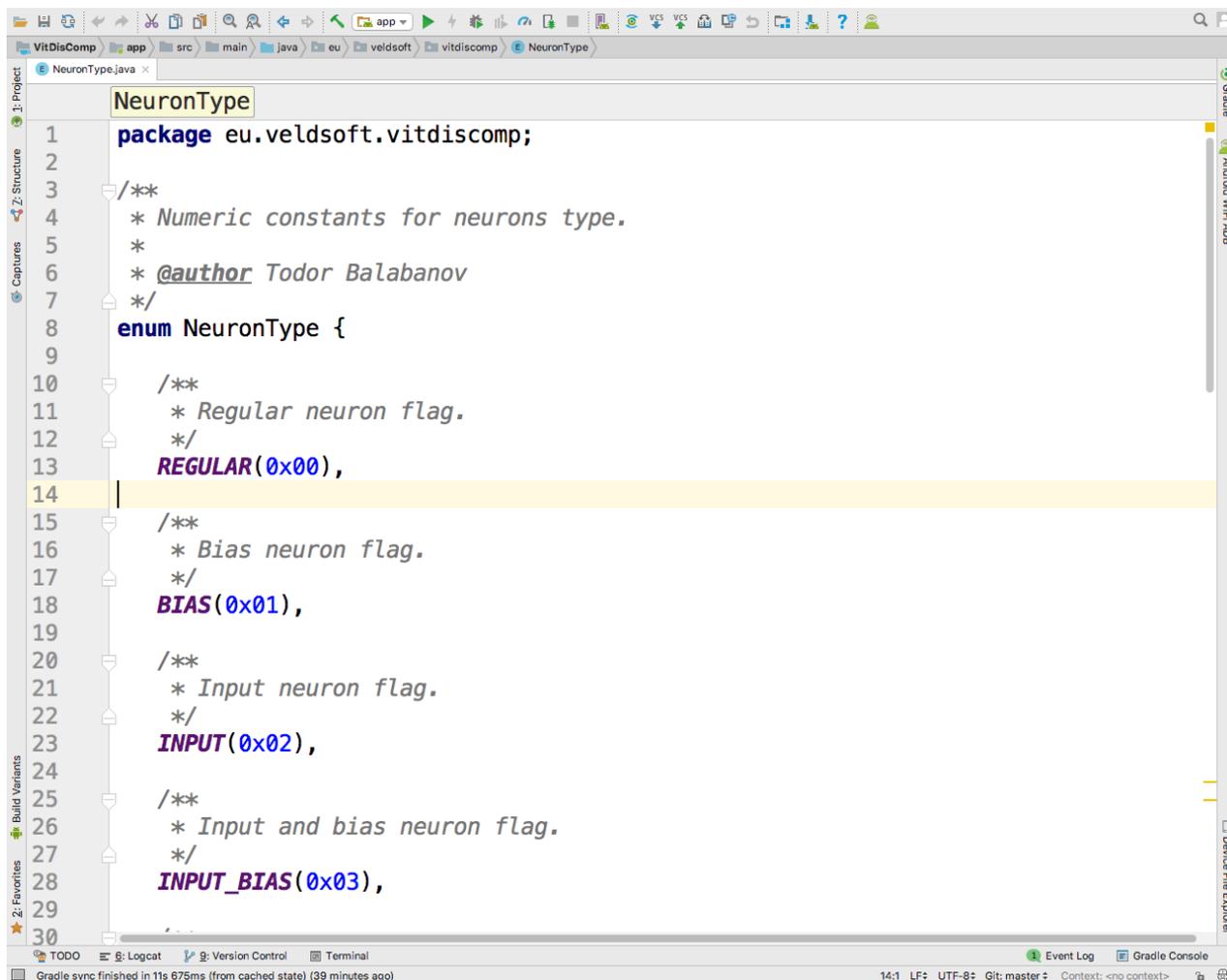
```

99
100 /**
101  * Constructor.
102  *
103  * @param context Context of database helper usage.
104  */
105 public ForecastDatabaseHelper(Context context) {
106     super(context, DATABASE_NAME, factory: null, DATABASE_VERSION);
107 }
108
109 /**
110  * {@inheritDoc}
111  */
112 @Override
113 public void onCreate(SQLiteDatabase db) {
114     db.execSQL(SQL_CREATE_RATES);
115     db.execSQL(SQL_CREATE_ANNS);
116 }
117
118 /**
119  * {@inheritDoc}
120  */
121 @Override
122 public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
123     db.execSQL(SQL_DELETE_ANNS);
124     db.execSQL(SQL_DELETE_RATES);
125     onCreate(db);
126 }
127 }
128

```

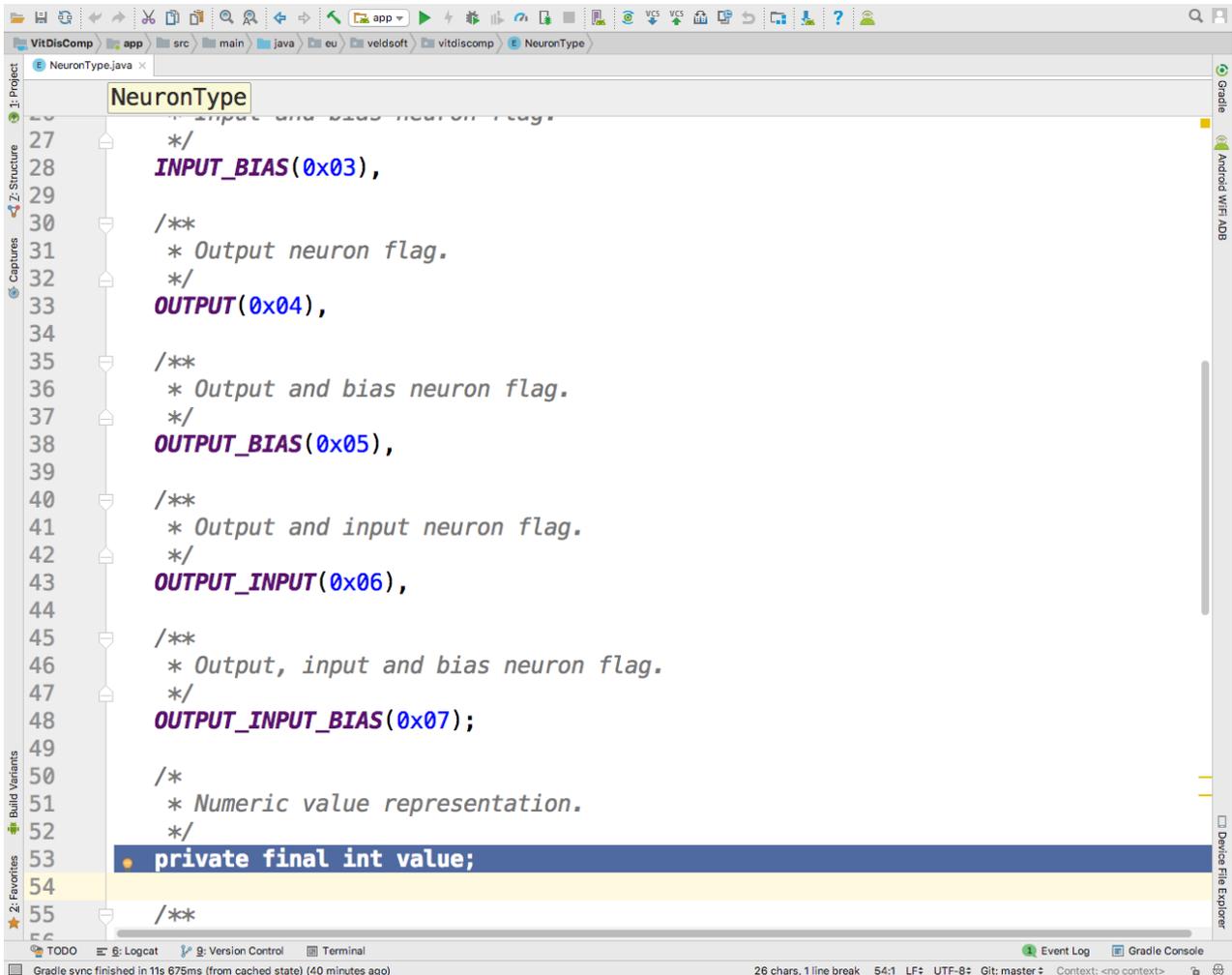
Фигура 4.81: Конструктор и виртуални методи на спомагателния клас за базата данни

Конструкторът на спомагателния клас за базата данни има единствена задача да извика конструкторът на родителския клас (Фиг. 4.81). Двете предефинирани виртуални функции имат за задача да извикат заявките за конструиране и обновяване на базата данни (Фиг. 4.81).



Фигура 4.82: Изброен тип за вида на невроните

Вторият изброен тип данни е по отношение на типа неврони. В общия случай невроните могат да бъдат обикновени, входни, изходни и отместващи (bias) (Фиг. 4.81).



Фигура 4.83: Целочислена константа за типа на неврона, която може да се ползва и като битово поле

Тъй като са възможни различни връзки между невроните, то се появяват и неврони с повече функции като: обикновен-входен, обикновен-изходен, входен-изходен, входен-обикновен-изходен. Всички възможни комбинации се записват в целочислена променлива, която може да послужи и като битово поле (Фиг. 4.83).

```

48  NeuronType valueOf()
49
50  /*
51   * Numeric value representation.
52   */
53  private final int value;
54
55  /**
56   * Value factory function.
57   *
58   * @param type Numerical type representation.
59   * @return Corresponding enumeration or regular if there is no correspondence.
60   */
61  public static NeuronType valueOf(int type) {
62      for (NeuronType item : NeuronType.values()) {
63          if (item.value() == type) {
64              return item;
65          }
66      }
67
68      return REGULAR;
69  }
70
71  /**
72   * Constructor with all parameters.
73   *
74   * @param value
75   */
76  private NeuronType(int value) { this.value = value; }
77
78

```

Фигура 4.84: Определяне на изброимата константа по числена стойност

Тъй като информацията от сървъра пристига предимно под формата на числа, то е рационално да бъде добавена статична функция за конструиране на обекта (Factory Method Design Pattern), която по числена стойност да определя константата в изброения тип (Фиг. 4.84).

```

66     }
67
68     return REGULAR;
69 }
70
71 /**
72  * Constructor with all parameters.
73  *
74  * @param value
75  */
76 private NeuronType(int value) {
77     this.value = value;
78 }
79
80 /**
81  * Value getter.
82  *
83  * @return Numeric representation of the type.
84  */
85 @ public int value() {
86     return value;
87 }
88 }
89

```

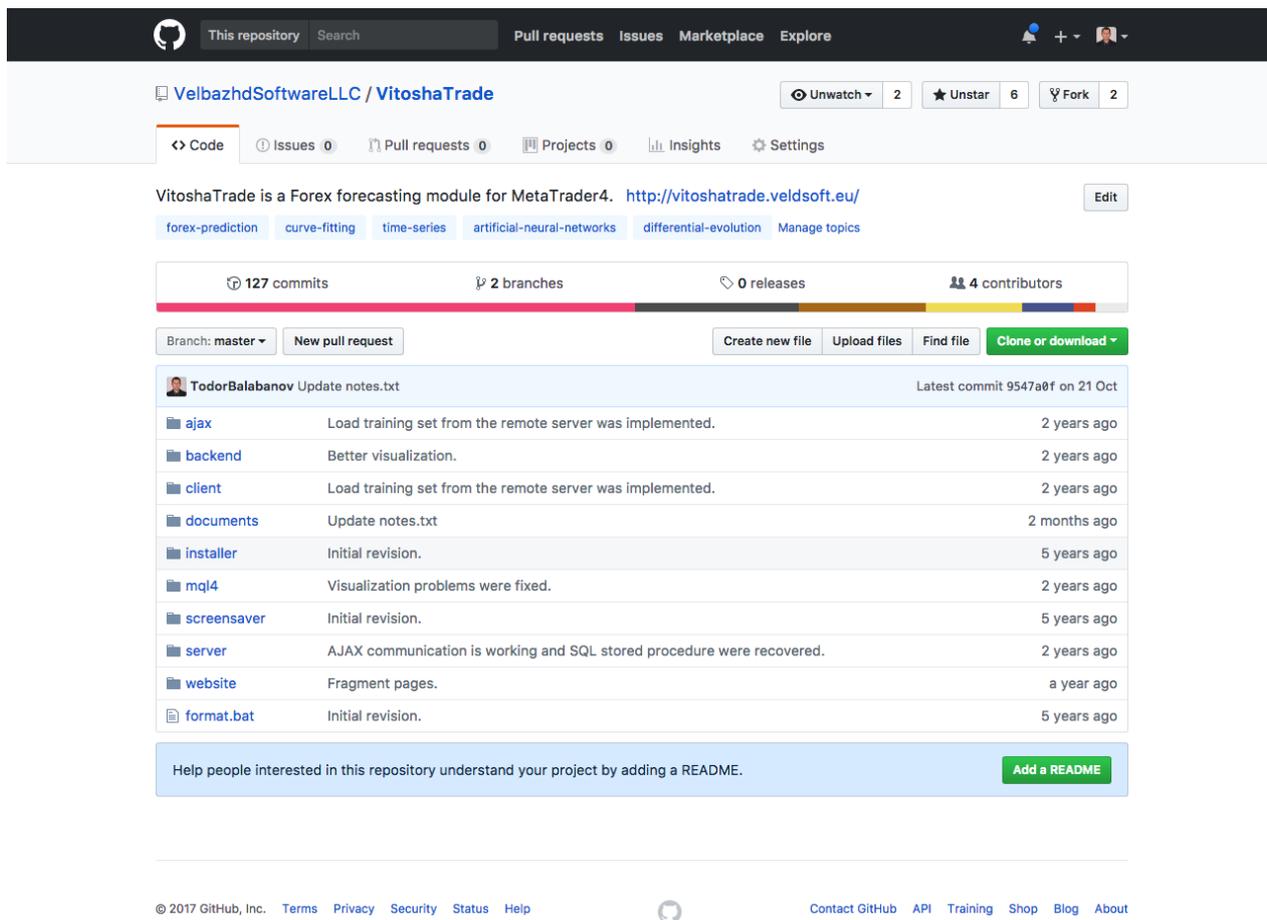
Фигура 4.85: Методи за достъп до вътрешните стойности на константите

За удобство, при работата с константите и със скритата в тях информация, често се използват методи за достъп (accessor). В този случай един публичен getter, но setter с частен достъп. Разумно е setter-ът да бъде с частен достъп, тъй като е разумно да се запази свойството на изброените константи да бъдат константи (Фиг. 4.85).

## Глава 5

# Централизиран сървър

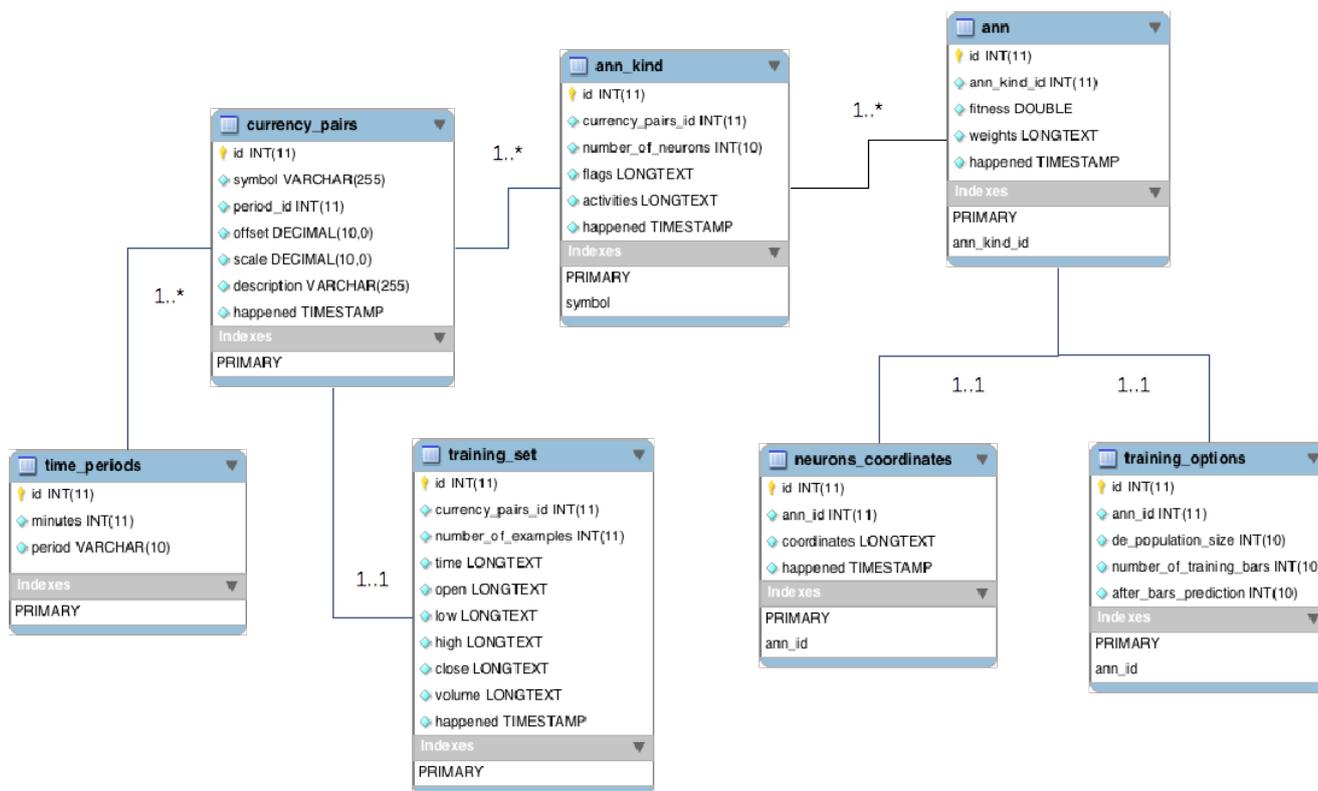
За да бъдат успешно извършени изчисленията в разпределена среда, освен наличието на множество устройства е от съществено значение да има централизирано място, от което да бъдат раздавани задачите и където да бъдат събирани резултатите. Архитектура от тип клиент-сървър би била изключително удачна при едно техническо решение за не особено интензивна комуникация, която не изисква постоянна връзка между възлите. Приложението, работещо на сървъра, се състои от два компонента - база данни и работна логика. Изборът на системата за управление на бази от данни и скриптовия език на сървъра е обоснован основно от преследваната висока финансова ефективност. Към настоящия момент най-изгодно е използването на комбинацията MySQL и PHP, както е в проекта VitoshaTrade[4] (Фиг. 5.1).



Фигура 5.1: Публично хранилище за кода на сървър приложението

## 5.1 Релационна база данни

За нуждите от съхраняването на информация от страна на сървъра, е достатъчно да се разработи база от данни, в която таблиците са нормализирани поне до трета нормална форма (Фиг. 5.2).



Фигура 5.2: Физическа структура на базата данни

Това, което всеки един финансов времеви ред има, е интервал на който се извършват отчитанията на нивата. Тази информация се записва в таблица с три колони - служебен идентификатор, брой минути за интервала и символно название на интервала (Фиг. 5.3).

```

215 -- Table structure for table `time_periods`
216 --
217
218 CREATE TABLE IF NOT EXISTS `time_periods` (
219   `id` int(11) NOT NULL AUTO_INCREMENT COMMENT 'Unique identifier.',
220   `minutes` int(11) NOT NULL COMMENT 'Number of minutes.',
221   `period` varchar(10) NOT NULL COMMENT 'Time period as symbols.',
222   PRIMARY KEY (`id`)
223 ) ENGINE=MyISAM DEFAULT CHARSET=utf8 COMMENT='Time periods minutes to
224 symbols mapping.' AUTO_INCREMENT=10 ;
225
226
227 -- Dumping data for table `time_periods`
228 --
229
230 INSERT INTO `time_periods` (`id`, `minutes`, `period`) VALUES
231 (1, 1, 'M1'),
232 (2, 5, 'M5'),
233 (3, 15, 'M15'),
234 (4, 30, 'M30'),
235 (5, 60, 'H1'),
236 (6, 240, 'H4'),
237 (7, 1440, 'D1'),
238 (8, 10080, 'W1'),
239 (9, 43200, 'MN1');
240
241 -----
242
243 --
244 -- Table structure for table `training_options`
245

```

Фигура 5.3: Таблица с времевите интервали за редовете

Най-съществената таблица в базата данни е таблицата за описване на валутните двойки (Фиг. 5.4). Тя съдържа следните полета - служебен идентификатор, название на валутната двойка, външен ключ към времевия интервал, отместване, необходимо за нормализиране на информацията, множител за мащабиране, също необходим за нормализирането на информацията, описание на валутната двойка и маркер за време, в което е направен записът в таблицата.

```

180 activities longtext NOT NULL COMMENT 'Artificial neural network weights activ
181 `happened` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP COMMENT 'The moment in
182 PRIMARY KEY (`id`),
183 KEY `symbol` (`currency_pairs_id`, `number_of_neurons`)
184 ) ENGINE=MyISAM DEFAULT CHARSET=utf8 COMMENT='Artificial neural network kind.'
185
186 --
187 -- Table structure for table `currency_pairs`
188 --
189
190 CREATE TABLE IF NOT EXISTS `currency_pairs` (
191 `id` int(11) NOT NULL AUTO_INCREMENT COMMENT 'Currency pairs unique identifier.',
192 `symbol` varchar(255) NOT NULL COMMENT 'MetaTrader 4 chart symbol.',
193 `period_id` int(11) NOT NULL COMMENT 'Time period ID.',
194 `offset` decimal(10,0) NOT NULL COMMENT 'Chart offset to map in ANN with 0.0-1.
195 `scale` decimal(10,0) NOT NULL COMMENT 'Chart scaling to map in ANN with 0.0-1.
196 `description` varchar(255) NOT NULL COMMENT 'Describe what kind of currency pa
197 `happened` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP COMMENT 'The moment in
198 PRIMARY KEY (`id`)
199 ) ENGINE=MyISAM DEFAULT CHARSET=utf8 AUTO_INCREMENT=2 ;
200
201 --
202 -- Table structure for table `neurons_coordinates`
203 --
204
205 CREATE TABLE IF NOT EXISTS `neurons_coordinates` (
206 `id` int(11) NOT NULL AUTO_INCREMENT COMMENT 'Unique identifier.',
207 `ann_id` int(11) NOT NULL COMMENT 'Link to ANN table.',
208 `coordinates` longtext NOT NULL COMMENT 'Neurons coordinates (x, y).',
209 `happened` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP COMMENT 'The moment in
210

```

Фигура 5.4: Таблица за представяне на валутните двойки

Топологията на всяка невронна мрежа се описва само един път в таблица за видовете мрежи (Фиг. 5.5). Тази таблица съдържа следните полета - служебен идентификатор, външен ключ към валутната двойка, за която се отнася мрежата, брой неврони, флагове за типа на невроните, стойности за връзките между невроните (според матрицата за съседство) и в кой момент от времето е направен записът в таблицата.

```

168     KEY `ann_kind_id` (`ann_kind_id`,`fitness`)
169 ) ENGINE=MyISAM DEFAULT CHARSET=utf8 COMMENT='Artificial neural network.' AUTO_I
170
171 --
172 -- Table structure for table `ann_kind`
173 --
174
175 CREATE TABLE IF NOT EXISTS `ann_kind` (
176   `id` int(11) NOT NULL AUTO_INCREMENT COMMENT 'Artificial neural network unique
177   `currency_pairs_id` int(11) NOT NULL COMMENT 'MetaTrader 4 chart currency pair
178   `number_of_neurons` int(10) unsigned NOT NULL COMMENT 'Number of neurons used f
179   `flags` longtext NOT NULL COMMENT 'Neurons flags (bias, input, output).',
180   `activities` longtext NOT NULL COMMENT 'Artificial neural network weights activ
181   `happened` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP COMMENT 'The moment in
182   PRIMARY KEY (`id`),
183   KEY `symbol` (`currency_pairs_id`,`number_of_neurons`)
184 ) ENGINE=MyISAM DEFAULT CHARSET=utf8 COMMENT='Artificial neural network kind.' A
185
186 --
187 -- Table structure for table `currency_pairs`
188 --
189
190 CREATE TABLE IF NOT EXISTS `currency_pairs` (
191   `id` int(11) NOT NULL AUTO_INCREMENT COMMENT 'Currency pairs unique identifier.
192   `symbol` varchar(255) NOT NULL COMMENT 'MetaTrader 4 chart symbol.',
193   `period_id` int(11) NOT NULL COMMENT 'Time period ID.',
194   `offset` decimal(10,0) NOT NULL COMMENT 'Chart offset to map in ANN with 0.0-1.
195   `scale` decimal(10,0) NOT NULL COMMENT 'Chart scaling to map in ANN with 0.0-1.
196   `description` varchar(255) NOT NULL COMMENT 'Describe what kind of currency pai
197   `happened` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP COMMENT 'The moment in

```

Фигура 5.5: Таблица за представяне на типовете мрежи

На всяка топология мрежа може да отговарят множество екземпляри от тип изкуствена невронна мрежа (Фиг. 5.6). Информацията в таблицата за екземплярите е организирана в следните колони - служебен идентификатор, външен ключ към типа мрежа за който се отнася екземплярът, жизнена стойност на екземпляра, тегла на връзките между невроните (по графа за съседство) и момент от времето, в който е направен записът в таблицата.

```

150     return amount;
151 end$$
152
153 DELIMITER ;
154
155 -----
156
157 --
158 -- Table structure for table `ann`
159 --
160
161 CREATE TABLE IF NOT EXISTS `ann` (
162   `id` int(11) NOT NULL AUTO_INCREMENT COMMENT 'Artificial neural network unique
163   `ann_kind_id` int(11) NOT NULL COMMENT 'Artificial neural network kind foreign
164   `fitness` double unsigned NOT NULL COMMENT 'Artificial neural network weights f
165   `weights` longtext NOT NULL COMMENT 'Artificial neural network weights.',
166   `happened` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP COMMENT 'The moment in
167   PRIMARY KEY (`id`),
168   KEY `ann_kind_id` (`ann_kind_id`,`fitness`)
169 ) ENGINE=MyISAM DEFAULT CHARSET=utf8 COMMENT='Artificial neural network.' AUTO I
170
171 --
172 -- Table structure for table `ann_kind`
173 --
174
175 CREATE TABLE IF NOT EXISTS `ann_kind` (
176   `id` int(11) NOT NULL AUTO_INCREMENT COMMENT 'Artificial neural network unique
177   `currency_pairs_id` int(11) NOT NULL COMMENT 'MetaTrader 4 chart currency pair
178   `number_of_neurons` int(10) unsigned NOT NULL COMMENT 'Number of neurons used f
179   `flags` longtext NOT NULL COMMENT 'Neurons flags (bias input output) '

```

Фигура 5.6: Таблица за представяне на типовете мрежи

За всеки екземпляр на невронна мрежа, е важно да се знае при какви параметри протича обучението му (Фиг. 5.7). За тази цел е предназначена отделна таблица със следните колони - служебен идентификатор, външен ключ към екземпляра, размер на популацията (когато става въпрос за обучаващ генетичен алгоритъм), брой барове на входа и брой барове на изхода.

```

240
241 -----
242
243 --
244 -- Table structure for table `training_options`
245 --
246
247 CREATE TABLE IF NOT EXISTS `training_options` (
248   `id` int(11) NOT NULL AUTO_INCREMENT COMMENT 'Training options unique identifier',
249   `ann_id` int(11) NOT NULL COMMENT 'Link to ANN table.',
250   `de_population_size` int(10) unsigned NOT NULL COMMENT 'Differential evolution population size',
251   `number_of_training_bars` int(10) unsigned NOT NULL COMMENT 'Number of training bars',
252   `after_bars_prediction` int(10) NOT NULL COMMENT 'Prediction after number of bars',
253   PRIMARY KEY (`id`),
254   UNIQUE KEY `ann_id` (`ann_id`)
255 ) ENGINE=MyISAM DEFAULT CHARSET=utf8 COMMENT='Artificial neural network training options'
256
257 --
258 -- Table structure for table `training_set`
259 --
260
261 CREATE TABLE IF NOT EXISTS `training_set` (
262   `id` int(11) NOT NULL AUTO_INCREMENT COMMENT 'Unique identifier.',
263   `currency_pairs_id` int(11) NOT NULL COMMENT 'Currency pair ID.',
264   `number_of_examples` int(11) NOT NULL COMMENT 'Number of training examples.',
265   `time` longtext NOT NULL COMMENT 'Time values of the time series.',
266   `open` longtext NOT NULL COMMENT 'Open values of time series.',
267   `low` longtext NOT NULL COMMENT 'Low values of time series.',
268   `high` longtext NOT NULL COMMENT 'High values of time series.',
269   `close` longtext NOT NULL COMMENT 'Close values of time series.',
270

```

Фигура 5.7: Таблица с опции за протичане на обучението

При нужда от визуално представяне на екземпляра за изкуствена невронна мрежа, е удачно да се съхранява информация за координатите на всеки неврон в равнината xOy (Фиг. 5.8). Тази таблица съдържа следните колони - служебен идентификатор, външен ключ към екземпляра, координати на невроните по реда на тяхното срещане в екземпляра и маркер за времето, в което е направен записът в таблицата.

```

196 `description` varchar(255) NOT NULL COMMENT 'Describe what kind of currency pair
197 `happened` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP COMMENT 'The moment in
198 PRIMARY KEY (`id`)
199 ) ENGINE=MyISAM DEFAULT CHARSET=utf8 AUTO_INCREMENT=2;
200
201
202 -- Table structure for table `neurons_coordinates`
203 --
204
205 CREATE TABLE IF NOT EXISTS `neurons_coordinates` (
206   `id` int(11) NOT NULL AUTO_INCREMENT COMMENT 'Unique identifier.',
207   `ann_id` int(11) NOT NULL COMMENT 'Link to ANN table.',
208   `coordinates` longtext NOT NULL COMMENT 'Neurons coordinates (x, y).',
209   `happened` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP COMMENT 'The moment in
210   PRIMARY KEY (`id`),
211   UNIQUE KEY `ann_id` (`ann_id`)
212 ) ENGINE=MyISAM DEFAULT CHARSET=utf8 COMMENT='Backend neurons visualisation infor
213
214
215 -- Table structure for table `time_periods`
216 --
217
218 CREATE TABLE IF NOT EXISTS `time_periods` (
219   `id` int(11) NOT NULL AUTO_INCREMENT COMMENT 'Unique identifier.',
220   `minutes` int(11) NOT NULL COMMENT 'Number of minutes.',
221   `period` varchar(10) NOT NULL COMMENT 'Time period as symbols.',
222   PRIMARY KEY (`id`)
223 ) ENGINE=MyISAM DEFAULT CHARSET=utf8 COMMENT='Time periods minutes to
224 symbols mapping.' AUTO_INCREMENT=10 ;
225

```

Фигура 5.8: Таблица с опции за протичане на обучението

Последната, но изключително важна таблица, е таблицата за съхранение на обучаващите примери, съдържащи информацията за финансовия времеви ред (Фиг. 5.9). Тази таблица съдържа следните колони - служебен идентификатор, външен ключ към валутната двойка, брой измервания, последователност от времеви маркери, последователност от нива (отваряне, най-ниско, най-високо, затваряне), търгуван обем и времеви маркер за момента, в който е направен записът в таблицата.

```

254     UNIQUE KEY `ann_id` (`ann_id`)
255 ) ENGINE=MyISAM DEFAULT CHARSET=utf8 COMMENT='Artificial neural network training
256
257 --
258 -- Table structure for table `training_set`
259 --
260
261 CREATE TABLE IF NOT EXISTS `training_set` (
262   `id` int(11) NOT NULL AUTO_INCREMENT COMMENT 'Unique identifier.',
263   `currency_pairs_id` int(11) NOT NULL COMMENT 'Currency pair ID.',
264   `number_of_examples` int(11) NOT NULL COMMENT 'Number of training examples.',
265   `time` longtext NOT NULL COMMENT 'Time values of the time series.',
266   `open` longtext NOT NULL COMMENT 'Open values of time series.',
267   `low` longtext NOT NULL COMMENT 'Low values of time series.',
268   `high` longtext NOT NULL COMMENT 'High values of time series.',
269   `close` longtext NOT NULL COMMENT 'Close values of time series.',
270   `volume` longtext NOT NULL COMMENT 'Volume values of time series.',
271   `happened` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP COMMENT 'The moment in
272   PRIMARY KEY (`id`)
273 ) ENGINE=MyISAM DEFAULT CHARSET=utf8 COMMENT='Training set examples table.' AUTO
274
275 /*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
276 /*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
277 /*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
278
    
```

Фигура 5.9: Таблица с опции за протичане на обучението

## 5.2 Алгоритмична обработка на суровите данни

За да се постигне добро разделяне, според трислойната архитектура на системата, от съществено значение е суровите данни да бъдат минимално обвързани с работната логика. Често срещана практика е смесването на работата със суровите данни и програмния код от работната логика. Такова смесване лесно се получава, ако в скриптовете от работната логика се изпълняват сложни SQL заявки. Подобен начин за конструиране на сървърното приложение прави системата трудна за поддържане и още по-трудна за реорганизиране, ако се наложи смяна на системата за управление на бази от данни или инструментите използвани в слоя на работната логика. Добро разделяне между двата слоя се получава, когато работната логика, под формата на SQL заявки, единствено вика съхранени функции и процедури (stored procedures) на системата за управление на бази от данни. От една страна синтаксисът на заявките в междинния слой става максимално опростен, от друга страна евентуална подмяна на базата данни би довела до единствена корекция в начина, по който се извикват съхранените функции и процедури. За да се удовлетвори стремежът за максимално разделяне между слоевете, към структурата на базата данни са добавени група съхранени функции и процедури.

```

17
18 CREATE DEFINER=`root`@`localhost` PROCEDURE `insertCurrencyPair`(
19     IN `symbol` VARCHAR(255), IN `period` INT(10))
20     MODIFIES SQL DATA
21     COMMENT 'Insert currency pair with currency symbol and minutes period.'
22 begin
23     declare period_id int;
24
25     select id    into period_id
26     from time_periods
27     where time_periods.minutes=period;
28
29     insert into currency_pairs (symbol, period_id, description)
30     values (symbol, period_id, "");
31 end$$
32
33 CREATE DEFINER=`root`@`localhost` PROCEDURE `saveAnn`(IN `symbol` VARCHAR(255), I
34     MODIFIES SQL DATA
35     COMMENT 'Save ANN information in many tables.'
36 begin
37     declare currency_pairs_id int;
38     declare ann_kind_id int;
39
40     set currency_pairs_id = checkCurrencyPairsId(symbol,period);
41     if currency_pairs_id=0 then
42         call insertCurrencyPair(symbol, period);
43         set currency_pairs_id = checkCurrencyPairsId(symbol,period);
44     end if;
45
46     set ann kind id = checkAnnKindId(symbol. period. number of neurons. flaas. ac

```

Фигура 5.10: Процедура за добавяне на валутна двойка

При добавянето на нова валутна двойка, като входни данни постъпват названието на валутната двойка и времевият интервал (в брой минути), който интервал трябва да бъде съпоставен на интервалите, изброени в таблицата с времеви интервали. Това налага използването на съхранена процедура, която да извърши обработката на входната информация (Фиг. 5.10).

```

33 CREATE DEFINER=`root`@`localhost` PROCEDURE `saveAnn` (
34         IN `symbol` VARCHAR(255), IN `period` INT(10),
35         IN `number_of_neurons` INT(10),
36         IN `flags` TEXT, IN `activities` TEXT,
37         IN `fitness` DOUBLE, IN `weights` TEXT)
38     MODIFIES SQL DATA COMMENT 'Save ANN information in many tables.'
39     begin
40         declare currency_pairs_id int;
41         declare ann_kind_id int;
42
43         set currency_pairs_id = checkCurrencyPairsId(symbol,period);
44         if currency_pairs_id=0 then
45             call insertCurrencyPair(symbol, period);
46             set currency_pairs_id = checkCurrencyPairsId(symbol,period);
47         end if;
48
49         set ann_kind_id = checkAnnKindId(symbol, period, number_of_neurons,
50             flags, activities);
51         if ann_kind_id=0 then
52             insert into ann_kind (currency_pairs_id, number_of_neurons, flags,
53                 activities) values (currency_pairs_id,
54                 number_of_neurons, flags, activities);
55             set ann_kind_id = checkAnnKindId(symbol, period, number_of_neurons,
56                 flags, activities);
57         end if;
58
59         insert into ann (ann_kind_id, fitness, weights) values (ann_kind_id,
60             fitness, weights);
61     end$$
62

```

Фигура 5.11: Процедура за запис на екземпляр от изкуствена невронна мрежа

При записването на екземпляр от изкуствена невронна мрежа, към базата данни постъпва информация за валутната двойка, периода на времевия ред, броя и типовете на невроните, матрици на съседство за връзките и жизнена оценка на екземпляра (Фиг. 5.11). Преди екземплярът да бъде добавен в таблицата с екземплярите се проверява дали валутната двойка е налична и се определя нейният идентификатор. Ако валутната двойка не е налична, то тя се създава. Следва проверка за съществуването на типа мрежа. Ако не съществува, то тя се създава. Последната инструкция е самото добавяне на екземпляра към таблицата с екземплярите. При така подбраната стратегия за работа максимално се спазват правилата за съблюдаване на референциалния интегритет.

```

62
63 CREATE DEFINER=`root`@`localhost` PROCEDURE `saveTrainingSet` (
64     IN `symbol` VARCHAR(255), IN `period` INT,
65     IN `number_of_examples` INT, IN `time` TEXT,
66     IN `open` TEXT, IN `low` TEXT, IN `high` TEXT,
67     IN `close` TEXT, IN `volume` TEXT)
68     MODIFIES SQL DATA COMMENT 'Save training set values.'
69 begin
70     declare currency_pairs_id int;
71     declare id int;
72
73     set currency_pairs_id = checkCurrencyPairsId(symbol,period);
74     if currency_pairs_id=0 then
75         call insertCurrencyPair(symbol, period);
76         set currency_pairs_id = checkCurrencyPairsId(symbol,period);
77     end if;
78     -- Delete previous record, because information will be old.
79     select training_set.id into id from training_set, currency_pairs,
80         time_periods where training_set.currency_pairs_id=currency_pairs.id
81         and currency_pairs.period_id=time_periods.id and
82         currency_pairs.symbol=symbol and time_periods.minutes=period;
83     delete from training_set where training_set.id = id;
84     -- Insert new data for the same record.
85     insert into training_set (training_set.currency_pairs_id,
86         training_set.number_of_examples, training_set.time,
87         training_set.open, training_set.low, training_set.high,
88         training_set.close, training_set.volume) values (currency_pairs_id,
89         number_of_examples, time, open, low, high, close, volume);
90 end$$
91

```

expected:  
expected: TRANSACTION

Фигура 5.12: Процедура за запис на тренировъчни примери

При записа на тренировъчни примери, на входа на базата данни се подават серия параметри за стойностите на времевия ред (Фиг. 5.12). По аналогичен начин се извършва проверка за съществуване на валутната двойка. Следва изтриване на предишни тренировъчни примери, ако такива бъдат открити. Финалната стъпка е добавянето на постъпилата информация.

```

92
93 -- Functions
94
95
96 CREATE DEFINER=`root`@`localhost` FUNCTION `checkAnnKindId`(
97     `symbol` VARCHAR(255), `period` INT(10),
98     `number_of_neurons` INT(10), `flags` TEXT,
99     `activities` TEXT) RETURNS int(11)
100
101     READS SQL DATA
102     COMMENT 'Check for ANN kind id. Return 0 if it was not found.'
103 begin
104     declare id int;
105
106     select ann_kind.id into id from ann_kind, currency_pairs, time_periods
107     where ann_kind.currency_pairs_id=currency_pairs.id
108     and currency_pairs.period_id=time_periods.id
109     and currency_pairs.symbol=symbol
110     and time_periods.minutes=period
111     and ann_kind.number_of_neurons=number_of_neurons
112     and ann_kind.flags=flags
113     and ann_kind.activities=activities;
114
115     if id is null then
116         set id = 0;
117     end if;
118
119     return id;
120 end$$
121
122 CREATE DEFINER=`root`@`localhost` FUNCTION `checkCurrencyPairsId`(`symbol` VARCHA

```

Фигура 5.13: Функция за определяне на типа мрежа

Помощна функция определя служебния идентификатор на типа мрежа по описанието на екземпляр (Фиг. 5.13). Ако не бъде открит такъв тип служебно се връща нулева стойност.

```

115     set id = 0;
116 end if;
117
118     return id;
119 end$$
120
121 CREATE DEFINER=`root`@`localhost` FUNCTION `checkCurrencyPairsId` (
122     `symbol` VARCHAR(255), `period` INT(10))
123     RETURNS int(11)
124     READS SQL DATA
125     COMMENT 'Check for currency pair id. Return 0 if it was not found.'
126 begin
127
128     declare id int;
129
130     select currency_pairs.id into id from currency_pairs, time_periods
131     where currency_pairs.period_id=time_periods.id
132     and currency_pairs.symbol=symbol
133     and time_periods.minutes=period;
134
135     if id is null then
136         set id = 0;
137     end if;
138
139     return id;
140 end$$
141
142 CREATE DEFINER=`root`@`localhost` FUNCTION `listAnns` (`id` INT) RETURNS text CHARACTER SET utf8
143     READS SQL DATA
144     COMMENT 'List ids of all ANN from particular type.'
145

```

Фигура 5.14: Функция за определяне на типа валутна двойка

По аналогичен начин, помощна функция служи за определяне на служебния идентификатор за валутна двойка, чрез зададено символно име и период (Фиг. 5.14). Ако валутната двойка не бъде открита се връща служебна нула.

```

153
154 CREATE DEFINER=`root`@`localhost` FUNCTION `loadBestFitness` (
155     `symbol` VARCHAR(255), `period` INT,
156     `number_of_neurons` INT, `flags` TEXT, `activities` TEXT)
157     RETURNS double
158     READS SQL DATA
159     COMMENT 'Load best fitness for particular ANN kind.'
160     begin
161     declare best double;
162     declare ann_kind_id int;
163
164     select ann_kind.id into ann_kind_id from ann_kind, currency_pairs,
165         time_periods where currency_pairs.symbol=symbol
166         and time_periods.minutes=period
167         and ann_kind.number_of_neurons=number_of_neurons
168         and ann_kind.flags=flags
169         and ann_kind.activities=activities
170         and ann_kind.currency_pairs_id=currency_pairs.id
171         and currency_pairs.period_id=time_periods.id;
172
173     select min(ann.fitness) into best from ann where ann.ann_kind_id=ann_kind_id;
174
175     if best is null then
176     set best = 10000;
177     end if;
178
179     return best;
180 end$$
181
182 CREATE DEFINER=`root`@`localhost` FUNCTION `loadNeuronsAmount` (`id` INT) RETURNS
183

```

Фигура 5.15: Функция за определяне на най-добрата жизненост

По зададено описание на изкуствена невронна мрежа, клиентските мобилни устройства запитват сървъра, на определени интервали от време, колко е стойността на най-жизнената мрежа. Това запитване е съществено, тъй като по него се взема решение дали мобилното устройство да докладва резултатите си на сървъра. Обработката се извършва на две стъпки - първо се намират всички мрежи с идентична топология, а след това се определя най-добрата жизненост (Фиг. 5.15).

```

171         and currency_pairs.period_id=time_periods.id;
172
173     select min(ann.fitness) into best from ann where ann.ann_kind_id=ann_kind_id;
174
175     if best is null then
176         set best = 10000;
177     end if;
178
179     return best;
180 end$$
181
182 CREATE DEFINER=`root`@`localhost` FUNCTION `loadNeuronsAmount`(`id` INT)
183     RETURNS int(11)
184     READS SQL DATA
185     COMMENT 'Check neurons amount in particular ANN.'
186 begin
187     declare amount int;
188
189     select ann_kind.number_of_neurons into amount from ann, ann_kind
190     where ann.id=id and ann.ann_kind_id=ann_kind.id;
191
192     if amount is null then
193         set amount = 0;
194     end if;
195
196     return amount;
197 end$$
198
199
200

```

Фигура 5.16: Функция за определяне на броя неврони за определен еземпляр мрежа

Помощна функция определя и броя неврони по зададен идентификатор на екземпляр мрежа (Фиг. 5.16).

### 5.3 Сървър скриптове

Директният достъп до базата данни (TCP свързване) често е нежелателен, тъй като създава определени опасности за сигурността на данните. Поради тази причина е прието да се използва междинен слой, който да осъществява комуникацията с базата данни. За нуждите на настоящото помагало това са група PHP скриптове.

```

29
30 <?php
31 /**
32  * Host name.
33  */
34 define("HOSTNAME", "localhost");
35
36 /**
37  * Database username.
38  */
39 define("USERNAME", "root");
40
41 /**
42  * Database user password.
43  */
44 define("PASSWORD", "");
45
46 /**
47  * Database name.
48  */
49 define("DBNAME", "veldoftware_vitoshatrade");
50
51 /**
52  * Database link global variable.
53  */
54 $GLOBALS[ 'link' ] = null;
55
56 /**
57  * Open global database link.
58  *
59  * @author Todor Balabanov

```

Фигура 5.17: Глобални променливи за достъп до базата данни

В специално отделен за целта файл (db.php) се задават следните глобални променливи за достъп до базата данни - хост на който е разположен MySQL сървърът, потребител в MySQL, парола на потребителя, название на базата данни и връзка (Фиг. 5.17).

```

52  * Database link global variable.
53  */
54  $GLOBALS[ 'link' ] = null;
55
56  /**
57  * Open global database link.
58  *
59  * @author Todor Balabanov
60  *
61  * @email tdb@tbsoft-bg.com
62  *
63  * @date 27 Apr 2009
64  */
65  function open_db() {
66      $GLOBALS[ 'link' ] = mysql_connect(HOSTNAME, USERNAME, PASSWORD);
67      mysql_select_db( DBNAME );
68  }
69
70  /**
71  * Database query.
72  *
73  * @param $qrystr SQL query string.
74  *
75  * @return Array with SQL query result.
76  *
77  * @author Todor Balabanov
78  *
79  * @email tdb@tbsoft-bg.com
80  *
81  * @date 27 Apr 2009
82  */

```

Фигура 5.18: Отваряне на връзка към базата данни

Две помощни функции служат за отваряне и затваряне на връзка към базата данни (Фиг. 5.18, 5.19).

```

102         $j++;
103     }
104     } else {
105         $resstrs = false;
106     }
107 }
108
109 return( $resstrs );
110 }
111
112 /**
113  * Close global database link.
114  *
115  * @author Todor Balabanov
116  *
117  * @email tdb@tbsoft-bg.com
118  *
119  * @date 27 Apr 2009
120  */
121 function close_db() {
122     if ( $GLOBALS['link'] ) {
123         mysql_close( $GLOBALS['link'] );
124     }
125 }
126 ?>
127

```

Фигура 5.19: Затваряне на връзката към базата данни

Допълнителна помощна функция изпълнява заявките към базата данни и връща резултата от изпълнението под формата на двумерен масив (Фиг. 5.20).

```

82  */
83  function query_db( $qrystr ) {
84      $resstrs = false;
85
86      if($qrystr[strlen($qrystr)-1] == ',') {
87          $qrystr[strlen($qrystr)-1] = " ";
88      }
89
90      if ( !$GLOBALS['link'] ) {
91          $resstrs = false;
92      } else {
93          $result = mysql_query($qrystr, $GLOBALS['link']);
94
95          $j = 0;
96          if ($result!=1 && $result!=false) {
97              while ($row = mysql_fetch_row($result)) {
98                  for ($i=0; $i<mysql_num_fields($result); $i++) {
99                      $resstrs[ $j ] [ $i ] = $row[ $i ];
100                  }
101
102                  $j++;
103              }
104          } else {
105              $resstrs = false;
106          }
107      }
108
109      return( $resstrs );
110  }
111
112

```

Фигура 5.20: Изпълнение на заявки към базата данни

На този етап в разработката клиентското приложение изпраща своите данни под формата на POST заявка, а сървърът отговаря с JSON съобщение. Макар и спестяващ допълнителна работа около съставянето на JSON съобщения от страна на клиента, този подход не е за препоръчване и е желателно клиентите също да изпращат информацията в JSON формат, така че да се спазват конвенциите на RESTful приложенията.

### 5.3.1 Зареждане на мрежа по идентификатор

Всеки екземпляр на изкуствена невронна мрежа има служебен идентификатор, който представлява и първичен ключ в съответната таблица. За да бъде заредена мрежата, се извиква PHP скрипт изпълняващ тази задача.

```

29
30 <?php
31 /*
32  * Include database functions.
33  */
34 include "../common/db.php";
35
36 header("access-control-allow-origin: *");
37
38 /*
39  * Response output variable.
40  */
41 $response = '';
42
43 /*
44  * Check input parameters.
45  */
46 if ( !isset($_POST['annid']) ) {
47     $annid = -1;
48 } else {
49     $annid = intval($_POST['annid']);
50 }
51
52 /*
53  * Open database.
54  */
55 open_db();
56
57 /*
58  * Prepare SQL query.
59  */

```

Фигура 5.21: Проверка на входните данни

Преди същинското зареждане на мрежата се включва модулът за работа с базата данни и се извършва проверка на входните параметри. В случая единственият параметър е цяло число което е идентификатор на екземпляра (Фиг. 5.21).

```

40 if ( !isset($_POST['annid']) ) {
41     $annid = -1;
42 } else {
43     $annid = intval($_POST['annid']);
44 }
45
46
47
48
49
50
51
52
53 /*
54  * Open database.
55  */
56 open_db();
57
58 /*
59  * Prepare SQL query.
60  */
61 $sql = 'select currency_pairs.symbol, time_periods.minutes, ann.fitness,
62         ann_kind.number_of_neurons, ann_kind.flags, ann.weights,
63         ann_kind.activities from ann, ann_kind, currency_pairs,
64         time_periods where ann_kind.currency_pairs_id=currency_pairs.id
65         and currency_pairs.period_id=time_periods.id
66         and ann.id= ' . $annid . ' and ann.ann_kind_id=ann_kind.id;';
67
68 /*
69  * Run SQL query.
70  */
71 $result = query_db( $sql );
72
73 /*
74  * Check SQL query result.
75  */
76 $response .= "{";
77 if ($result != false) {

```

Фигура 5.22: Заявка за извличане на екземпляр по идентификатор

Следва отваряне на връзка към базата данни, съставяне на заявка (добрата практика изисква извикване на съхранена функция) и изпълнение на заявката (Фиг. 5.22).

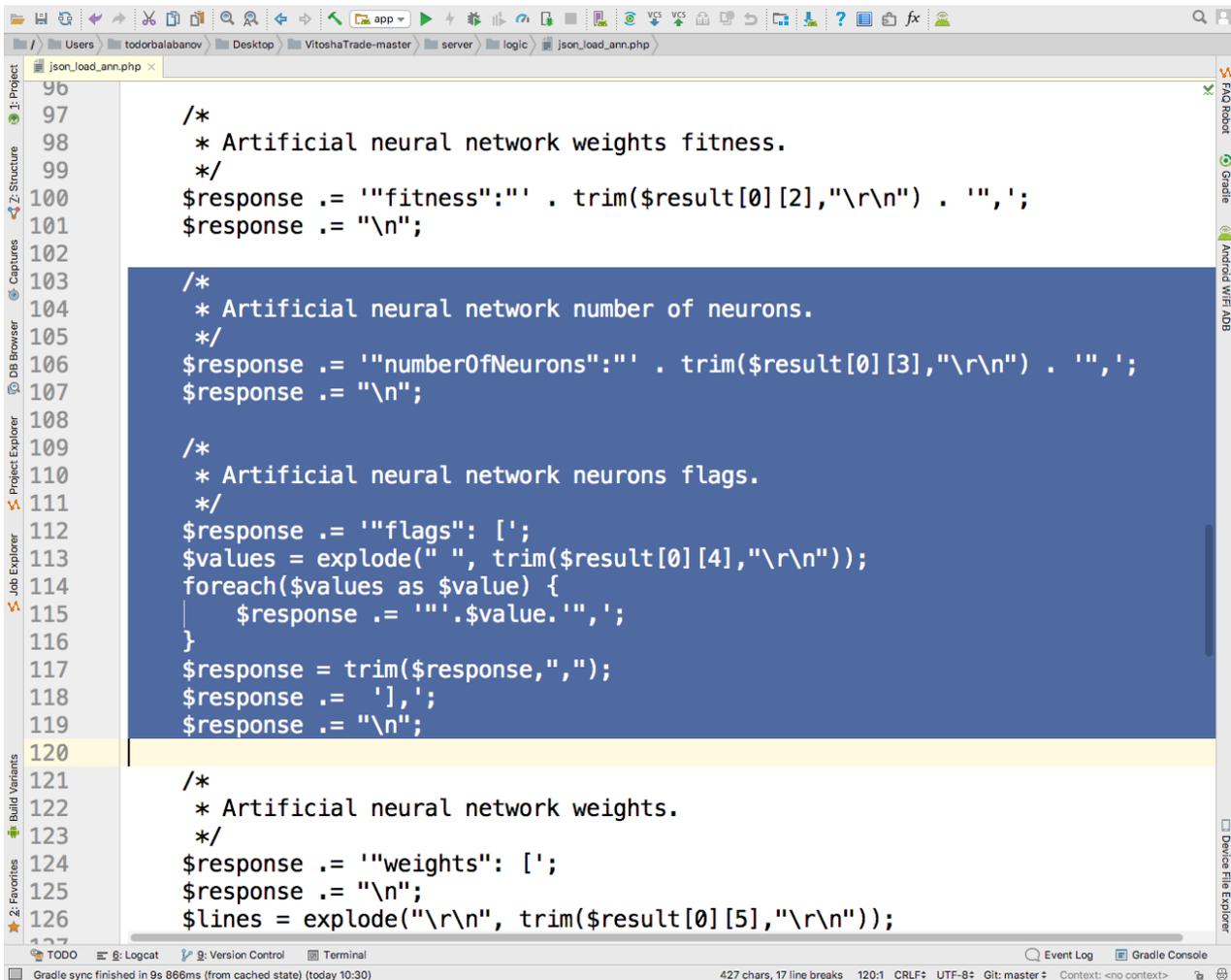
```

72  /*
73  * Check SQL query result.
74  */
75  $response .= "{";
76  if ($result != false) {
77      $response .= "\n";
78
79      /*
80      * Response with artificial neural networks structure.
81      */
82      $response .= '"size":' . trim($result[0][0], "\r\n") . ',';
83      $response .= "\n";
84
85      /*
86      * MetaTrader 4 symbol.
87      */
88      $response .= '"symbol":"' . trim($result[0][0], "\r\n") . ',';
89      $response .= "\n";
90
91      /*
92      * MetaTrader 4 period.
93      */
94      $response .= '"period":"' . trim($result[0][1], "\r\n") . ',';
95      $response .= "\n";
96
97      /*
98      * Artificial neural network weights fitness.
99      */
100     $response .= '"fitness":"' . trim($result[0][2], "\r\n") . ',';
101     $response .= "\n";
102

```

Фигура 5.23: Проверка за наличност на екземпляра

Възможни са два изхода от изпълнението на заявката, спрямо това дали такъв екземпляр присъства в базата данни или не присъства. Ако екземплярът бъде открит започва изграждането на JSON съобщение с отговор към клиента. Отговорът съдържа единица, за намерен екземпляр, названието на валутната двойка, интервала на времевия ред, жизнената стойност на екземпляра, брой и типове на невроните, матрица на връзките и матрица на теглата (Фиг. 5.23-5.26).



```
96
97
98  /*
99   * Artificial neural network weights fitness.
100  */
101  $response .= "fitness:" . trim($result[0][2], "\r\n") . ',';
102  $response .= "\n";
103
104  /*
105   * Artificial neural network number of neurons.
106  */
107  $response .= "numberOfNeurons:" . trim($result[0][3], "\r\n") . ',';
108  $response .= "\n";
109
110  /*
111   * Artificial neural network neurons flags.
112  */
113  $response .= "flags: [';
114  $values = explode(" ", trim($result[0][4], "\r\n"));
115  foreach($values as $value) {
116  |   $response .= "'".$value.'',';
117  }
118  $response = trim($response, ",");
119  $response .= '],';
120  $response .= "\n";
121
122  /*
123   * Artificial neural network weights.
124  */
125  $response .= "weights: [';
126  $response .= "\n";
127  $lines = explode("\r\n", trim($result[0][5], "\r\n"));
```

Фигура 5.24: Информация за невроните

```

118 $response .= '],';
119 $response .= "\n";
120
121 /*
122  * Artificial neural network weights.
123  */
124 $response .= '"weights": [';
125 $response .= "\n";
126 $lines = explode("\r\n", trim($result[0][5], "\r\n"));
127 foreach($lines as $line) {
128     $response .= '[';
129     $values = explode(" ", $line);
130     foreach($values as $value) {
131         $response .= "'".$value.'"';
132     }
133     $response = trim($response, ",");
134     $response .= '],';
135     $response .= "\n";
136 }
137 $response = trim($response, "\n,");
138 $response .= "\n";
139 $response .= '],';
140 $response .= "\n";
141
142 /*
143  * Artificial neural network weights activities.
144  */
145 $response .= '"activities": [';
146 $response .= "\n";
147 $lines = explode("\r\n", trim($result[0][6], "\r\n"));
148 foreach($lines as $line) {

```

Фигура 5.25: Информация за връзките

```

130 }
137 $response = trim($response, "\n,");
138 $response .= "\n";
139 $response .= '],';
140 $response .= "\n";
141
142 /*
143  * Artificial neural network weights activities.
144  */
145 $response .= '"activities": [';
146 $response .= "\n";
147 $lines = explode("\r\n", trim($result[0][6], "\r\n"));
148 foreach($lines as $line) {
149     $response .= '[';
150     $values = explode(" ", $line);
151     foreach($values as $value) {
152         $response .= "'".$value.'"';
153     }
154     $response = trim($response, ",");
155     $response .= '],';
156     $response .= "\n";
157 }
158 $response = trim($response, "\n,");
159 $response .= "\n";
160 $response .= '],';
161 $response .= "\n";
162 } else {
163     /*
164     * Response with -1 flag that artificial neural networks can not be loaded.
165     */
166     $response .= '"size":"0"';
167

```

Фигура 5.26: Информация за теглата

В ситуацията, когато екземплярът не е открит се връща нулев флаг. И при двата случая (открит или не) JSON съобщението се завършва, базата данни се затваря и информацията се изпраща към клиента (Фиг. 5.27).

```

153     }
154     $response = trim($response,",");
155     $response .= ',';
156     $response .= "\n";
157 }
158 $response = trim($response,"\n,");
159 $response .= "\n";
160 $response .= ',';
161 $response .= "\n";
162 } else {
163     /*
164     * Response with 0 flag that artificial neural networks can not be loaded.
165     */
166     $response .= '"size":"0"';
167     $response .= "\n";
168 }
169 $response .= "}";
170
171 /*
172 * Close database.
173 */
174 close_db();
175
176 /*
177 * Output response.
178 */
179 echo( $response );
180 flush();
181 ?>
182

```

Фигура 5.27: Приключване на процедурата по изпращане на отговор

### 5.3.2 Зареждане на случайно избрана мрежа

Когато изчисленията се извършват на принципа за дарената изчислителна мощност, логично е изборът на мрежата, която ще бъде допълнително тренирана да е на случаен принцип. За тази цел, скриптът от страна на сървъра избира мрежата по случаен начин.

```

36  * Response output variable.
37  */
38  $response = '';
39
40  /*
41  * Open database.
42  */
43  open_db();
44
45  /*
46  * Prepare SQL query.
47  */
48  $sql = 'select  currency_pairs.symbol,
49                time_periods.minutes,
50                ann.fitness,
51                ann_kind.number_of_neurons,
52                ann_kind.flags,
53                ann.weights,
54                ann_kind.activities
55
56                from
57                | ann, ann_kind, currency_pairs, time_periods
58                where
59                | ann_kind.currency_pairs_id=currency_pairs.id and
60                | currency_pairs.period_id=time_periods.id and
61                | ann.ann_kind_id=ann_kind.id
62                order by
63                | rand()
64                limit 1;';
65
66  /*
67  * Run SQL query

```

Фигура 5.28: Заявка за зареждане на случайна мрежа

Единствената разлика от зареждането на мрежа по идентификатори и случайно избрана мрежа е в SQL заявката, която се изпълнява от MySQL (Фиг. 5.28).

### 5.3.3 Зареждане на най-добрата жизненост от глобалната популация

Един от основните критерии, по които клиентските приложения вземат решение дали да докладват откритите от тях резултати, е стойността на най-добрата жизненост в глобалната популация (популацията, която се съхранява на сървъра). Скриптът, отговарящ за тази проверка, също включва модула за работа с базата данни и извършва серия проверки на входните параметри.

```

40  */
41  if ( !isset($_POST['symbol']) ) {
42      $symbol = "";
43  } else {
44      $symbol = trim($_POST['symbol'], "\r\n");
45  }
46
47  if ( !isset($_POST['period']) ) {
48      $period = 0;
49  } else {
50      $period = intval($_POST['period']);
51  }
52
53  if ( !isset($_POST['number_of_neurons']) ) {
54      $number_of_neurons = 0;
55  } else {
56      $number_of_neurons = intval($_POST['number_of_neurons']);
57  }
58
59  if ( !isset($_POST['flags']) ) {
60      $flags = "";
61  } else {
62      $flags = trim($_POST['flags'], "\r\n");
63  }
64
65  if ( !isset($_POST['activities']) ) {
66      $activities = "";
67  } else {
68      $activities = trim($_POST['activities'], "\r\n");
69  }
70

```

Фигура 5.29: Входни параметри за конкретна топология на невронна мрежа

Проверяват се названието на валутната двойка, периодът на времевия ред, броят и типовете на невроните, както и връзките между тях (Фиг. 5.29).

```

70
71 $tok = strtok($flags, " \r\n");
72 $buffer = "";
73 for ($i=0; $i<$number_of_neurons; $i++) {
74     if ($tok != false) {
75         $buffer .= (intval($tok) . " ");
76     } else {
77         $buffer .= "0" . " ";
78     }
79     $tok = strtok(" \r\n");
80 }
81 $buffer = trim($buffer, " \r\n");
82 $flags = $buffer;
83 $tok = strtok($activities, " \r\n");
84 $buffer = "";
85 for ($j=0; $j<$number_of_neurons; $j++) {
86     for ($i=0; $i<$number_of_neurons; $i++) {
87         if ($tok != false) {
88             $buffer .= (floatval($tok) . " ");
89         } else {
90             $buffer .= "0" . " ";
91         }
92     }
93     $tok = strtok(" \r\n");
94 }
95 $buffer = trim($buffer, " \r\n");
96 $buffer .= "\n";
97 }
98 $buffer = trim($buffer, " \r\n");
99 $activities = $buffer;
100

```

Фигура 5.30: Флагове на невроните и връзките между тях

Тъй като типовете на невроните са представени като масив, а връзките между самите неврони в матрица на съседство, то проверката им изисква малко по-сложна обработка в цикли (Фиг. 5.30).

```

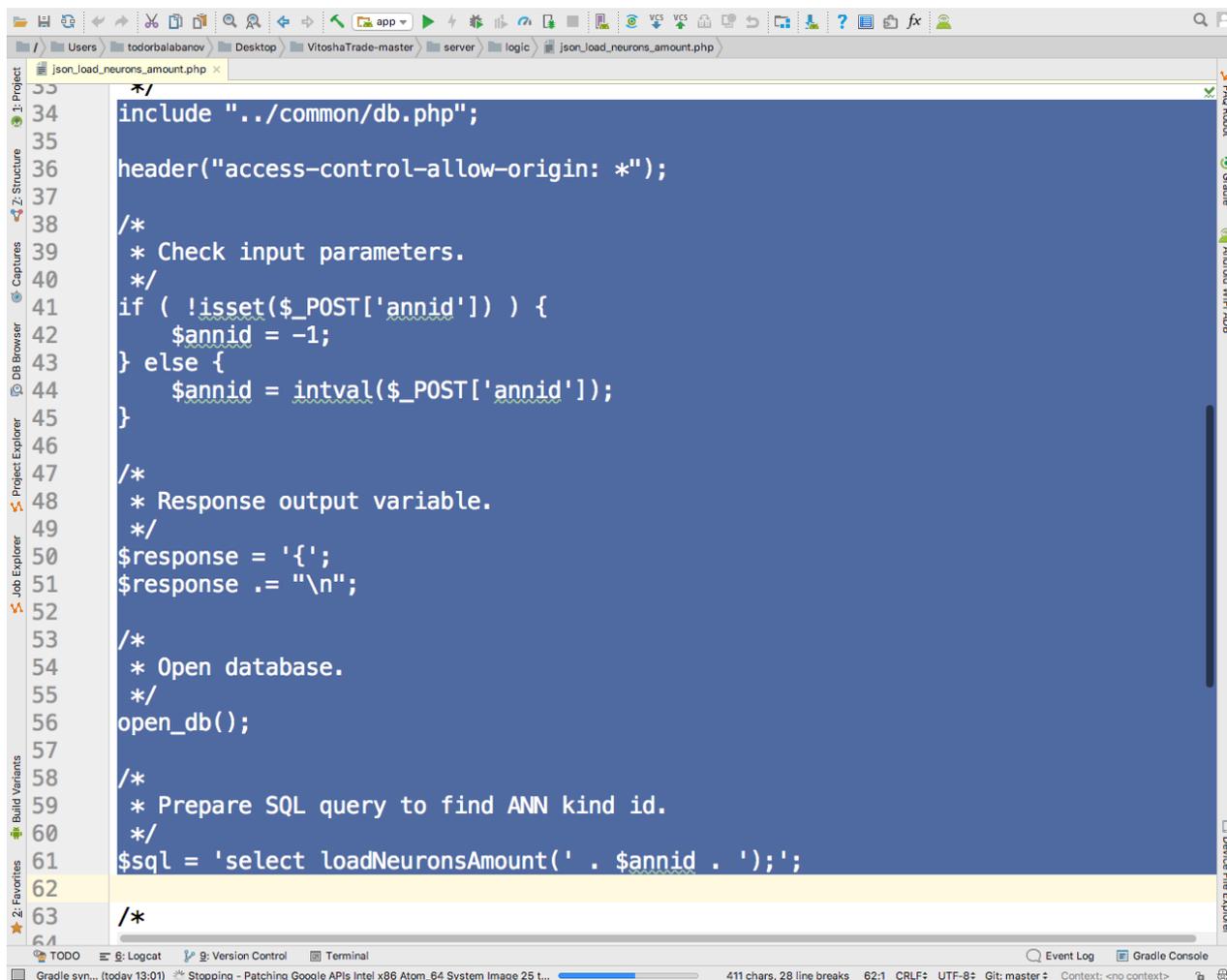
101  /*
102  * Response output variable.
103  */
104  $response = '{}';
105  $response .= "\n";
106  /*
107  * Open database.
108  */
109  open_db();
110  /*
111  * Prepare SQL query to find ANN kind id.
112  */
113  $sql = "select loadBestFitness('" . $symbol . "', " . $period . ", " .
114  $number_of_neurons . ", '" . $flags . "', '" . $activities . "')";
115  $result = query_db( $sql );
116  /*
117  * Close database.
118  */
119  close_db();
120  /*
121  * Response with best fitness or very high unreal value.
122  */
123  $response .= "fitness:" . trim($result[0][0], "\r\n") . "'";
124  $response .= "\n";
125  $response .= "}";
126  /*
127  * Output response.
128  */
129  echo( $response );
130  flush();
131  ?~
    
```

Фигура 5.31: Заявка за най-добра жизненост

Следва отваряне на връзката към базата данни, изпълнение на съхранената функция за откриване на най-добра жизненост, затваряне на базата данни и изпращане до клиента на JSON пакетирани отговор (Фиг. 5.31).

### 5.3.4 Зареждане на брой неврони по идентификатор

При работата с екземпляри на изкуствените невронни мрежи, съществено е защитването за броя неврони в мрежата.

The image shows a screenshot of an IDE window displaying a PHP file named 'json\_load\_neurons\_amount.php'. The code is as follows:

```
34 include "../common/db.php";
35
36 header("access-control-allow-origin: *");
37
38 /*
39  * Check input parameters.
40  */
41 if ( !isset($_POST['annid']) ) {
42     $annid = -1;
43 } else {
44     $annid = intval($_POST['annid']);
45 }
46
47 /*
48  * Response output variable.
49  */
50 $response = '{}';
51 $response .= "\n";
52
53 /*
54  * Open database.
55  */
56 open_db();
57
58 /*
59  * Prepare SQL query to find ANN kind id.
60  */
61 $sql = 'select loadNeuronsAmount(' . $annid . ');';
62
63 /*
64
```

The IDE interface includes a toolbar at the top, a sidebar on the left with various tool windows (Project Explorer, Job Explorer, Build Variants, Favorites, DB Browser, Captures, Structure), and a bottom status bar showing file encoding (UTF-8) and other details.

Фигура 5.32: Определяне на броя неврони по идентификатор на мрежа

Процедурата отново включва модула за работа с базата данни, проверка на входния параметър за идентификатор на екземпляр, отваряне на връзка към базата данни и стартиране на съхранена функция (Фиг. 5.32).

```

54  * open database.
55  */
56  open_db();
57
58  /*
59  * Prepare SQL query to find ANN kind id.
60  */
61  $sql = 'select loadNeuronsAmount(' . $annid . ');';
62
63  /*
64  * Run SQL query.
65  */
66  $result = query_db( $sql );
67
68
69  $response .= '"neuronsAmount":"' . trim($result[0][0],"\r\n") . '"';
70  $response .= "\n";
71  $response .= "}";
72
73  /*
74  * Close database.
75  */
76  close_db();
77
78  /*
79  * Output response.
80  */
81  echo( $response );
82  flush();
83  ?>
84

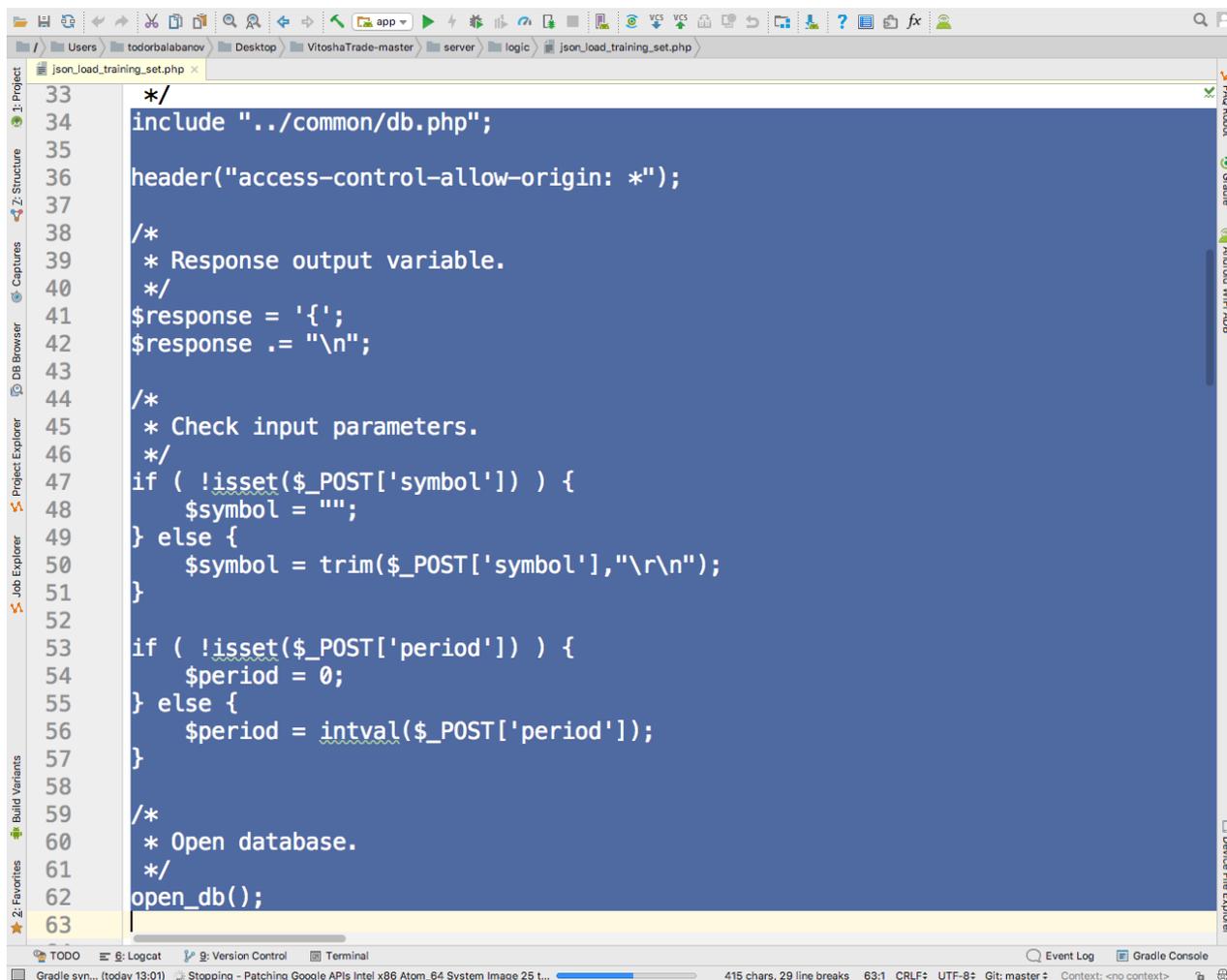
```

Фигура 5.33: Отговор към клиента

Резултатът от съхранената функция се опакова в JSON съобщение и се изпраща на клиента (Фиг. 5.33).

### 5.3.5 Зареждане на тренировъчно множество по информация за валутна двойка

За обучението на всяка изкуствена невронна мрежа се използва тренировъчно множество, което е различно, според названието на валутната двойка и времевия интервал на реда.



```
33  */
34  include "../common/db.php";
35
36  header("access-control-allow-origin: *");
37
38  /*
39   * Response output variable.
40   */
41  $response = '{}';
42  $response .= "\n";
43
44  /*
45   * Check input parameters.
46   */
47  if ( !isset($_POST['symbol']) ) {
48      $symbol = "";
49  } else {
50      $symbol = trim($_POST['symbol'], "\r\n");
51  }
52
53  if ( !isset($_POST['period']) ) {
54      $period = 0;
55  } else {
56      $period = intval($_POST['period']);
57  }
58
59  /*
60   * Open database.
61   */
62  open_db();
63
```

Фигура 5.34: Проверка на входните параметри

Зареждането започва по класическия начин с включване на модула за работа с базата данни, проверка на входната информация и отваряне на връзка към базата данни (Фиг. 5.34).

```

60  * Open database.
61  */
62  open_db();
63
64  /*
65  * Prepare SQL query.
66  */
67  $sql = "select training_set.number_of_examples, training_set.time,
68         training_set.open, training_set.low, training_set.high,
69         training_set.close, training_set.volume from training_set,
70         currency_pairs, time_periods
71         where training_set.currency_pairs_id=currency_pairs.id and
72         currency_pairs.period_id=time_periods.id and
73         currency_pairs.symbol='" . $symbol . "' and
74         time_periods.minutes='" . $period . "'";
75
76  /*
77  * Run SQL query.
78  */
79  $result = query_db( $sql );
80
81  /*
82  * Close database.
83  */
84  close_db();
85
86  /*
87  * Check SQL query result.
88  */
89  if ($result != false) {
90      /*

```

Фигура 5.35: Заявка за тренировъчно множество

Следва заявка, извличане на резултата и затваряне на връзката към базата данни (Фиг. 5.35). Макар и да не е направено, добрият стил за разделяне на словесите предполага вместо заявка да се извика съхранена функция.

```

88  */
89  if ($result != false) {
90      /*
91       * Response with number of training examples.
92       */
93      $response .= "numberOfExamples:" . trim($result[0][0], "\r\n") . ',';
94      $response .= "\n";
95
96      /*
97       * Time values of training examples.
98       */
99      $response .= "time: [';
100     $values = explode(" ", trim($result[0][1], "\r\n"));
101     foreach($values as $value) {
102         $response .= "'".$value.'",';
103     }
104     $response = trim($response, ",");
105     $response .= '],';
106     $response .= "\n";
107
108     /*
109     * Open values of training examples.
110     */
111     $response .= "open: [';
112     $values = explode(" ", trim($result[0][2], "\r\n"));
113     foreach($values as $value) {
114         $response .= "'".$value.'",';
115     }
116     $response = trim($response, ",");
117     $response .= '],';
118     $response .= "\n";
119
120

```

Фигура 5.36: Брой примери, времеви маркери и нива на отваряне

```

115     }
116     $response = trim($response, ",");
117     $response .= '],';
118     $response .= "\n";
119
120     /*
121     * Low values of training examples.
122     */
123     $response .= "'low': [';
124     $values = explode(" ", trim($result[0][3], "\r\n"));
125     foreach($values as $value) {
126         $response .= "'".$value.'",';
127     }
128     $response = trim($response, ",");
129     $response .= '],';
130     $response .= "\n";
131
132     /*
133     * High values of training examples.
134     */
135     $response .= "'high': [';
136     $values = explode(" ", trim($result[0][4], "\r\n"));
137     foreach($values as $value) {
138         $response .= "'".$value.'",';
139     }
140     $response = trim($response, ",");
141     $response .= '],';
142     $response .= "\n";
143
144     /*
145

```

Фигура 5.37: Най-ниска и най-висока постигната стойност

```

139     }
140     $response = trim($response, ",");
141     $response .= '],';
142     $response .= "\n";
143
144     /*
145     * Close values of training examples.
146     */
147     $response .= '"close": [';
148     $values = explode(" ", trim($result[0][5], "\r\n"));
149     foreach($values as $value) {
150         $response .= '"'.$value.'",';
151     }
152     $response = trim($response, ",");
153     $response .= '],';
154     $response .= "\n";
155
156     /*
157     * Volume values of training examples.
158     */
159     $response .= '"volume": [';
160     $values = explode(" ", trim($result[0][6], "\r\n"));
161     foreach($values as $value) {
162         $response .= '"'.$value.'",';
163     }
164     $response = trim($response, ",");
165     $response .= '],';
166     $response .= "\n";
167 } else {
168     /*
169     * Response with zero available training examples.

```

Фигура 5.38: Нива на затваряне и търгуван обем

Ако е намерено тренировъчно множество за посочената валутна двойка и времеви интервал, то стойностите от времевия ред се пакетират и изпращат на клиента (Фиг. 5.36-5.38).

```

154     $response .= "\n";
155
156     /*
157     * Volume values of training examples.
158     */
159     $response .= "volume": [';
160     $values = explode(" ", trim($result[0][6], "\r\n"));
161     foreach($values as $value) {
162         $response .= "'".$value.'"';
163     }
164     $response = trim($response, ",");
165     $response .= ']';
166     $response .= "\n";
167 } else {
168     /*
169     * Response with zero available training examples.
170     */
171     $response .= "numberOfExamples": "0";
172     $response .= "\n";
173 }
174
175 $response .= "}";
176
177 /*
178 * Output response.
179 */
180 echo( $response );
181 flush();
182 ?>
183

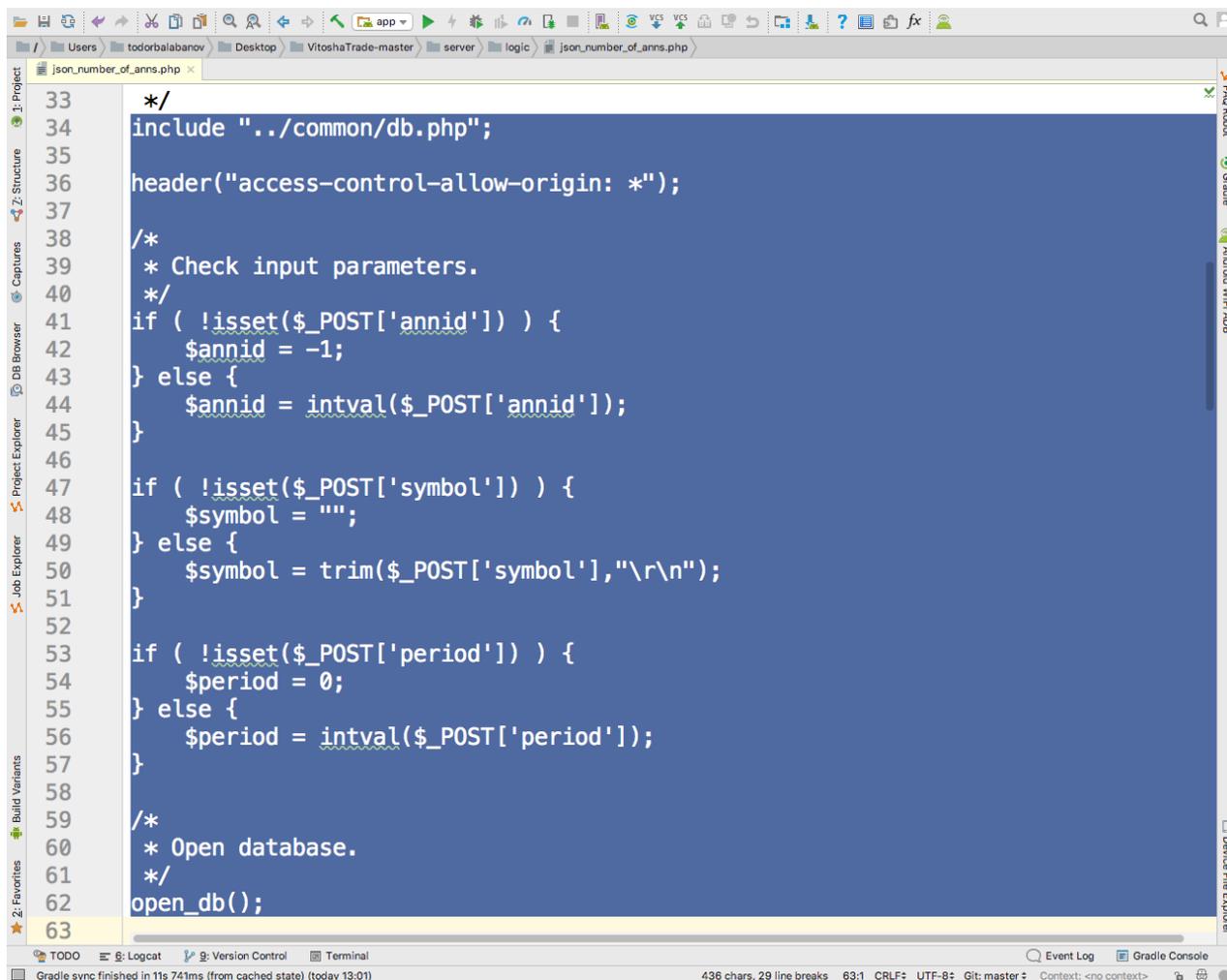
```

Фигура 5.39: Изпращане на отговор до клиента

Ако тренировъчно множество не бъде открито, се изпраща съобщение с нулев размер за данните. Без значение дали е открито множество или не, JSON съобщението се изпраща до клиента (Фиг. 5.39).

### 5.3.6 Зареждане на брой екземпляри по идентификатор или информация за валутна двойка

За подбирането на подмножество от глобалната популация е нужно да се знае колко екземпляра присъстват в базата данни по зададен идентификатор на екземпляр или название на валутна двойка с период.



```
33  */
34  include "../common/db.php";
35
36  header("access-control-allow-origin: *");
37
38  /*
39  * Check input parameters.
40  */
41  if ( !isset($_POST['annid']) ) {
42      $annid = -1;
43  } else {
44      $annid = intval($_POST['annid']);
45  }
46
47  if ( !isset($_POST['symbol']) ) {
48      $symbol = "";
49  } else {
50      $symbol = trim($_POST['symbol'], "\r\n");
51  }
52
53  if ( !isset($_POST['period']) ) {
54      $period = 0;
55  } else {
56      $period = intval($_POST['period']);
57  }
58
59  /*
60  * Open database.
61  */
62  open_db();
63
```

Фигура 5.40: Проверка на входните аргументи

Проверката отново започва с включване на модула за работа с базата данни, проверка на входните аргументи и отваряне на връзка към базата данни (Фиг. 5.40).

```

60  * Open database.
61  */
62  open_db();
63
64  /*
65  * Prepare SQL query by ANN id.
66  */
67  $sql = 'select id from ann where ann.ann_kind_id=(select ann_kind_id from ann
68  where id=' . $annid . ') order by fitness asc;';
69
70  /*
71  * Run SQL query.
72  */
73  $result = query_db( $sql );
74
75  /*
76  * Count number of artificial neural networks available.
77  */
78  $count = 0;
79
80  /*
81  * Response output variable.
82  */
83  $response = '';
84
85  /*
86  * Check SQL query result.
87  */
88  if ( $result != false ) {
89      /*
90      * Response with all artificial neural networks identifiers available

```

Фигура 5.41: Еземпляри от типа на подадения идентификатор

Първо се изброяват екземплярите, които са от типа на подадения идентификатор (Фиг. 5.41).

```

80  /*
81  * Response output variable.
82  */
83  $response = '';
84
85  /*
86  * Check SQL query result.
87  */
88  if ($result != false) {
89      /*
90       * Response with all artificial neural networks identifiers available.
91       */
92      $response .= '"identifiers": [';
93
94      for ($i=0; $i<count($result); $i++) {
95          $response .= ' ' . trim($result[$i][0], "\r\n") . ', ';
96          $count++;
97      }
98
99      $response = trim($response, "\n,");
100     $response .= ']';
101     $response .= "\n";
102 }
103
104 /*
105 * Prepare SQL query by ANN symbol and period.
106 */
107 $sql = 'select id, ann_kind_id from ann where ann.ann_kind_id=(
108     select ann_kind_id from ann_kind, currency_pairs, time_periods
109     where currency_pairs_id=currency_pairs.id and period_id=time_periods.id
110     and symbol=' . $symbol . ' and period=' . $period . ')';

```

Фигура 5.42: Пакетиране на получените стойности в JSON съобщение

Получените стойности се пакетират в JSON съобщение (Фиг. 5.42).

```

101     $response .= "\n";
102 }
103
104 /*
105  * Prepare SQL query by ANN symbol and period.
106  */
107 $sql = 'select id, ann_kind_id from ann where ann.ann_kind_id=(
108     select ann_kind_id from ann_kind, currency_pairs, time_periods
109     where currency_pairs_id=currency_pairs.id and period_id=time_periods.id
110     and symbol=' . $symbol . ' and period=' . $period . ')
111     order by fitness asc;';
112
113 /*
114  * Run SQL query.
115  */
116 $result = query_db( $sql );
117
118 /*
119  * Check SQL query result if there is no list by ANN id.
120  */
121 if ($count==0 && $result!=false) {
122     /*
123      * Artificial neural networks should be from the same kind.
124      */
125     $ann_kind_id = $result[0][1];
126
127     /*
128      * Response with all artificial neural networks identifiers available.
129      */
130     $response .= '"identifiers": [';
131
132

```

Фигура 5.43: Проверка за екземпляри по название на валутна двойка и период

Следва проверка за екземпляри по название на валутна двойка и период. Този списък се използва само, ако първоначално не е открито подмножество (Фиг. 5.43).

```

117
118 /*
119  * Check SQL query result if there is no list by ANN id.
120  */
121 if ($count==0 && $result!=false) {
122     /*
123     * Artificial neural networks should be from the same kind.
124     */
125     $ann_kind_id = $result[0][1];
126
127     /*
128     * Response with all artificial neural networks identifiers available.
129     */
130     $response .= "identifiers: [";
131
132     for ($i=0; $i<count($result); $i++) {
133         if ($result[$i][1] == $ann_kind_id) {
134             $response .= "' . trim($result[$i][0],"\r\n") . ',';
135             $count++;
136         }
137     }
138
139     $response = trim($response,"\n,");
140     $response .= "];";
141     $response .= "\n";
142 }
143
144 /*
145  * Response with number of artificial neural networks available.
146  */
147 $response = "size:'' . $count . ''.' . "\n" . $response:

```

Фигура 5.44: Пакетиране на получените стойности в JSON съобщение

Пакетирането в JSON съобщение е аналогично на предходното (Фиг. 5.44).

```

136     }
137 }
138
139 $response = trim($response, "\n,");
140 $response .= "];";
141 $response .= "\n";
142 }
143
144 /*
145  * Response with number of artificial neural networks available.
146  */
147 $response = "size:" . $count . ',' . "\n" . $response;
148 $response = "{\n" . trim($response, "\n,") . "\n}";
149
150 /*
151  * Close database.
152  */
153 close_db();
154
155 /*
156  * Output response.
157  */
158 echo( trim($response, "\r\n" ) );
159 flush();
160 ?>
161

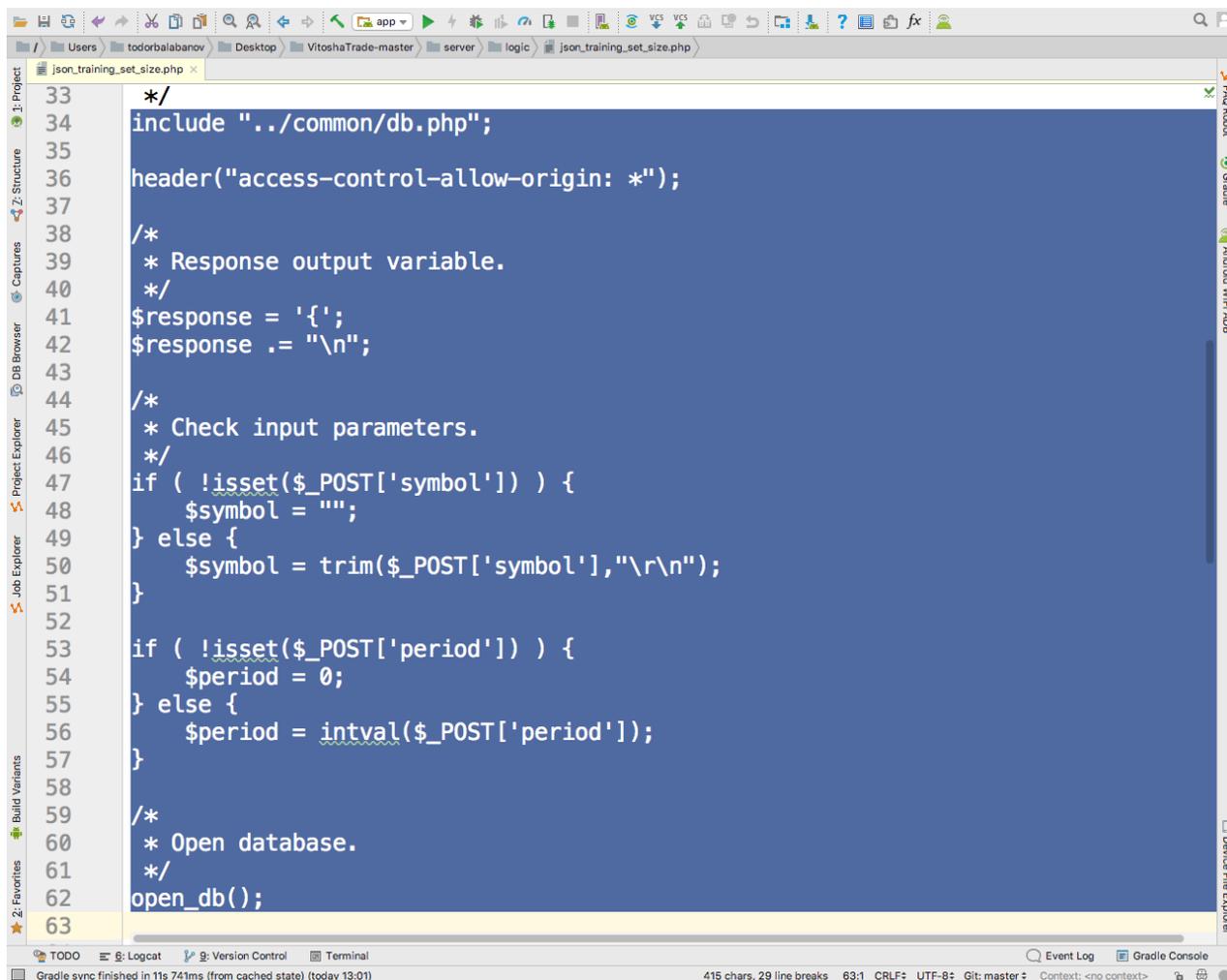
```

Фигура 5.45: Изпращане на резултата

Скриптът завършва със затваряне на връзката към базата данни, оформяне и изпращане на резултата до клиента (Фиг. 5.45).

### 5.3.7 Зареждане на брой тренировъчни примери по информация за валутна двойка

За определянето на броя тренировъчни примери се подхожда по аналогичен начин с включване на модула за работа с базата данни, проверка на входните аргументи и отваряне на връзка към базата данни (Фиг. 5.46).



```
33 */
34 include "../common/db.php";
35
36 header("access-control-allow-origin: *");
37
38 /*
39  * Response output variable.
40  */
41 $response = '{}';
42 $response .= "\n";
43
44 /*
45  * Check input parameters.
46  */
47 if ( !isset($_POST['symbol']) ) {
48     $symbol = "";
49 } else {
50     $symbol = trim($_POST['symbol'], "\r\n");
51 }
52
53 if ( !isset($_POST['period']) ) {
54     $period = 0;
55 } else {
56     $period = intval($_POST['period']);
57 }
58
59 /*
60  * Open database.
61  */
62 open_db();
63
```

Фигура 5.46: Брой тренировъчни примери по валутна информация

Изпълнението на заявката може да открие подходящо тренировъчно множество или да не открие такова (Фиг. 5.47).

```

63
64
65 /*
66  * Prepare SQL query.
67  */
68 $sql = "select number_of_examples from training_set, currency_pairs,
69         time_periods where training_set.currency_pairs_id=currency_pairs.id and
70         currency_pairs.period_id=time_periods.id and currency_pairs.symbol=" .
71         $symbol . "' and time_periods.minutes=" . $period . " ";
72
73 /*
74  * Run SQL query.
75  */
76 $result = query_db( $sql );
77
78 /*
79  * Check SQL query result.
80  */
81 if ($result != false) {
82     /*
83      * Response with number of training examples.
84      */
85     $response .= "numberOfExamples:" . trim($result[0][0], "\r\n") . "'";
86     $response .= "\n";
87 } else {
88     /*
89      * Response with zero available training examples.
90      */
91     $response .= "numberOfExamples:" . "0" . "'";
92     $response .= "\n";
93 }

```

Фигура 5.47: Заявка за брой тренировъчни примери

Скриптът приключва със затваряне на връзката към базата данни, пакетизиране и изпращане на отговора до клиента (Фиг. 5.48).

```

82      * response with number of training examples.
83      */
84      $response .= "numberOfExamples:" . trim($result[0][0], "\r\n") . "'";
85      $response .= "\n";
86  } else {
87      /*
88      * Response with zero available training examples.
89      */
90      $response .= "numberOfExamples:" . "0";
91      $response .= "\n";
92  }
93
94  $response .= "}";
95
96  /*
97  * Close database.
98  */
99  close_db();
100
101  /*
102  * Output response.
103  */
104  echo( $response );
105  flush();
106  ?>
107

```

Фигура 5.48: Изпращане на отговор до клиента

### 5.3.8 Съхраняване на екземпляр изкуствена невронна мрежа

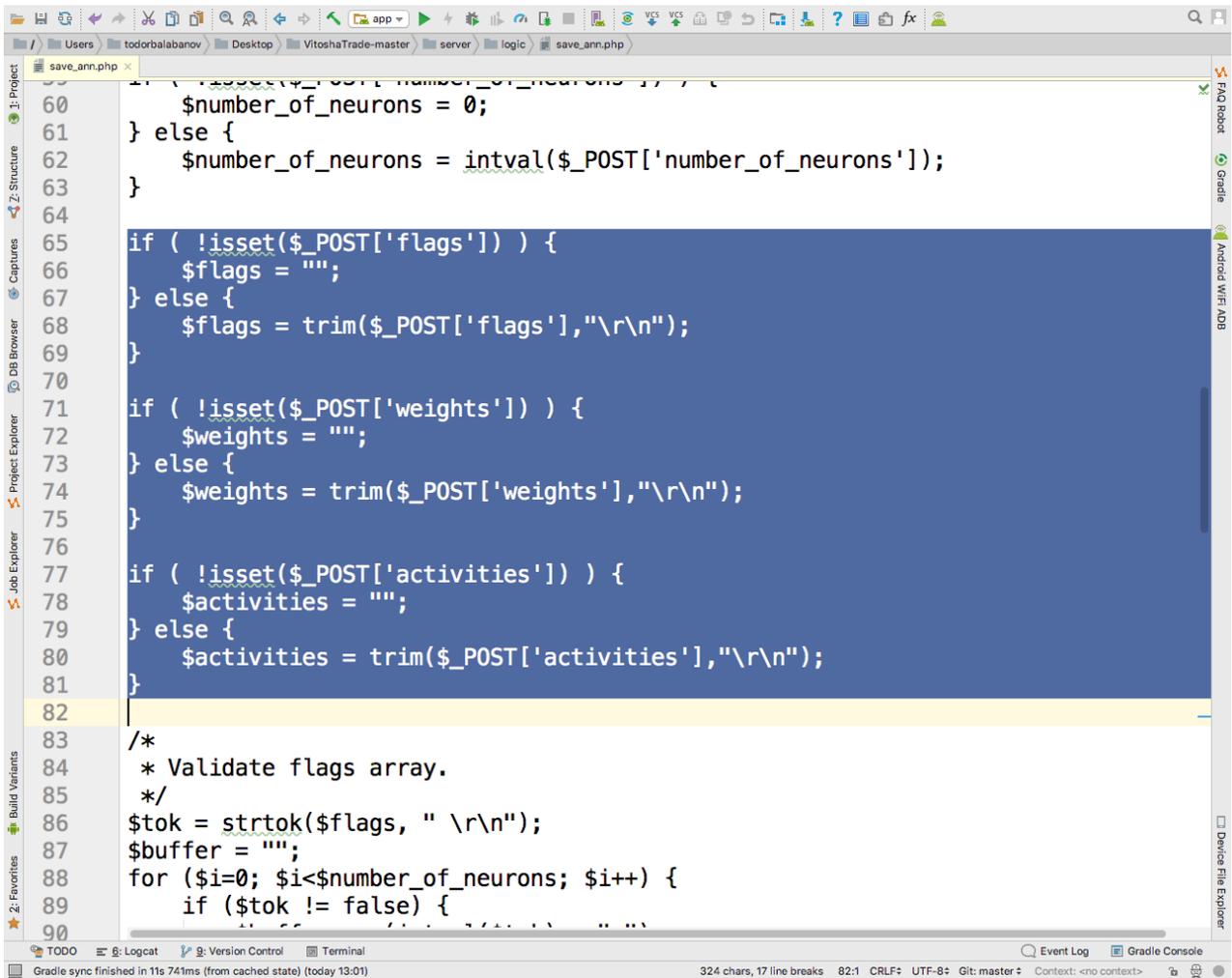
Съхраняването на екземпляр от изкуствената невронна мрежа не връща резултат за изпълнението на операцията и поради тази причина не се използва JSON. След включването на модула за работа с базата от данни се изисква по-сложна проверка на входните аргументи, тъй като те са повече на брой и по-сложни (Фиг. 5.49-5.53).

```

34 include "../common/db.php";
35
36 header("access-control-allow-origin: *");
37
38 /*
39  * Check input parameters.
40  */
41 if ( !isset($_POST['symbol']) ) {
42     $symbol = "";
43 } else {
44     $symbol = trim($_POST['symbol'], "\r\n");
45 }
46
47 if ( !isset($_POST['period']) ) {
48     $period = 0;
49 } else {
50     $period = intval($_POST['period']);
51 }
52
53 if ( !isset($_POST['fitness']) ) {
54     $fitness = PHP_INT_MAX;
55 } else {
56     $fitness = floatval($_POST['fitness']);
57 }
58
59 if ( !isset($_POST['number_of_neurons']) ) {
60     $number_of_neurons = 0;
61 } else {
62     $number_of_neurons = intval($_POST['number_of_neurons']);
63 }
64

```

Фигура 5.49: Проверка на входните аргументи за валутна двойка, период, жизненост и брой неврони



```
60     $number_of_neurons = 0;
61 } else {
62     $number_of_neurons = intval($_POST['number_of_neurons']);
63 }
64
65 if ( !isset($_POST['flags']) ) {
66     $flags = "";
67 } else {
68     $flags = trim($_POST['flags'], "\r\n");
69 }
70
71 if ( !isset($_POST['weights']) ) {
72     $weights = "";
73 } else {
74     $weights = trim($_POST['weights'], "\r\n");
75 }
76
77 if ( !isset($_POST['activities']) ) {
78     $activities = "";
79 } else {
80     $activities = trim($_POST['activities'], "\r\n");
81 }
82
83 /*
84  * Validate flags array.
85  */
86 $tok = strtok($flags, " \r\n");
87 $buffer = "";
88 for ($i=0; $i<$number_of_neurons; $i++) {
89     if ($tok != false) {
90
```

Фигура 5.50: Проверка на входните аргументи за флагове на невроните, връзки и тегла между невроните

```

78     $activities = "";
79 } else {
80     $activities = trim($_POST['activities'], "\r\n");
81 }
82
83 /*
84  * Validate flags array.
85  */
86 $tok = strtok($flags, " \r\n");
87 $buffer = "";
88 for ($i=0; $i<$number_of_neurons; $i++) {
89     if ($tok != false) {
90         $buffer .= (intval($tok) . " ");
91     } else {
92         $buffer .= "0" . " ";
93     }
94     $tok = strtok(" \r\n");
95 }
96 $buffer = trim($buffer, " \r\n");
97 $flags = $buffer;
98
99 /*
100  * Validate weights matrix.
101  */
102 //TODO Be very careful with matrix transpose (C++ vs PHP).
103 $tok = strtok($weights, " \r\n");
104 $buffer = "";
105 for ($j=0; $j<$number_of_neurons; $j++) {
106     for ($i=0; $i<$number_of_neurons; $i++) {
107         if ($tok != false) {
108             $buffer .= (floatval($tok) . " ");
109         }
110     }
111     $tok = strtok(" \r\n");
112 }
113 $buffer = trim($buffer, " \r\n");
114 $weights = $buffer;
115 }
116 }
117 }
118 }
119 }
120 }
121 }
122 }
123 }
124 }
125 }
126 }
127 }
128 }
129 }
130 }
131 }
132 }
133 }
134 }
135 }
136 }
137 }
138 }
139 }
140 }
141 }
142 }
143 }
144 }
145 }
146 }
147 }
148 }
149 }
150 }
151 }
152 }
153 }
154 }
155 }
156 }
157 }
158 }
159 }
160 }
161 }
162 }
163 }
164 }
165 }
166 }
167 }
168 }
169 }
170 }
171 }
172 }
173 }
174 }
175 }
176 }
177 }
178 }
179 }
180 }
181 }
182 }
183 }
184 }
185 }
186 }
187 }
188 }
189 }
190 }
191 }
192 }
193 }
194 }
195 }
196 }
197 }
198 }
199 }
200 }
201 }
202 }
203 }
204 }
205 }
206 }
207 }
208 }
209 }
210 }
211 }
212 }
213 }
214 }
215 }
216 }
217 }
218 }
219 }
220 }
221 }
222 }
223 }
224 }
225 }
226 }
227 }
228 }
229 }
230 }
231 }
232 }
233 }
234 }
235 }
236 }
237 }
238 }
239 }
240 }
241 }
242 }
243 }
244 }
245 }
246 }
247 }
248 }
249 }
250 }
251 }
252 }
253 }
254 }
255 }
256 }
257 }
258 }
259 }
260 }
261 }
262 }
263 }
264 }
265 }
266 }
267 }
268 }
269 }
270 }
271 }
272 }
273 }
274 }
275 }
276 }
277 }
278 }
279 }
280 }
281 }
282 }
283 }
284 }
285 }
286 }
287 }
288 }
289 }
290 }
291 }
292 }
293 }
294 }
295 }
296 }
297 }
298 }
299 }
300 }
301 }
302 }
303 }
304 }
305 }
306 }
307 }
308 }
309 }
310 }
311 }
312 }
313 }
314 }
315 }
316 }
317 }
318 }
319 }
320 }
321 }
322 }
323 }
324 }
325 }
326 }
327 }
328 }
329 }
330 }
331 }
332 }
333 }
334 }
335 }
336 }
337 }
338 }
339 }
340 }
341 }
342 }
343 }
344 }
345 }
346 }
347 }
348 }
349 }
350 }
351 }
352 }
353 }
354 }
355 }
356 }
357 }
358 }
359 }
360 }
361 }
362 }
363 }
364 }
365 }
366 }
367 }
368 }
369 }
370 }
371 }
372 }
373 }
374 }
375 }
376 }
377 }
378 }
379 }
380 }
381 }
382 }
383 }
384 }
385 }
386 }
387 }
388 }
389 }
390 }
391 }
392 }
393 }
394 }
395 }
396 }
397 }
398 }
399 }
400 }
401 }
402 }
403 }
404 }
405 }
406 }
407 }
408 }
409 }
410 }
411 }
412 }
413 }
414 }
415 }
416 }
417 }
418 }
419 }
420 }
421 }
422 }
423 }
424 }
425 }
426 }
427 }
428 }
429 }
430 }
431 }
432 }
433 }
434 }
435 }
436 }
437 }
438 }
439 }
440 }
441 }
442 }
443 }
444 }
445 }
446 }
447 }
448 }
449 }
450 }
451 }
452 }
453 }
454 }
455 }
456 }
457 }
458 }
459 }
460 }
461 }
462 }
463 }
464 }
465 }
466 }
467 }
468 }
469 }
470 }
471 }
472 }
473 }
474 }
475 }
476 }
477 }
478 }
479 }
480 }
481 }
482 }
483 }
484 }
485 }
486 }
487 }
488 }
489 }
490 }
491 }
492 }
493 }
494 }
495 }
496 }
497 }
498 }
499 }
500 }
501 }
502 }
503 }
504 }
505 }
506 }
507 }
508 }
509 }
510 }
511 }
512 }
513 }
514 }
515 }
516 }
517 }
518 }
519 }
520 }
521 }
522 }
523 }
524 }
525 }
526 }
527 }
528 }
529 }
530 }
531 }
532 }
533 }
534 }
535 }
536 }
537 }
538 }
539 }
540 }
541 }
542 }
543 }
544 }
545 }
546 }
547 }
548 }
549 }
550 }
551 }
552 }
553 }
554 }
555 }
556 }
557 }
558 }
559 }
560 }
561 }
562 }
563 }
564 }
565 }
566 }
567 }
568 }
569 }
570 }
571 }
572 }
573 }
574 }
575 }
576 }
577 }
578 }
579 }
580 }
581 }
582 }
583 }
584 }
585 }
586 }
587 }
588 }
589 }
590 }
591 }
592 }
593 }
594 }
595 }
596 }
597 }
598 }
599 }
600 }
601 }
602 }
603 }
604 }
605 }
606 }
607 }
608 }
609 }
610 }
611 }
612 }
613 }
614 }
615 }
616 }
617 }
618 }
619 }
620 }
621 }
622 }
623 }
624 }
625 }
626 }
627 }
628 }
629 }
630 }
631 }
632 }
633 }
634 }
635 }
636 }
637 }
638 }
639 }
640 }
641 }
642 }
643 }
644 }
645 }
646 }
647 }
648 }
649 }
650 }
651 }
652 }
653 }
654 }
655 }
656 }
657 }
658 }
659 }
660 }
661 }
662 }
663 }
664 }
665 }
666 }
667 }
668 }
669 }
670 }
671 }
672 }
673 }
674 }
675 }
676 }
677 }
678 }
679 }
680 }
681 }
682 }
683 }
684 }
685 }
686 }
687 }
688 }
689 }
690 }
691 }
692 }
693 }
694 }
695 }
696 }
697 }
698 }
699 }
700 }
701 }
702 }
703 }
704 }
705 }
706 }
707 }
708 }
709 }
710 }
711 }
712 }
713 }
714 }
715 }
716 }
717 }
718 }
719 }
720 }
721 }
722 }
723 }
724 }
725 }
726 }
727 }
728 }
729 }
730 }
731 }
732 }
733 }
734 }
735 }
736 }
737 }
738 }
739 }
740 }
741 }
742 }
743 }
744 }
745 }
746 }
747 }
748 }
749 }
750 }
751 }
752 }
753 }
754 }
755 }
756 }
757 }
758 }
759 }
760 }
761 }
762 }
763 }
764 }
765 }
766 }
767 }
768 }
769 }
770 }
771 }
772 }
773 }
774 }
775 }
776 }
777 }
778 }
779 }
780 }
781 }
782 }
783 }
784 }
785 }
786 }
787 }
788 }
789 }
790 }
791 }
792 }
793 }
794 }
795 }
796 }
797 }
798 }
799 }
800 }
801 }
802 }
803 }
804 }
805 }
806 }
807 }
808 }
809 }
810 }
811 }
812 }
813 }
814 }
815 }
816 }
817 }
818 }
819 }
820 }
821 }
822 }
823 }
824 }
825 }
826 }
827 }
828 }
829 }
830 }
831 }
832 }
833 }
834 }
835 }
836 }
837 }
838 }
839 }
840 }
841 }
842 }
843 }
844 }
845 }
846 }
847 }
848 }
849 }
850 }
851 }
852 }
853 }
854 }
855 }
856 }
857 }
858 }
859 }
860 }
861 }
862 }
863 }
864 }
865 }
866 }
867 }
868 }
869 }
870 }
871 }
872 }
873 }
874 }
875 }
876 }
877 }
878 }
879 }
880 }
881 }
882 }
883 }
884 }
885 }
886 }
887 }
888 }
889 }
890 }
891 }
892 }
893 }
894 }
895 }
896 }
897 }
898 }
899 }
900 }
901 }
902 }
903 }
904 }
905 }
906 }
907 }
908 }
909 }
910 }
911 }
912 }
913 }
914 }
915 }
916 }
917 }
918 }
919 }
920 }
921 }
922 }
923 }
924 }
925 }
926 }
927 }
928 }
929 }
930 }
931 }
932 }
933 }
934 }
935 }
936 }
937 }
938 }
939 }
940 }
941 }
942 }
943 }
944 }
945 }
946 }
947 }
948 }
949 }
950 }
951 }
952 }
953 }
954 }
955 }
956 }
957 }
958 }
959 }
960 }
961 }
962 }
963 }
964 }
965 }
966 }
967 }
968 }
969 }
970 }
971 }
972 }
973 }
974 }
975 }
976 }
977 }
978 }
979 }
980 }
981 }
982 }
983 }
984 }
985 }
986 }
987 }
988 }
989 }
990 }
991 }
992 }
993 }
994 }
995 }
996 }
997 }
998 }
999 }
1000 }

```

Фигура 5.51: Проверка за типа на всеки неврон

```

96 $buffer = trim($buffer, " \r\n");
97 $flags = $buffer;
98
99 /*
100  * Validate weights matrix.
101  */
102 //TODO Be very careful with matrix transpose (C++ vs PHP).
103 $tok = strtok($weights, " \r\n");
104 $buffer = "";
105 for ($j=0; $j<$number_of_neurons; $j++) {
106     for ($i=0; $i<$number_of_neurons; $i++) {
107         if ($tok != false) {
108             $buffer .= (floatval($tok) . " ");
109         } else {
110             $buffer .= "0" . " ";
111         }
112     }
113     $tok = strtok(" \r\n");
114 }
115
116 $buffer = trim($buffer, " \r\n");
117 $buffer .= "\r\n";
118 }
119 $buffer = trim($buffer, " \r\n");
120 $weights = $buffer;
121
122 /*
123  * Validate activities matrix.
124  */
125 $tok = strtok($activities, " \r\n");
126 $buffer = "";

```

Фигура 5.52: Проверка за стойността на всяко тегло

```

115
116     $buffer = trim($buffer, " \r\n");
117     $buffer .= "\r\n";
118 }
119 $buffer = trim($buffer, " \r\n");
120 $weights = $buffer;
121
122 /*
123  * Validate activities matrix.
124  */
125 $tok = strtok($activities, " \r\n");
126 $buffer = "";
127 //TODO Be very careful with matrix transpose (C++ vs PHP).
128 for ($j=0; $j<$number_of_neurons; $j++) {
129     for ($i=0; $i<$number_of_neurons; $i++) {
130         if ($tok != false) {
131             $buffer .= (floatval($tok) . " ");
132         } else {
133             $buffer .= "0" . " ";
134         }
135
136         $tok = strtok(" \r\n");
137     }
138
139     $buffer = trim($buffer, " \r\n");
140     $buffer .= "\r\n";
141 }
142 $buffer = trim($buffer, " \r\n");
143 $activities = $buffer;
144
145 /*
146

```

Фигура 5.53: Проверка за стойността на всяка връзка

При този скрипт проверките са значително по-сложни, тъй като освен наличност на аргументите се следи за числените стойности, които пристигат от клиентското приложение.

```

139     $buffer = trim($buffer, " \r\n");
140     $buffer .= "\r\n";
141 }
142 $buffer = trim($buffer, " \r\n");
143 $activities = $buffer;
144
145 /*
146  * Open database.
147  */
148 open_db();
149
150 /*
151  * SQL query for ANN save.
152  */
153 $sql = "call saveAnn('" . $symbol . "', " . $period . "', " . $number_of_neurons .
154     ", '" . $flags . "', '" . $activities . "', " . $fitness . "', '" .
155     $weights . "')";
156
157 $result = query_db( $sql );
158
159 /*
160  * Close database.
161  */
162 close_db();
163 ?>
164

```

Фигура 5.54: Съхраняване на екземпляр изкуствена невронна мрежа

Същината на скрипта представлява отваряне на връзка към базата данни, изпълнение на съхранена процедура и затваряне на връзката към базата данни (Фиг. 5.54).

### 5.3.9 Съхраняване на екземпляр изкуствена невронна мрежа след дообучение

Когато определена изкуствена невронна мрежа е преминала допълнителен цикъл на обучение и бъде изпратена до отдалечения сървър, разумно е информацията да се съхранява само, ако води до подобряване на жизнеността.

```

138 }
139 $buffer = trim($buffer, " \r\n");
140 $activities = $buffer;
141
142 /*
143  * Open database.
144  */
145 open_db();
146
147 /*
148  * Load best known fitness.
149  */
150 $sql = "select loadBestFitness('" . $symbol . "', " . $period . ", " .
151 $number_of_neurons . ", '" . $flags . "', '" . $activities . "')";
152 $result = query_db( $sql );
153
154 /*
155  * Store the information only if the fitness is better.
156  */
157 if($result[0][0] > $fitness) {
158     /*
159     * SQL query for ANN save.
160     */
161     $sql = "call saveAnn('" . $symbol . "', " . $period . ", " . $number_of_neurons .
162     $result = query_db( $sql );
163 }
164
165
166 /*
167  * Close database.
168  */
169
170

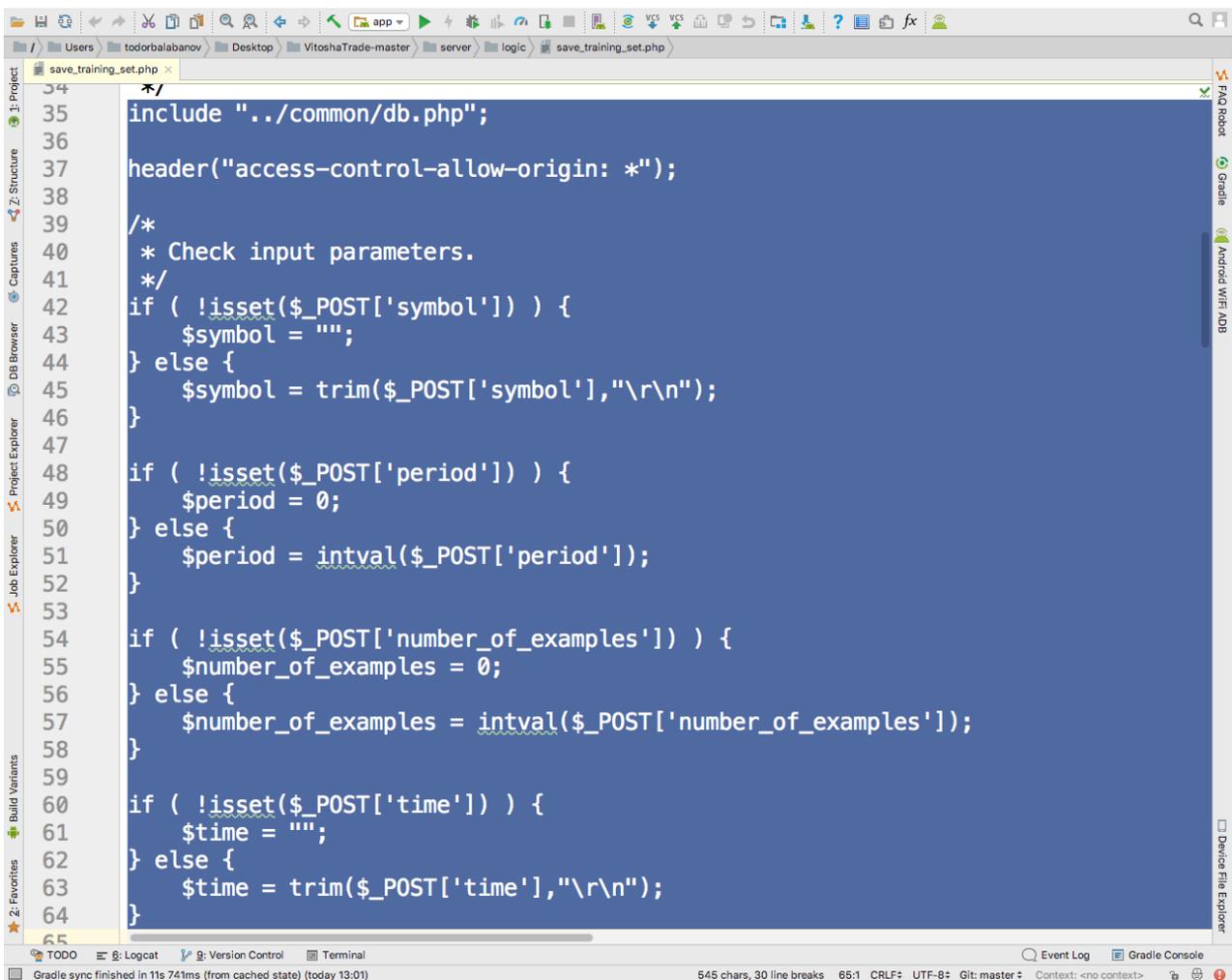
```

Фигура 5.55: Проверка на жизнеността преди съхраняване на информацията

Разликата между общата процедура за съхраняване на изкуствена невронна мрежа и съхраняването след допълнително обучение е в проверката на най-добрата постигната жизнена стойност (Фиг. 5.55).

### 5.3.10 Съхраняване на тренировъчно множество

По аналогичен начин, при съхраняването на тренировъчни примери не се връща JSON съобщение и кодът по проверката на входните данни превъзхожда кода за самото съхраняване. Първоначално се проверява наличието на аргументите (Фиг. 5.56,5.57).



```
35 include "../common/db.php";
36
37 header("access-control-allow-origin: *");
38
39 /*
40  * Check input parameters.
41  */
42 if ( !isset($_POST['symbol']) ) {
43     $symbol = "";
44 } else {
45     $symbol = trim($_POST['symbol'], "\r\n");
46 }
47
48 if ( !isset($_POST['period']) ) {
49     $period = 0;
50 } else {
51     $period = intval($_POST['period']);
52 }
53
54 if ( !isset($_POST['number_of_examples']) ) {
55     $number_of_examples = 0;
56 } else {
57     $number_of_examples = intval($_POST['number_of_examples']);
58 }
59
60 if ( !isset($_POST['time']) ) {
61     $time = "";
62 } else {
63     $time = trim($_POST['time'], "\r\n");
64 }
65
```

Фигура 5.56: Проверка на аргументите за валутна двойка, период, брой примери и времеви маркери

```

65
66 if ( !isset($_POST['open']) ) {
67     $open = "";
68 } else {
69     $open = trim($_POST['open'], "\r\n");
70 }
71
72 if ( !isset($_POST['low']) ) {
73     $low = "";
74 } else {
75     $low = trim($_POST['low'], "\r\n");
76 }
77
78 if ( !isset($_POST['high']) ) {
79     $high = "";
80 } else {
81     $high = trim($_POST['high'], "\r\n");
82 }
83
84 if ( !isset($_POST['close']) ) {
85     $close = "";
86 } else {
87     $close = trim($_POST['close'], "\r\n");
88 }
89
90 if ( !isset($_POST['volume']) ) {
91     $volume = "";
92 } else {
93     $volume = trim($_POST['volume'], "\r\n");
94 }
95

```

Фигура 5.57: Проверка на аргументите за отваряне, най-ниска стойност, най-висока стойност, затваряне и изтъргуван обем

Всеки от масивите преминава допълнителна проверка за стойностите, които съдържа (Фиг. 5.58-5.60).

```

98  */
99  $tok = strtok($time, " \r\n");
100  $buffer = "";
101  for ($i=0; $i<$number_of_examples; $i++) {
102    if ($tok != false) {
103      $buffer .= (intval($tok) . " ");
104    } else {
105      $buffer .= "0" . " ";
106    }
107    $tok = strtok(" \r\n");
108  }
109  $buffer = trim($buffer, " \r\n");
110  $time = $buffer;
111
112  /*
113   * Validate open array.
114   */
115  $tok = strtok($open, " \r\n");
116  $buffer = "";
117  for ($i=0; $i<$number_of_examples; $i++) {
118    if ($tok != false) {
119      $buffer .= (floatval($tok) . " ");
120    } else {
121      $buffer .= "0" . " ";
122    }
123    $tok = strtok(" \r\n");
124  }
125  $buffer = trim($buffer, " \r\n");
126  $open = $buffer;
127
128  /*
  
```

Фигура 5.58: Проверка на стойностите за времеви маркери и нива на отваряне

```

130  */
131  $tok = strtok($low, " \r\n");
132  $buffer = "";
133  for ($i=0; $i<$number_of_examples; $i++) {
134      if ($tok != false) {
135          $buffer .= (floatval($tok) . " ");
136      } else {
137          $buffer .= "0" . " ";
138      }
139      $tok = strtok(" \r\n");
140  }
141  $buffer = trim($buffer, " \r\n");
142  $low = $buffer;
143
144  /*
145   * Validate high array.
146   */
147  $tok = strtok($high, " \r\n");
148  $buffer = "";
149  for ($i=0; $i<$number_of_examples; $i++) {
150      if ($tok != false) {
151          $buffer .= (floatval($tok) . " ");
152      } else {
153          $buffer .= "0" . " ";
154      }
155      $tok = strtok(" \r\n");
156  }
157  $buffer = trim($buffer, " \r\n");
158  $high = $buffer;
159
160  /*

```

Фигура 5.59: Проверка на стойностите за най-ниско и най-високо постигнато ниво

```

162  */
163  $tok = strtok($close, " \r\n");
164  $buffer = "";
165  for ($i=0; $i<$number_of_examples; $i++) {
166      if ($tok != false) {
167          $buffer .= (floatval($tok) . " ");
168      } else {
169          $buffer .= "0" . " ";
170      }
171      $tok = strtok(" \r\n");
172  }
173  $buffer = trim($buffer, " \r\n");
174  $close = $buffer;
175
176  /*
177  * Validate volume array.
178  */
179  $tok = strtok($volume, " \r\n");
180  $buffer = "";
181  for ($i=0; $i<$number_of_examples; $i++) {
182      if ($tok != false) {
183          $buffer .= (floatval($tok) . " ");
184      } else {
185          $buffer .= "0" . " ";
186      }
187      $tok = strtok(" \r\n");
188  }
189  $buffer = trim($buffer, " \r\n");
190  $volume = $buffer;
191
192

```

Фигура 5.60: Проверка на стойностите за затваряне и изтъргуван обем

Същинската част на скрипта отваря връзка към базата данни, изпълнява съхранена процедура и затваря връзката към базата данни (Фиг. 5.61).

```

185     $buffer .= "\n";
186 }
187 $tok = strtok(" \r\n");
188 }
189 $buffer = trim($buffer, " \r\n");
190 $volume = $buffer;
191
192 /*
193  * Open database.
194  */
195 open_db();
196
197 /*
198  * Run SQL query to replace previous record or insert new one.
199  */
200 $sql = "call saveTrainingSet('" . $symbol . "', $period, $number_of_examples,
201     '" . $time . "', '" . $open . "', '" . $low . "', '" . $high . "',
202     '" . $close . "', '" . $volume . "')";
203 query_db( $sql );
204
205 /*
206  * Close database.
207  */
208 close_db();
209 ?>
210

```

Фигура 5.61: Съхраняване на тренировъчно множество

## Глава 6

# Комуникация клиент-сървър

Комуникацията между Android клиента и PHP базирания сървър се извършва през HTTP с размяна на JSON пакетирани съобщения.

### 6.1 Hypertext Transfer Protocol

При всяко създаване на инстанция за тапета е нужно да се извърши свързване към отдалечения сървър и да бъде заявен пакет за работа. Ако връзката не е възможна се използват данните, записани в локалната база данни.

```

322
323
324     /*
325      * Return minimum and maximum values of the activation function output.
326      */
327     return new double[]{check[0], check[check.length - 1]};
328 }
329
330 /**
331  * {@inheritDoc}
332  */
333 @Override
334 public void onCreate() {
335     super.onCreate();
336
337     initialize();
338 }
339
340 /**
341  * {@inheritDoc}
342  */
343 @Override
344 public Engine onCreateEngine() { return new WallpaperEngine(); }
345
346
347 /**
348  * Wallpaper engine class.
349  *
350  * @author Todor Balabanov
351  */
352 private class WallpaperEngine extends Engine {
353

```

Фигура 6.1: Инициализация на вътрешните променливи

Инициализацията на вътрешните променливи се осъществява в събитието onCreate (Фиг. 6.1).

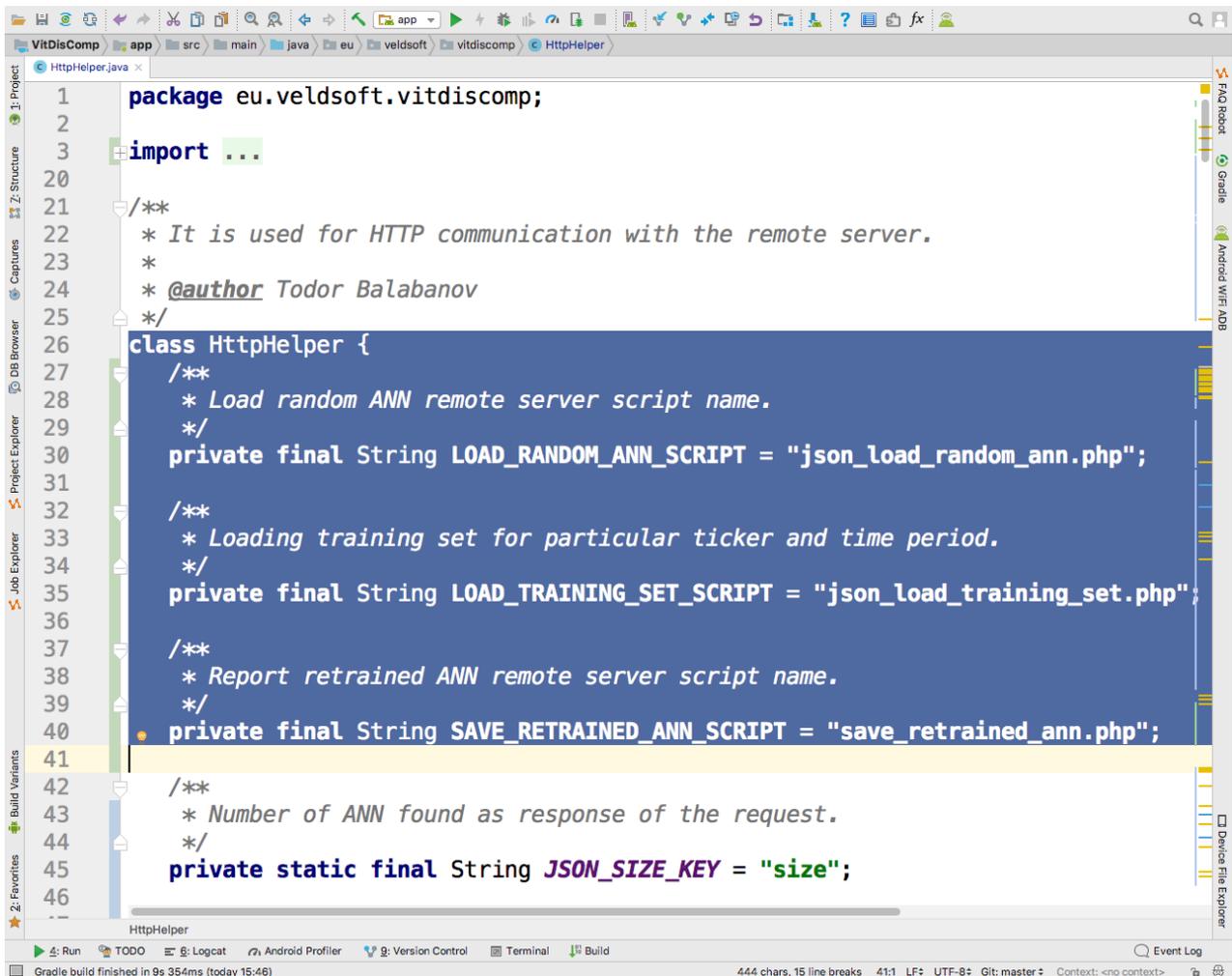
```

148
149
150     /**
151      * Training rule object.
152      */
153
154     private static ResilientPropagation train = null;
155
156     /**
157      * Initialize common class members.
158      */
159     private void initialize() {
160         /**
161          * Load ANN structure and time series data from the remote server.
162          */
163         HttpHelper helper = new HttpHelper(PreferenceManager
164             .getDefaultSharedPreferences(
165                 context: VitoshaTradeWallpaperService.this).
166                 getString(s: "server_url", s1: "localhost"));
167
168         if(helper.load() == false) {
169             // TODO Use local data if the remote server is not available.
170         }
171
172         Map<NeuronType, Integer> counters = new HashMap<>();
173         counters.put(NeuronType.REGULAR, v: 0);
174         counters.put(NeuronType.BIAS, v: 0);
175         counters.put(NeuronType.INPUT, v: 0);
176         counters.put(NeuronType.OUTPUT, v: 0);
177
178         for (int type : InputData.NEURONS) {
179             counters.put(NeuronType.valueOf(type),

```

Фигура 6.2: Зареждане на данните с помощта на HTTP комуникация

Инициализирането на вътрешните променливи през HTTP се поема от допълнително създаден помощен обект, който е от клас отговорен за комуникацията (Фиг. 6.2).



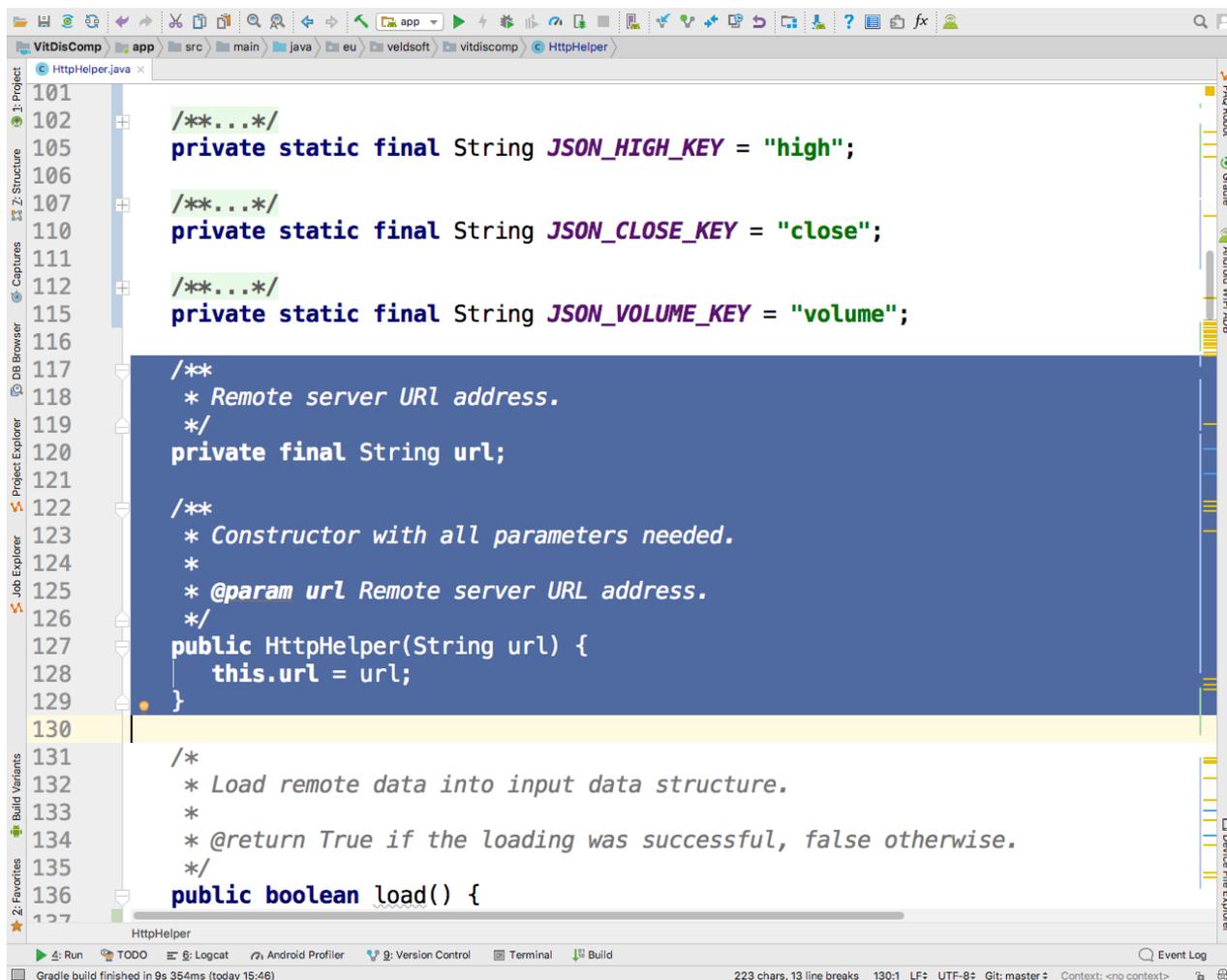
```

1 package eu.veldsoft.vitdiscomp;
2
3 import ...
4
20
21 /**
22  * It is used for HTTP communication with the remote server.
23  *
24  * @author Todor Balabanov
25  */
26 class HttpHelper {
27     /**
28     * Load random ANN remote server script name.
29     */
30     private final String LOAD_RANDOM_ANN_SCRIPT = "json_load_random_ann.php";
31
32     /**
33     * Loading training set for particular ticker and time period.
34     */
35     private final String LOAD_TRAINING_SET_SCRIPT = "json_load_training_set.php";
36
37     /**
38     * Report retrained ANN remote server script name.
39     */
40     private final String SAVE_RETRAINED_ANN_SCRIPT = "save_retrained_ann.php";
41
42     /**
43     * Number of ANN found as response of the request.
44     */
45     private static final String JSON_SIZE_KEY = "size";
46

```

Фигура 6.3: HTTP комуникационен файл

Комуникацията по HTTP с отдалечения сървър и синтактичната обработка на JSON съобщенията са изнесени в помощен клас HttpHelper (Фиг. 6.3). Съществуват различни възможности за задаване на имената, които се ползват за отдалечените скриптове, но един от най-елементарните е чрез именуванни константи. Имената на сървърните скриптове подлежат на промяна, но се разчита, че тя ще е значително по-рядко от смяната на URL адреса. Точно поради тази причина параметризирането на информацията е по различен начин.



Фигура 6.4: Конструктор на спомагателния клас

Тъй като URL адресът на отдалечения сървър е възможно да търпи многократни промени, е разумно тази информация да постъпва в конструктора на класа (Фиг. 6.4).

```

130
131  /*
132   * Load remote data into input data structure.
133   *
134   * @return True if the loading was successful, false otherwise.
135   */
136  public boolean load() {
137      String symbol = InputData.SYMBOL;
138      int period = InputData.PERIOD;
139      int[] flags = InputData.NEURONS;
140      double[][] weights = InputData.WEIGHTS;
141      double[][] activities = InputData.ACTIVITIES;
142      long[] time = InputData.TIME;
143      double[] open = InputData.OPEN;
144      double[] low = InputData.LOW;
145      double[] high = InputData.HIGH;
146      double[] close = InputData.HIGH;
147      double[] volume = InputData.VOLUME;
148
149      HttpClient client = new DefaultHttpClient();
150      client.getParams().setParameter(s: "http.protocol.content-charset", o: "UTF-8")
151
152      /*
153       * Load randomly selected ANN.
154       */
155      HttpPost post = new HttpPost(uri: "http://" + url + "/" + LOAD_RANDOM_ANN_SCRIPT
156
157      try {
158          HttpResponse response = client.execute(post);
159

```

Фигура 6.5: Функция за зареждане на случайно избрана от сървъра мрежа

За зареждане на случайно избрана мрежа се извиква определеният за целта скрипт на отдалечения сървър. Това се извършва в спомагателна функция load, която зарежда информацията в публично достъпната глобална структура за представяне на работните данни (Фиг. 6.5).

```

210     return false;
211 } catch (JSONException exception) {
212     return false;
213 } catch (Exception exception) {
214     return false;
215 }
216
217 /*
218  * Load training set for the selected ANN.
219  */
220 post = new HttpPost(uri:"http://" + url + "/" + LOAD_TRAINING_SET_SCRIPT);
221 List<NameValuePair> pairs = new ArrayList<NameValuePair>();
222 pairs.add(new BasicNameValuePair(name:"symbol", symbol));
223 pairs.add(new BasicNameValuePair(name:"period", value:"" + period));
224 try {
225     post.setEntity(new UrlEncodedFormEntity(pairs));
226 } catch (UnsupportedEncodingException e) {
227     return false;
228 }
229
230 try {
231     HttpResponse response = client.execute(post);
232
233     JSONObject result = new JSONObject(EntityUtils.toString(response.getEntity()));
234
235     int size = result.getInt(JSON_NUMBER_OF_EXAMPLES_KEY);
236
237     /*
238      * If there is no ANN training set on the server side nothing can be loaded.
239      */

```

Фигура 6.6: Зареждане на тренировъчно множество

След като бъде заредена случайно избрана мрежа, се зарежда и подходящо за мрежата тренировъчно множество (Фиг. 6.6). За съпоставка на наличните времеви редове се използва названието на символната двойка и периодът на реда.

```

306 }
307
308 /**
309  * Store calculated ANN weights on the remote web server.
310  *
311  * @return True if the saving was successful, false otherwise.
312  */
313 public boolean store() {
314     HttpClient client = new DefaultHttpClient();
315     client.getParams().setParameter("http.protocol.content-charset", "UTF-8");
316
317     /**
318      * Store retrained ANN.
319      */
320     HttpPost post = new HttpPost("http://" + url + "/" + SAVE_RETRAINED_ANN_SC
321     List<NameValuePair> pairs = new ArrayList<NameValuePair>();
322
323     pairs.add(new BasicNameValuePair( name: "symbol", InputData.SYMBOL));
324     pairs.add(new BasicNameValuePair( name: "period", value: "" + InputData.PERIOD));
325     pairs.add(new BasicNameValuePair( name: "fitness", value: "" + InputData.FITNESS));
326     pairs.add(new BasicNameValuePair( name: "number_of_neurons", value: "" + InputData.NEURONS));
327
328     String flags = "";
329     for (int i=0; i<InputData.NEURONS.length; i++) {
330         flags += InputData.NEURONS[i] + " ";
331     }
332     pairs.add(new BasicNameValuePair( name: "flags", value: "" + flags.trim()));
333
334     //TODO Matrix transpose is possible.
335     String weights = "";
336 }
    
```

Фигура 6.7: Функция за запазване на допълнително обучавана мрежа

При процеса за съхраняване на допълнително обучената мрежа се използва втора помощна функция save, която има за цел да изпрати по HTTP заявка информацията налична в глобалната структура с данните от обучението (Фиг. 6.7).

```

323 pairs.add(new BasicNameValuePair(name: "symbol", InputData.SYMBOL));
324 pairs.add(new BasicNameValuePair(name: "period", value: "" + InputData.PERIOD));
325 pairs.add(new BasicNameValuePair(name: "fitness", value: "" + InputData.FITNESS));
326 pairs.add(new BasicNameValuePair(name: "number_of_neurons", value: "" + InputData.NUMBER_OF_NEURONS));
327
328 String flags = "";
329 for (int i=0; i<InputData.NEURONS.length; i++) {
330     flags += InputData.NEURONS[i] + " ";
331 }
332 pairs.add(new BasicNameValuePair(name: "flags", value: "" + flags.trim()));
333
334 //TODO Matrix transpose is possible.
335 String weights = "";
336 for (int j=0; j<InputData.NEURONS.length; j++) {
337     for (int i = 0; i < InputData.NEURONS.length; i++) {
338         weights += InputData.WEIGHTS[i][j] + " ";
339     }
340     weights = weights.trim() + "\r\n";
341 }
342 pairs.add(new BasicNameValuePair(name: "weights", value: "" + weights.trim()));
343
344 //TODO Matrix transpose is possible.
345 String activites = "";
346 for (int j=0; j<InputData.NEURONS.length; j++) {
347     for (int i = 0; i < InputData.NEURONS.length; i++) {
348         activites += InputData.ACTIVITIES[i][j] + " ";
349     }
350     activites = activites.trim() + "\r\n";
351 }
352 pairs.add(new BasicNameValuePair(name: "activites", value: "" + activites.trim()));

```

Фигура 6.8: Подготовка на POST заявка

Данните от глобалната структура се оформят като HTTP заявка от тип POST като това е свързано с трансформирането им от бинарни стойности в текстове (Фиг. 6.8).

```

344 //TODO Matrix transpose is possible.
345 String activites = "";
346 for (int j=0; j<InputData.NEURONS.length; j++) {
347     for (int i = 0; i < InputData.NEURONS.length; i++) {
348         activites += InputData.ACTIVITIES[i][j] + " ";
349     }
350     activites = activites.trim() + "\r\n";
351 }
352 pairs.add(new BasicNameValuePair( name: "activites", value: "" + activites.tr
353
354     try {
355         post.setEntity(new UrlEncodedFormEntity(pairs));
356         client.execute(post);
357     } catch (UnsupportedEncodingException exception) {
358         return false;
359     } catch (ClientProtocolException exception) {
360         return false;
361     } catch (IOException exception) {
362         return false;
363     }
364
365     return true;
366 }
367 }
368

```

HttpHelper -> store()

Run | TODO | Logcat | Android Profiler | Version Control | Terminal | Build

Gradle build finished in 9s 354ms (today 15:46) 277 chars, 10 line breaks 364:1 LF: UTF-8: Git: master Context: <no context>

Фигура 6.9: Изпълнение на POST заявката

Изпращането на информацията приключва с изпълнение на HTTP заявката (Фиг. 6.9).

## 6.2 JavaScript Object Notation

Веднъж получена по HTTP, информацията пакетирана с JSON, подлежи на синтактичен анализ и трансформиране от поток байтове във вътрешната структура за представяне на пакет за пресмятане.

```

39  */
40  private final String SAVE_RETRAINED_ANN_SCRIPT = "save_retrained_ann.php";
41
42  /**...*/
45  private static final String JSON_SIZE_KEY = "size";
46
47  /**...*/
50  private static final String JSON_SYMBOL_KEY = "symbol";
51
52  /**...*/
55  private static final String JSON_PERIOD_KEY = "period";
56
57  /**...*/
60  private static final String JSON_FITNESS_KEY = "fitness";
61
62  /**...*/
65  private static final String JSON_FLAGS_KEY = "flags";
66
67  /**...*/
70  private static final String JSON_WEIGHTS_KEY = "weights";
71
72  /**...*/
75  private static final String JSON_ACTIVITIES_KEY = "activities";
76
77  /**...*/
80  private final String JSON_NUMBER_OF_NEURONS_KEY = "numberOfNeurons";
81
82  /**...*/
85  private static final String JSON_NUMBER_OF_EXAMPLES_KEY = "numberOfExamples";
86

```

Фигура 6.10: Ключови стойности за описване на мрежата

Една от основните роли на JSON е в изграждането на комуникационни протоколи, която в настоящата разработка се състои в информация организирана йерархично на принципа „ключ-стойност“. Тъй като комуникационните протоколи подлежат на промяна, е удачно ключовите стойности да се представят под формата на преименувани константи (Фиг. 6.10,6.11).

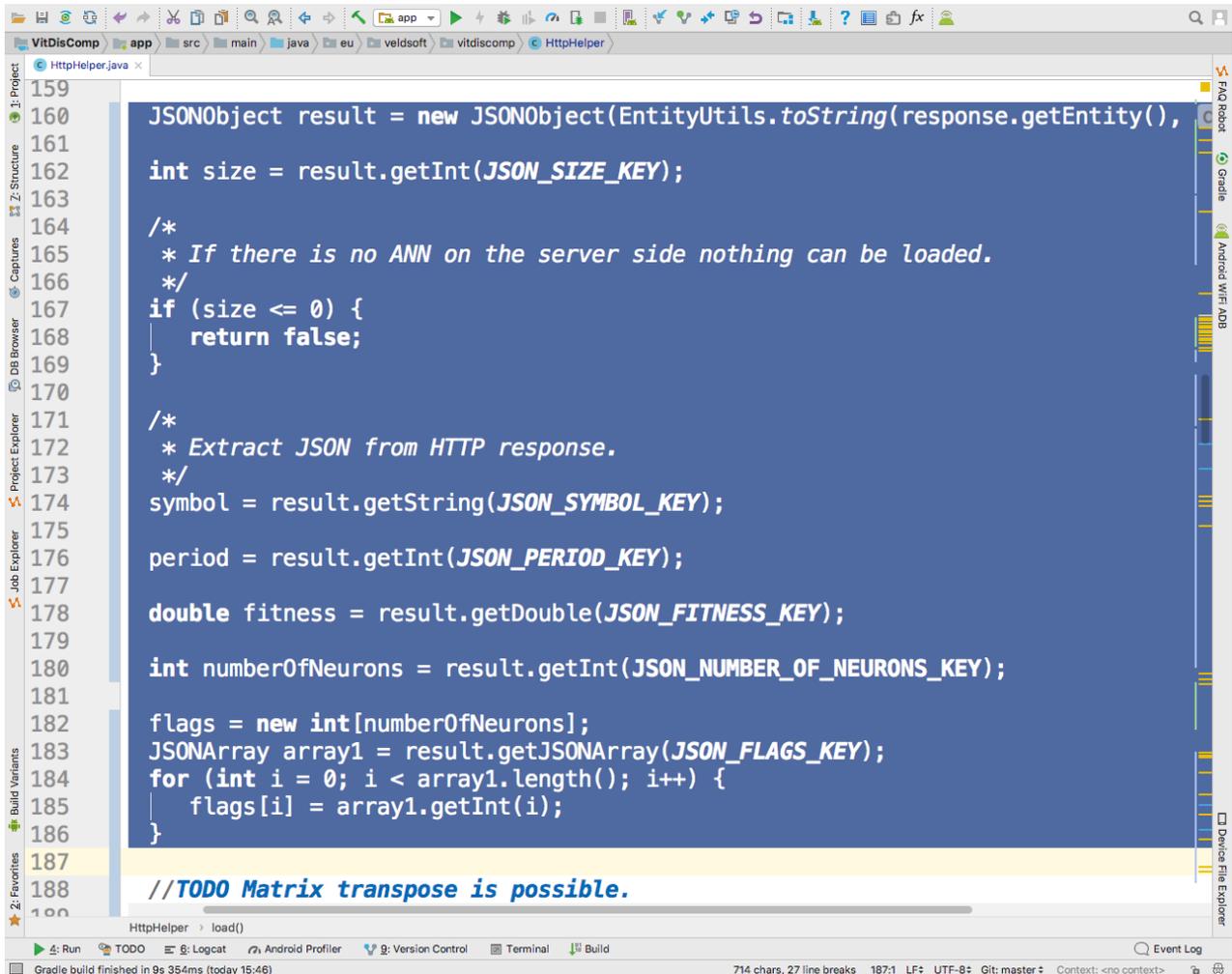
```

77  /**...*/
80  private final String JSON_NUMBER_OF_NEURONS_KEY = "numberOfNeurons";
81
82  /**...*/
85  private static final String JSON_NUMBER_OF_EXAMPLES_KEY = "numberOfExamples";
86
87  /**...*/
90  private static final String JSON_TIME_KEY = "time";
91
92  /**...*/
95  private static final String JSON_OPEN_KEY = "open";
96
97  /**...*/
100 private static final String JSON_LOW_KEY = "low";
101
102 /**...*/
105 private static final String JSON_HIGH_KEY = "high";
106
107 /**...*/
110 private static final String JSON_CLOSE_KEY = "close";
111
112 /**...*/
115 private static final String JSON_VOLUME_KEY = "volume";
116
117 /**
118  * Remote server URL address.
119  */
120 private final String url;
121
122 /**
123

```

Фигура 6.11: Ключови стойности за описване на тренировъчното множество

При зареждането на изкуствена невронна мрежа и времеви ред за обучението ѝ от отдалечения сървър, информацията се получава с две отделни HTTP заявки. Това налага два отделни фрагмента да обработят JSON съобщенията.



Фигура 6.12: Стойности за времевия ред и невроните

При първия фрагмент се получава информация за вида на времевия ред, периода на времевия ред, броя неврони в мрежата и техния тип (Фиг. 6.12), стойности на теглата и стойности за силата на връзките между невроните (Фиг. 6.13).

```

187
188 //TODO Matrix transpose is possible.
189 weights = new double[numberOfNeurons][numberOfNeurons];
190 JSONArray array2 = result.getJSONArray(JSON_WEIGHTS_KEY);
191 for (int j = 0; j < array2.length(); j++) {
192     JSONArray array3 = array2.getJSONArray(j);
193     for (int i = 0; i < array3.length(); i++) {
194         weights[i][j] = array3.getDouble(i);
195     }
196 }
197
198 //TODO Matrix transpose is possible.
199 activities = new double[numberOfNeurons][numberOfNeurons];
200 JSONArray array4 = result.getJSONArray(JSON_ACTIVITIES_KEY);
201 for (int j = 0; j < array4.length(); j++) {
202     JSONArray array5 = array4.getJSONArray(j);
203     for (int i = 0; i < array5.length(); i++) {
204         activities[i][j] = array5.getDouble(i);
205     }
206 }
207 } catch (ClientProtocolException exception) {
208     return false;
209 } catch (IOException exception) {
210     return false;
211 } catch (JSONException exception) {
212     return false;
213 } catch (Exception exception) {
214     return false;
215 }
216
217
HttpHelper -> load()

```

Фигура 6.13: Стойности за теглата и силата на връзките

При втория фрагмент се получава информация за времевия ред, което включва времеви масив, масив отваря, масив най-ниска (Фиг. 6.14), масив най-висока, масив затваря и масив за търгуван обем (Фиг. 6.15).

```

231 HttpResponse response = client.execute(post);
232
233 JSONObject result = new JSONObject(EntityUtils.toString(response.getEntity(), "UTF-8"));
234
235 int size = result.getInt(JSON_NUMBER_OF_EXAMPLES_KEY);
236
237 /*
238  * If there is no ANN training set on the server side nothing can be loaded.
239  */
240 if (size <= 0) {
241     return false;
242 }
243
244 long[] time = new long[size];
245 JSONArray array1 = result.getJSONArray(JSON_TIME_KEY);
246 for (int i = 0; i < array1.length(); i++) {
247     time[i] = array1.getLong(i);
248 }
249
250 double[] open = new double[size];
251 JSONArray array2 = result.getJSONArray(JSON_OPEN_KEY);
252 for (int i = 0; i < array2.length(); i++) {
253     open[i] = array2.getDouble(i);
254 }
255
256 double[] low = new double[size];
257 JSONArray array3 = result.getJSONArray(JSON_LOW_KEY);
258 for (int i = 0; i < array3.length(); i++) {
259     low[i] = array3.getDouble(i);
260 }
261

```

HttpHelper → load()

Run | TODO | Logcat | Android Profiler | Version Control | Terminal | Build

Gradle build finished in 9s 354ms (today 15:46) | 803 chars, 28 line breaks | 261:1 | LF | UTF-8 | Git: master | Context: <no context>

Фигура 6.14: Стойности за време, отваряне и най-ниска постигната стойност

```

260     }
261
262     high = new double[size];
263     JSONArray array4 = result.getJSONArray(JSON_HIGH_KEY);
264     for (int i = 0; i < array4.length(); i++) {
265         high[i] = array4.getDouble(i);
266     }
267
268     close = new double[size];
269     JSONArray array5 = result.getJSONArray(JSON_CLOSE_KEY);
270     for (int i = 0; i < array5.length(); i++) {
271         close[i] = array5.getDouble(i);
272     }
273
274     volume = new double[size];
275     JSONArray array6 = result.getJSONArray(JSON_VOLUME_KEY);
276     for (int i = 0; i < array6.length(); i++) {
277         volume[i] = array6.getDouble(i);
278     }
279 } catch (ClientProtocolException exception) {
280     return false;
281 } catch (IOException exception) {
282     return false;
283 } catch (JSONException exception) {
284     return false;
285 } catch (Exception exception) {
286     return false;
287 }
288
289 /*
290

```

Фигура 6.15: Стойности за най-висока постигната стойност, затваряне и изтъргуван обем

При успешно получаване на информацията от отдалечения сървър, данните се записват в глобалната структура за пресмятане (Фиг. 6.16).

```

287 }
288
289 /*
290  * Load data in the global data structure.
291 */
292 InputData.SYMBOL = symbol;
293 InputData.PERIOD = period;
294 InputData.NEURONS = flags;
295 InputData.WEIGHTS = weights;
296 InputData.ACTIVITIES = activities;
297 InputData.TIME = time;
298 InputData.OPEN = open;
299 InputData.LOW = low;
300 InputData.HIGH = high;
301 InputData.CLOSE = close;
302 InputData.VOLUME = volume;
303 InputData.RATES = new double[][]{InputData.OPEN, InputData.LOW, InputData.HIGH,
304
305 return true;
306
307
308 k
309 Store calculated ANN weights on the remote web server.
310
311 @return True if the saving was successful, false otherwise.
312
313 public boolean store() {
314     HttpClient client = new DefaultHttpClient();
315     client.getParams().setParameter(s: "http.protocol.content-charset", o: "UTF-8");
316
317

```

Фигура 6.16: Записване на данните в глобалната структура

## Глава 7

# Обучение на мрежата

Обучението при класическите многослойни изкуствени невронни мрежи е свързано с оптимизация на теглата, които характеризират връзките между отделните неврони. По своята същност задачата за обучение представлява числена оптимизация в многомерно пространство на реалните числа. През последните няколко десетилетия са разработени множество точни числени методи и множество евристични методи. В настоящата разработка акцентът пада на два метода (единият точен числен, а другият евристичен). Като точен числен метод софтуерната библиотека Encog предлага широко използвания метод с обратно разпространение на грешката (както и някои негови модификации). От групата на евристичните методи се използват генетични алгоритми с модификация на мутацията, според метода за еволюция на разлики-те. За реализацията на генетичния алгоритъм се използва софтуерната библиотека Genetic Algorithms - Apache Commons.

### 7.1 Обратно разпространение на грешката

При обратното разпространение на грешката, в работен режим, сигналите пътуват от входа на мрежата към нейния изход. Тренировъчните примери се прилагат в случайно определен ред, така че мрежата да не заучи схемата на подаване, а да развие своите обобщаващи способности. Общата допусната грешка се изчислява на изхода на мрежата и след това се връща по обратния път, от изход към вход, за да бъде изчислена грешката с която всеки неврон допринася. След определянето на индивидуалната грешка за всеки неврон се извършва числена корекция (според градиента) на теглата, които свързват невроните. Този обратен ход на пресмятане дава името на метода и също така го поставя в групата на точните числени методи, които разчитат на градиента за да извършат корекция в теглата. Същественото за обучението с обратно разпространение на грешката е, че активационната функция на неврона трябва да бъде диференцируема. Библиотеката Encog предлага група методи за оптимизация на теглата в мрежата.

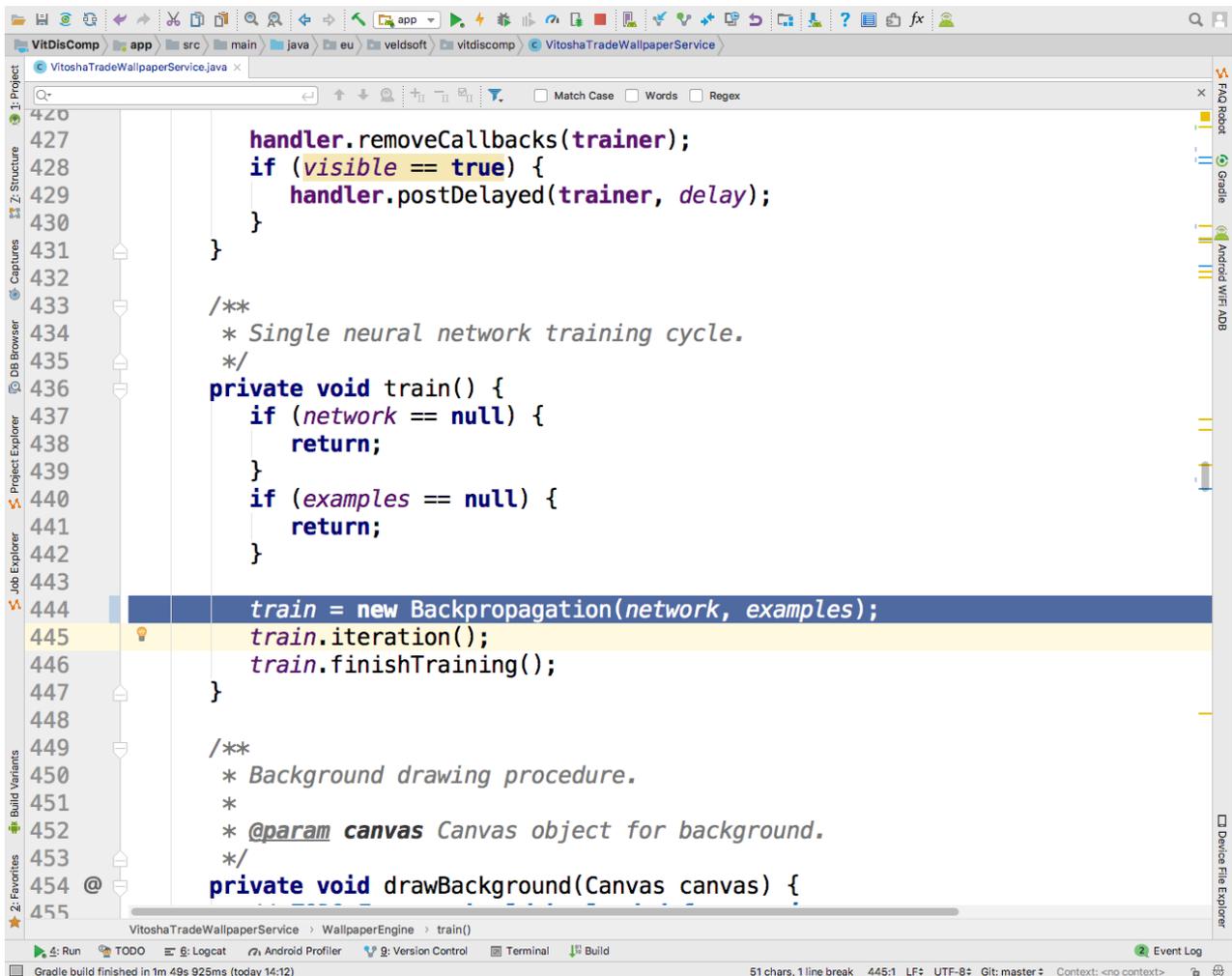
```

427         handler.postDelayed(trainer, delay);
428     }
429 }
430
431 /**
432  * Single neural network training cycle.
433  */
434 private void train() {
435     if (network == null) {
436         return;
437     }
438     if (examples == null) {
439         return;
440     }
441
442     train = new ResilientPropagation(network, examples);
443     train.iteration();
444     train.finishTraining();
445 }
446
447 /**
448  * Background drawing procedure.
449  *
450  * @param canvas Canvas object for background.
451  */
452 private void drawBackground(Canvas canvas) {
453     // TODO Images should be loaded from an image server.
454     /*
455     * Change picture according the day in the year.
456     */
457 }

```

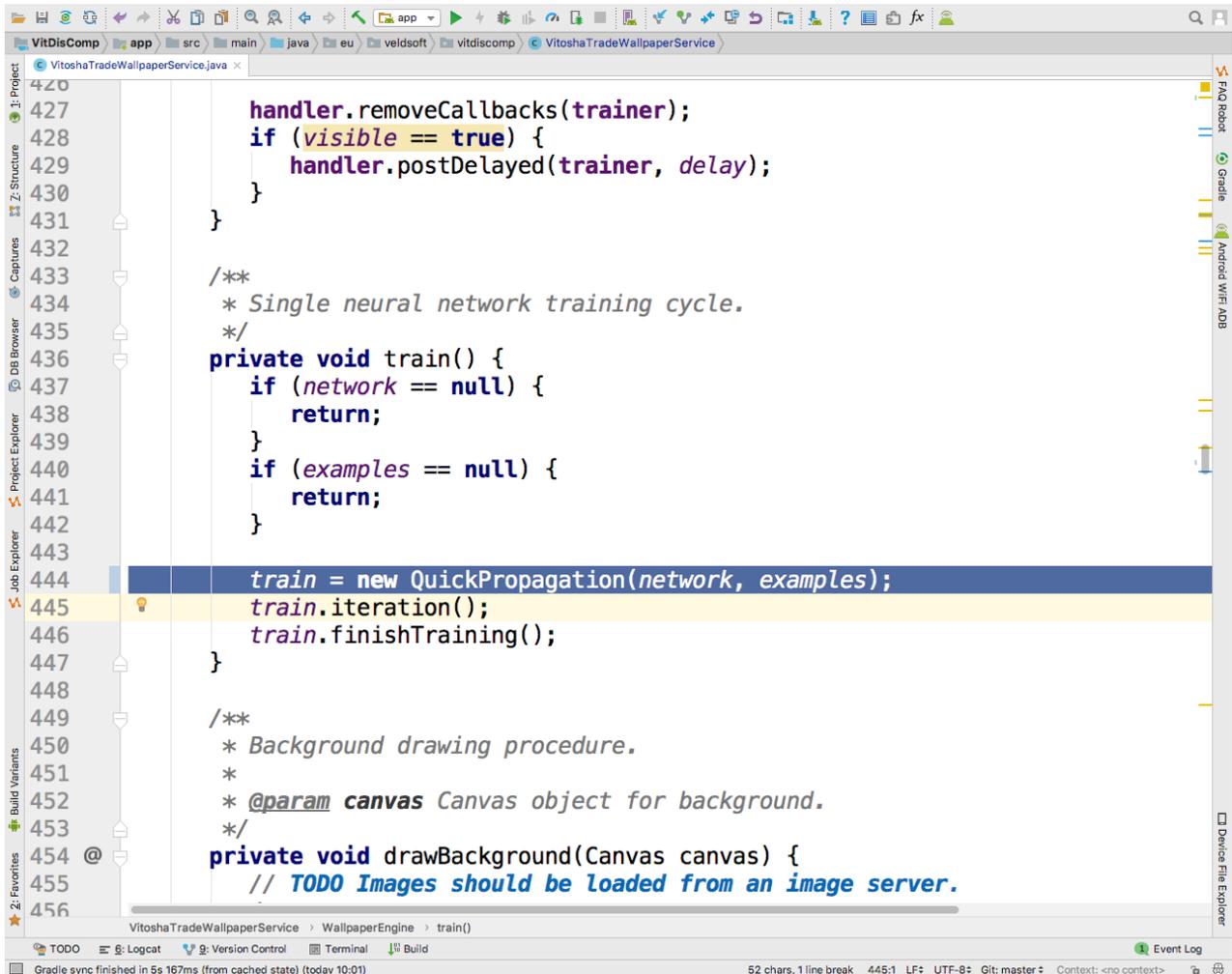
Фигура 7.1: Единична стъпка за обучение на мрежата

Библиотеката Encog позволява няколко възможности за обучение с точни числени методи и в настоящото помагало е избран методът Resilient Propagation, който представлява еластично обратно разпространение на грешката (Фиг. 7.1). Еластичното разпространение на грешката е модификация на основния метод, в която модификация степента за научаване (learning rate) се определя динамично и не е само една стойност за цялата мрежа. При класическия подход с обратно разпространение на грешката степента за научаване е само една стойност, обикновено емпирично нагласена между 0.0 и 1.0. Най-често се използва 0.35, което е компромис между бързината за научаване и достигането на оптимална стойност за теглата (фината стъпка за нагласяване на теглата в крайната фаза от обучението).



Фигура 7.2: Класическо обратно разпространение на грешката

Фактът, че библиотеката Encog е написана по каноните на обектно-ориентираното програмиране, позволява само с подмяната на един ред (Фиг. 7.2) от еластично обучение да преминем към класическо обратно разпространение на грешката.



Фигура 7.3: Бързо разпространение на грешката

С абсолютно същата лекота, библиотеката позволява да се премине към бързо разпространение на грешката (Фиг. 7.3). Тази модификация на алгоритъма за обратно разпространение на грешката въвежда подобрение, чрез използването на метода Нютон-Равсън за допълнително определяне на втората производна, така че допълнителната информация от втората производна да подобри начина, по който се коригират теглата в мрежата. Благодарение на тази модификация, по наклоните на функцията за грешка се правят по-големи стъпки, а при доближаване на екстремалната точка стъпката се намалява.

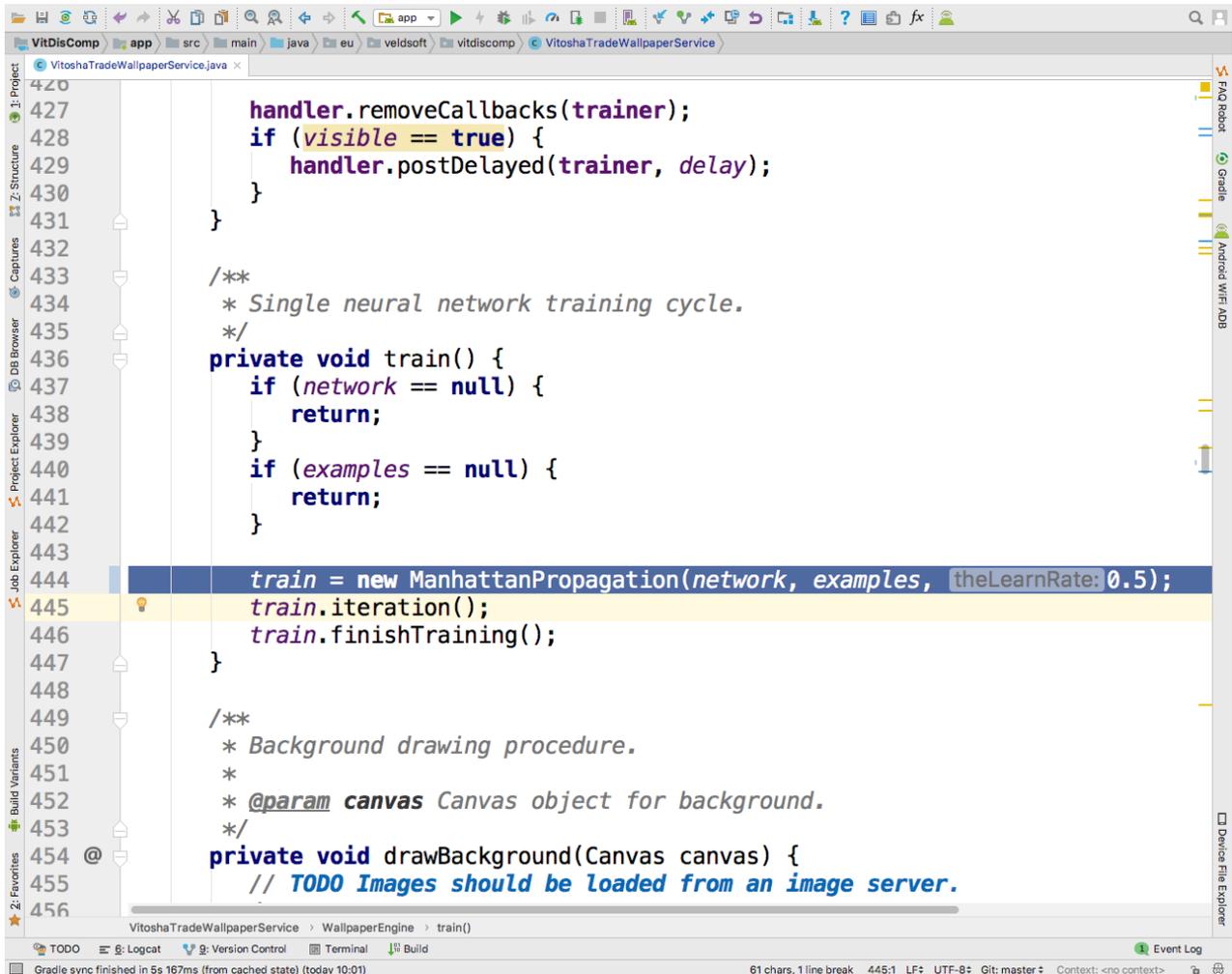
```

420
427     handler.removeCallbacks(trainer);
428     if (visible == true) {
429         handler.postDelayed(trainer, delay);
430     }
431 }
432
433 /**
434  * Single neural network training cycle.
435  */
436 private void train() {
437     if (network == null) {
438         return;
439     }
440     if (examples == null) {
441         return;
442     }
443
444     train = new ScaledConjugateGradient(network, examples);
445     train.iteration();
446     train.finishTraining();
447 }
448
449 /**
450  * Background drawing procedure.
451  *
452  * @param canvas Canvas object for background.
453  */
454 @
455 private void drawBackground(Canvas canvas) {
456     // TODO Images should be loaded from an image server.

```

Фигура 7.4: Мащабирано съединение на градиента

При мащабираното съединение на градиента (Фиг. 7.4) също се използва информацията от втората производна. По този начин се намира по-добър път към оптималната точка в сравнение с методите, които използват само първа производна. Цената на такова подобрение е нуждата от използването на повече изчислителни ресурси. Разликата при този метод е, че корекцията на теглата не винаги са в посоката посочена от първата производна.



Фигура 7.5: Корекция на теглата по Манхатън правило

Опростен вариант на еластичното обучение е корекцията на теглата по правилото на Манхатън (Фиг. 7.5). При тази модификация на обратното разпространение на грешката се използва само посоката на първата производна като за корекция на теглата се прилага единична, предварително дефинирана стойност (`theLearningRate`). При този метод ползата идва от това, че понякога първата производна се пресмята до твърде голяма или твърде малка стойност. Когато корекцията е с фиксирана стойност и се използва само посоката на производната се избягва негативният ефект на големи/малки стойности за корекция. Недостатък на този алгоритъм е, че единичната стойност за корекция трябва да бъде адаптивно нагласяна. Обикновено се започва с по-голяма стойност и се намалява, по аналогия с метода за симулирано закаляване.

## 7.2 Генетични алгоритми

Въпреки че библиотеката Епсог има възможности за обучение на изкуствени невронни мрежи с генетични алгоритми (класът `MLMethodGeneticAlgorithm`) в настоящото помагало е предпочетена библиотеката `Genetic Algorithms - Apache Commons`, тъй като тя дава много повече свобода за реализация на операциите в генетичния алгоритъм и разширения в посока на метода за еволюция на разликите.

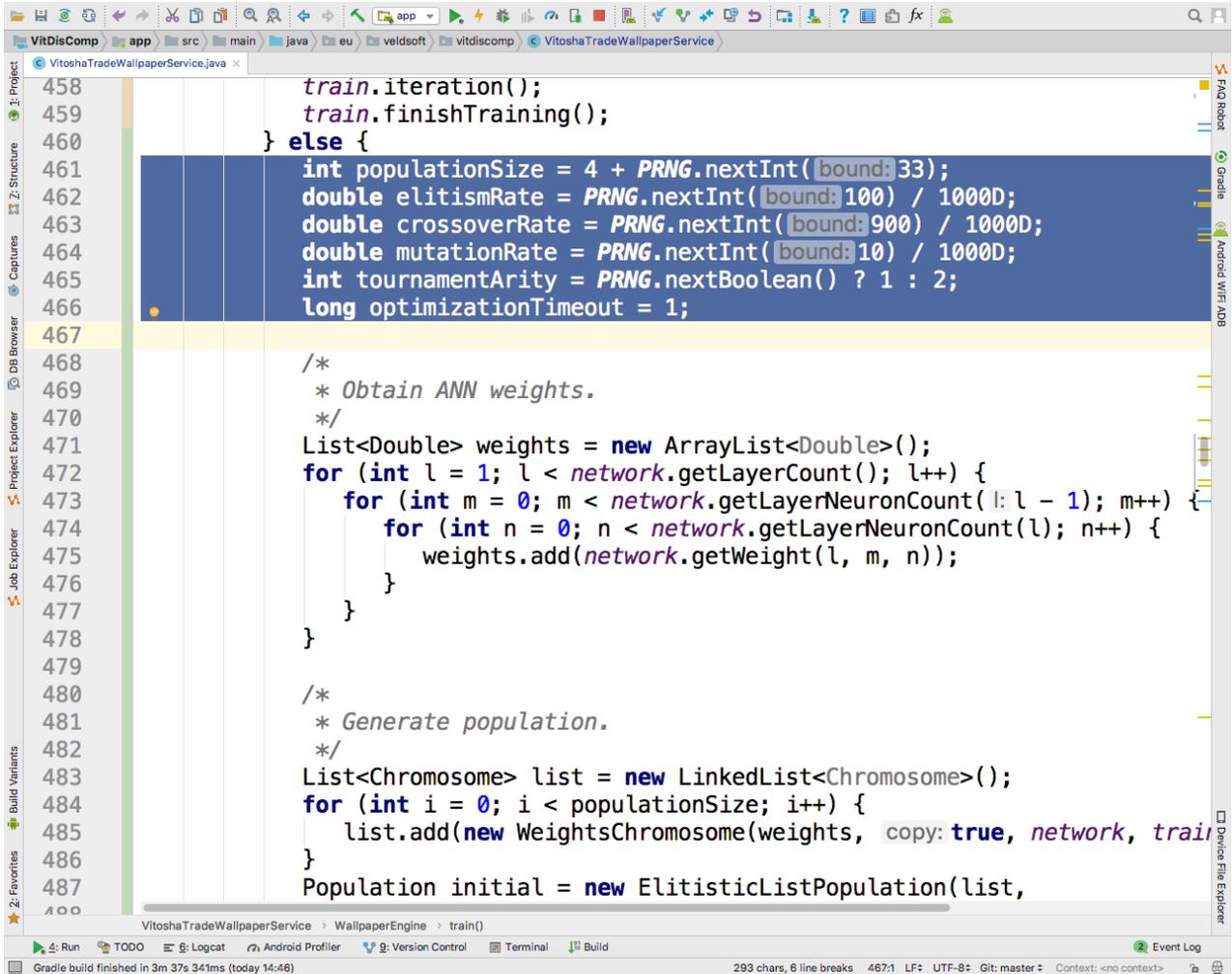
```

449     return;
450 }
451
452 train = new ResilientPropagation(network, examples);
453
454 /*
455  * Switch between backpropagation and genetic algorithm.
456  */
457 if (PRNG.nextBoolean() == true) {
458     train.iteration();
459     train.finishTraining();
460 } else {
461     int populationSize = 4 + PRNG.nextInt( bound: 33);
462     double elitismRate = PRNG.nextInt( bound: 100) / 1000D;
463     double crossoverRate = PRNG.nextInt( bound: 900) / 1000D;
464     double mutationRate = PRNG.nextInt( bound: 10) / 1000D;
465     int tournamentArity = PRNG.nextBoolean() ? 1 : 2;
466     long optimizationTimeout = 1;
467
468     /*
469     * Obtain ANN weights.
470     */
471     List<Double> weights = new ArrayList<Double>();
472     for (int l = 1; l < network.getLayerCount(); l++) {
473         for (int m = 0; m < network.getLayerNeuronCount( l: l - 1); m++) {
474             for (int n = 0; n < network.getLayerNeuronCount( l); n++) {
475                 weights.add(network.getWeight( l, m, n));
476             }
477         }
478     }

```

Фигура 7.6: Превключване между обратно разпространение на грешката и генетичен алгоритъм

С добавянето на генетичен алгоритъм се получава хибридна система за обучение на изкуствени невронни мрежи (Фиг. 7.6). На случаен принцип, в половината от случаите се извършва обучение с точен числен метод, а в другата половина с евристичен метод.



```

458     train.iteration();
459     train.finishTraining();
460 } else {
461     int populationSize = 4 + PRNG.nextInt(bound: 33);
462     double elitismRate = PRNG.nextInt(bound: 100) / 1000D;
463     double crossoverRate = PRNG.nextInt(bound: 900) / 1000D;
464     double mutationRate = PRNG.nextInt(bound: 10) / 1000D;
465     int tournamentArity = PRNG.nextBoolean() ? 1 : 2;
466     long optimizationTimeout = 1;
467
468     /*
469      * Obtain ANN weights.
470      */
471     List<Double> weights = new ArrayList<Double>();
472     for (int l = 1; l < network.getLayerCount(); l++) {
473         for (int m = 0; m < network.getLayerNeuronCount(l - 1); m++) {
474             for (int n = 0; n < network.getLayerNeuronCount(l); n++) {
475                 weights.add(network.getWeight(l, m, n));
476             }
477         }
478     }
479
480     /*
481      * Generate population.
482      */
483     List<Chromosome> list = new LinkedList<Chromosome>();
484     for (int i = 0; i < populationSize; i++) {
485         list.add(new WeightsChromosome(weights, copy: true, network, train));
486     }
487     Population initial = new ElitisticListPopulation(list,
488

```

Фигура 7.7: Параметри на генетичния алгоритъм

Генетичният алгоритъм се настройва с група параметри (размер на популацията, процент на елита в популацията, честота за кръстосване, честота за мутация, численост за турнира при селекция и максимален брой секунди за извършване на оптимизация), които в случая са статично зададени, но е разумно да бъдат управлявани от конфигурационен файл или от екран за настройки (Фиг. 7.7).

```

452     train = new ResilientPropagation(network, examples);
453
454     /*
455      * Switch between backpropagation and genetic algorithm.
456      */
457     if (PRNG.nextBoolean() == true) {
458         train.iteration();
459         train.finishTraining();
460     } else {
461         int populationSize = 4 + PRNG.nextInt( bound: 33);
462         double elitismRate = PRNG.nextInt( bound: 100) / 1000D;
463         double crossoverRate = PRNG.nextInt( bound: 900) / 1000D;
464         double mutationRate = PRNG.nextInt( bound: 10) / 1000D;
465         int tournamentArity = PRNG.nextBoolean() ? 1 : 2;
466         long optimizationTimeout = 1;
467
468         /*
469          * Obtain ANN weights.
470          */
471         List<Double> weights = new ArrayList<Double>();
472         for (int l = 1; l < network.getLayerCount(); l++) {
473             for (int m = 0; m < network.getLayerNeuronCount(l - 1); m++) {
474                 for (int n = 0; n < network.getLayerNeuronCount(l); n++) {
475                     weights.add(network.getWeight(l, m, n));
476                 }
477             }
478         }
479
480         /*
481          * Generate population.

```

Фигура 7.8: Извличане на теглата от мрежата

Теглата на мрежата се подреждат във вектор от реални стойности (Фиг. 7.8).

```

468     /*
469     * Obtain ANN weights.
470     */
471     List<Double> weights = new ArrayList<Double>();
472     for (int l = 1; l < network.getLayerCount(); l++) {
473         for (int m = 0; m < network.getLayerNeuronCount(l - 1); m++) {
474             for (int n = 0; n < network.getLayerNeuronCount(l); n++) {
475                 weights.add(network.getWeight(l, m, n));
476             }
477         }
478     }
479
480     /*
481     * Generate population.
482     */
483     List<Chromosome> list = new LinkedList<Chromosome>();
484     for (int i = 0; i < populationSize; i++) {
485         list.add(new WeightsChromosome(weights, copy: true, network, train));
486     }
487     Population initial = new ElitisticListPopulation(list,
488         populationLimit: 2 * list.size(), elitismRate);
489
490     /*
491     * Initialize genetic algorithm.
492     */
493     GeneticAlgorithm algorithm = new GeneticAlgorithm(
494         new UniformCrossover<WeightsChromosome>(ratio: 0.5),
495         crossoverRate, new UniformBinaryMutation(),
496         mutationRate, new TournamentSelection(tournamentArity));
497
498

```

Фигура 7.9: Начална популация

От извлечените тегла се формира начална популация за стартиране на оптимизацията (Фиг. 7.9).

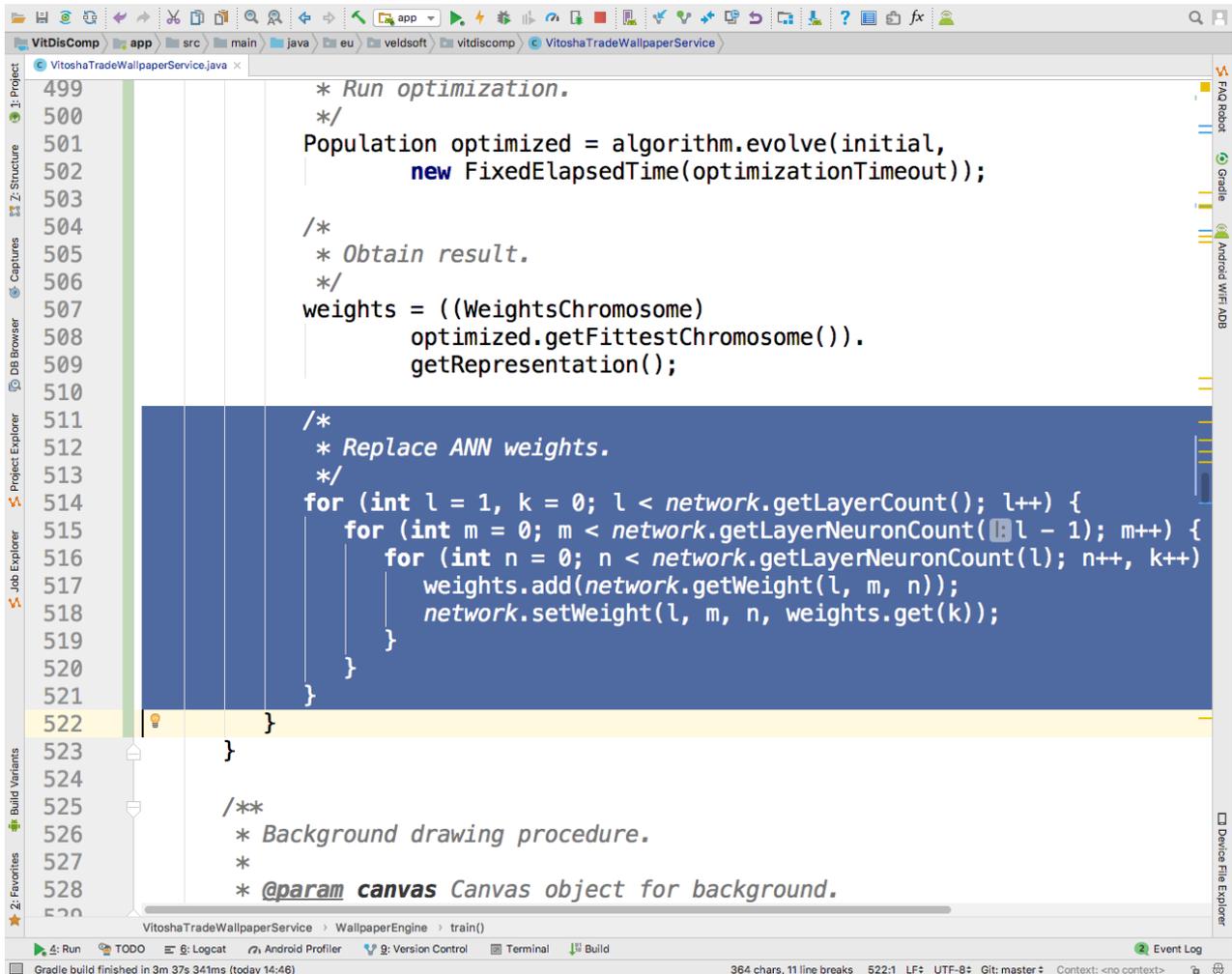
```

485         list.add(new WeightsChromosome(weights, copy: true, network, train));
486     }
487     Population initial = new ElitisticListPopulation(list,
488         populationLimit: 2 * list.size(), elitismRate);
489
490     /*
491     * Initialize genetic algorithm.
492     */
493     GeneticAlgorithm algorithm = new GeneticAlgorithm(
494         new UniformCrossover<WeightsChromosome>(ratio: 0.5),
495         crossoverRate, new UniformBinaryMutation(),
496         mutationRate, new TournamentSelection(tournamentArity));
497
498     /*
499     * Run optimization.
500     */
501     Population optimized = algorithm.evolve(initial,
502         new FixedElapsedTime(optimizationTimeout));
503
504     /*
505     * Obtain result.
506     */
507     weights = ((WeightsChromosome)
508         optimized.getFittestChromosome()).
509         getRepresentation();
510
511     /*
512     * Replace ANN weights.
513     */
514     weights = new ArrayList<Double>();

```

Фигура 7.10: Изпълнение на генетичния алгоритъм

Следва изпълнение на генетичния алгоритъм (Фиг. 7.10). За кръстосване се прилага равномерно случайно кръстосване. За мутация се прилага равномерна бинарна мутация върху всички гени на хромозомата. Най-добрият индивид от оптимизираната популация се зарежда обратно в мрежата (Фиг. 7.11).



Фигура 7.11: Обновяване на теглата в мрежата

### 7.2.1 Кодирание на хромозомите

Тъй като теглата на мрежата са множество от реални числа, то те идеално могат да бъдат представени в хромозома съдържаща списък от стойности (Фиг. 7.12). Тъй като за оценката на жизнеността е нужно теглата да бъдат зареждани в мрежата и над мрежата да се изпълнява правият пас от обучението с обратно разпространение на грешката, то хромозомата съдържа референции към обекта за мрежа и обекта за обучение.

```

7
8 import java.util.List;
9
10 /**
11  * Chromosome with double values of the weights from the artificial neural
12  * network.
13  *
14  * @author Todor Balabanov
15  */
16 class WeightsChromosome extends AbstractListChromosome<Double> {
17
18     /**
19      * Reference to external neural network object.
20      */
21     private BasicNetwork network = null;
22
23     /**
24      * Reference to external training strategy object.
25      */
26     private Train train = null;
27
28     /**
29      * Constructor used to initialize the chromosome with array of values.
30      *
31      * @param representation Values as array.
32      * @param network         Neural network reference.
33      * @param train           Neural network training strategy.
34      * @throws InvalidRepresentationException Rise an exception if the values
35      * are not valid.
36      */

```

Фигура 7.12: Кодирание на теглата в хромозомите

За гъвкавост при създаването на обектите хромозоми са предефинирани трите наследени конструктора (Фиг. 7.13, Фиг. 7.14, Фиг. 7.15).

```

25  */
26  private Train train = null;
27
28  /**
29   * Constructor used to initialize the chromosome with array of values.
30   *
31   * @param representation Values as array.
32   * @param network        Neural network reference.
33   * @param train          Neural network training strategy.
34   * @throws InvalidRepresentationException Rise an exception if the values
35   * are not valid.
36   */
37  public WeightsChromosome(Double[] representation,
38                          BasicNetwork network, Train train)
39      throws InvalidRepresentationException {
40      super(representation);
41      this.network = network;
42      this.train = train;
43
44      if (network == null) {
45          throw new RuntimeException("Neural network should be provided for the f
46      }
47
48      if (train == null) {
49          throw new RuntimeException("Training object should be provided for the f
50      }
51  }
52
53  /**
54   * Constructor used to initialize the chromosome with list of values.
55

```

Фигура 7.13: Конструирание по зададен масив

```

51 }
52
53 /**
54  * Constructor used to initialize the chromosome with list of values.
55  *
56  * @param representation Values as list.
57  * @param network         Neural network reference.
58  * @param train           Neural network training strategy.
59  * @throws InvalidRepresentationException Rise an exception if the values
60  * are not valid.
61  */
62 public WeightsChromosome(List<Double> representation,
63                          BasicNetwork network, Train train)
64     throws InvalidRepresentationException {
65     super(representation);
66     this.network = network;
67     this.train = train;
68
69     if (network == null) {
70         throw new RuntimeException("Neural network should be provided for the fitness function");
71     }
72
73     if (train == null) {
74         throw new RuntimeException("Training object should be provided for the fitness function");
75     }
76 }
77
78 /**
79  * Constructor used to initialize the chromosome with list of values.
80  *
81  */

```

Фигура 7.14: Конструирание по зададен списък

```

79  * Constructor used to initialize the chromosome with list of values.
80  *
81  * @param representation Values as list.
82  * @param copy           Deep copy flag.
83  * @param network        Neural network reference.
84  * @param train          Neural network training strategy.
85  */
86  public WeightsChromosome(List<Double> representation, boolean copy,
87                          BasicNetwork network, Train train) {
88      super(representation, copy);
89      this.network = network;
90      this.train = train;
91
92      if (network == null) {
93          throw new RuntimeException("Neural network should be provided for the fitness");
94      }
95
96      if (train == null) {
97          throw new RuntimeException("Training object should be provided for the fitness");
98      }
99  }
100
101  /**
102   * {@inheritDoc}
103   */
104  @Override
105  public double fitness() {
106      if (network == null) {
107          throw new RuntimeException("Neural network should be provided for the fitness");
108      }
109  }

```

Фигура 7.15: Конструирание по зададен списък с възможност за дълбоко копиране

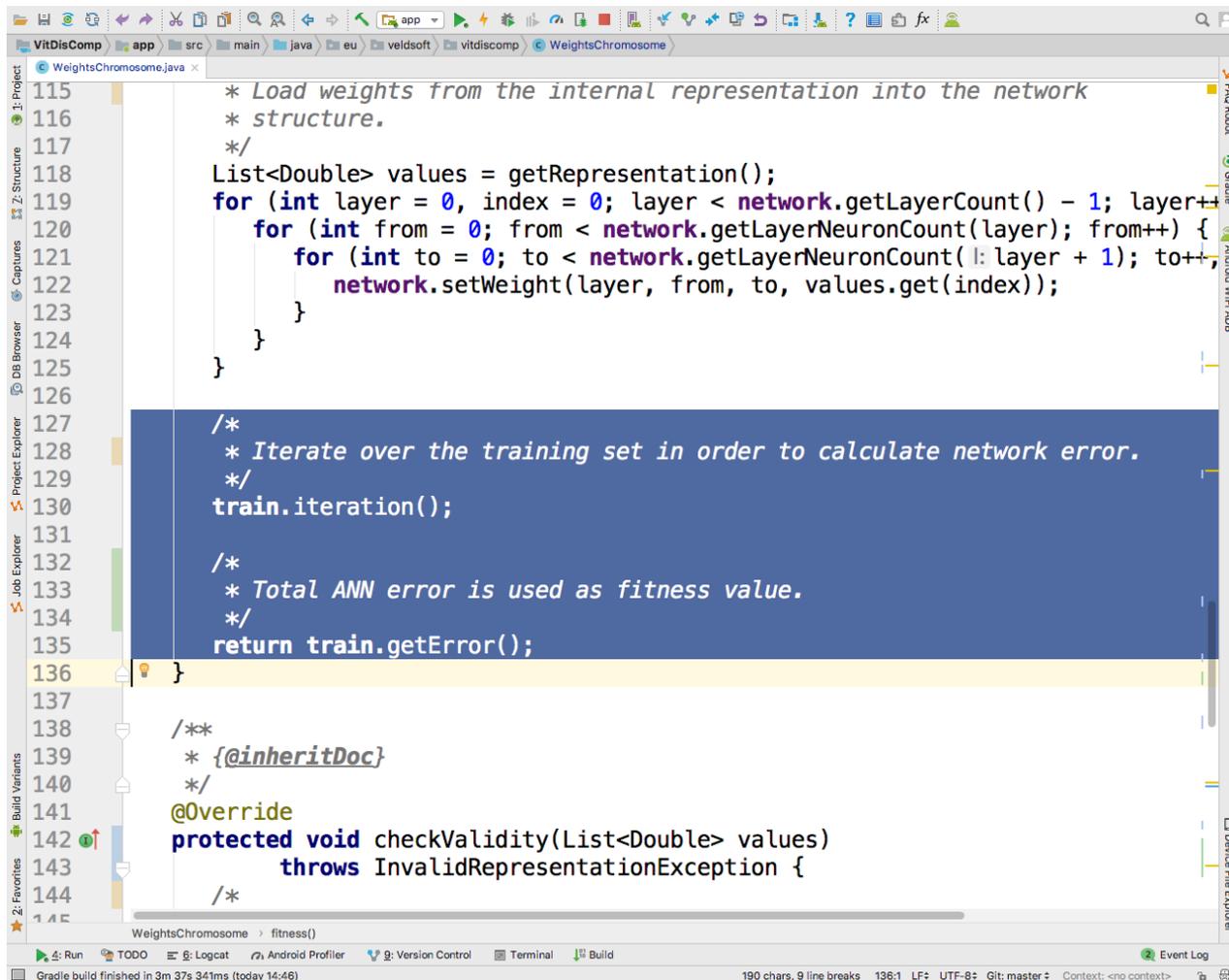
Определянето на жизнеността започва със зареждане на дробните стойности като тегла в изкуствената невронна мрежа (Фиг. 7.16).

```

99     }
100
101     /**
102     * {@inheritDoc}
103     */
104     @Override
105     public double fitness() {
106         if (network == null) {
107             throw new RuntimeException("Neural network should be provided for the fitness() method");
108         }
109
110         if (train == null) {
111             throw new RuntimeException("Training object should be provided for the fitness() method");
112         }
113
114         /**
115          * Load weights from the internal representation into the network
116          * structure.
117          */
118         List<Double> values = getRepresentation();
119         for (int layer = 0, index = 0; layer < network.getLayerCount() - 1; layer++) {
120             for (int from = 0; from < network.getLayerNeuronCount(layer); from++) {
121                 for (int to = 0; to < network.getLayerNeuronCount(layer + 1); to++) {
122                     network.setWeight(layer, from, to, values.get(index));
123                 }
124             }
125         }
126
127         /**
128          * Iterate over the training set in order to calculate network error.
129          */
130     }
    
```

Фигура 7.16: Зареждане на стойностите като тегла в мрежата

Изпълнява се правият пас и общата грешка допусната от мрежата се връща като жизнена стойност (Фиг. 7.17).



Фигура 7.17: Изпълнение на правия пас

Валидността на хромозомата зависи единствено от това да има достатъчно дробни числа, така че да се съпостави стойност за всяко тегло в мрежата (Фиг. 7.18).

```

133     * Total ANN error is used as fitness value.
134     */
135     return train.getError();
136 }
137
138 /**
139  * {@inheritDoc}
140  */
141 @Override
142 protected void checkValidity(List<Double> values)
143     throws InvalidRepresentationException {
144     /*
145     * Length of the values should match the number of weights in the neural
146     * network structure.
147     */
148     int counter = 0;
149     for (int layer = 0; layer < network.getLayerCount() - 1; layer++) {
150         for (int from = 0; from < network.getLayerNeuronCount(layer); from++) {
151             for (int to = 0; to < network.getLayerNeuronCount(layer + 1); to++)
152                 counter++;
153         }
154     }
155
156     if (values == null || counter != values.size()) {
157         // TODO Report the size problem.
158     }
159 }
160
161
162 /**
163

```

Фигура 7.18: Условие за валидност на хромозомата

Конструирането на нова хромозома с фиксирана дължина се извършва чрез извикването на един от конструкторите и използването на дълбоко копиране за стойностите (Фиг. 7.19).

```

152         counter++;
153     }
154 }
155 }
156 }
157     if (values == null || counter != values.size()) {
158         // TODO Report the size problem.
159     }
160 }
161
162 /**
163  * {@inheritDoc}
164  */
165 @Override
166 public AbstractListChromosome<Double> newFixedLengthChromosome(
167     List<Double> values) {
168     return new WeightsChromosome(values, copy: true, network, train);
169 }
170
171 /**
172  * Chromosome representation getter.
173  *
174  * @return List with chromosome values.
175  */
176 public List<Double> getRepresentation() {
177     return getRepresentation();
178 }
179 }
180

```

Фигура 7.19: Конструирание на нова хромозома с фиксирана дължина

За нуждите на пресмятанията извън класа е създадена функция за извеждане на стойностите (Фиг. 7.20).

```

152         counter++;
153     }
154 }
155 }
156
157     if (values == null || counter != values.size()) {
158         // TODO Report the size problem.
159     }
160 }
161
162 /**
163  * {@inheritDoc}
164  */
165 @Override
166 public AbstractListChromosome<Double> newFixedLengthChromosome(
167     List<Double> values) {
168     return new WeightsChromosome(values, copy: true, network, train);
169 }
170
171 /**
172  * Chromosome representation getter.
173  *
174  * @return List with chromosome values.
175  */
176 public List<Double> getRepresentation() {
177     return getRepresentation();
178 }
179 }
180

```

Фигура 7.20: Функция за извеждане на стойностите извън хромозомата

## 7.2.2 Случайна равномерна мутация

Тъй като библиотеката Genetic Algorithms - Apache Commons не предлага подходяща бинарна мутация за вектор от реални числа, то за нуждите на разработката е предложен клас изпълняващ операцията по мутация (Фиг. 7.21).

```

8  import org.apache.commons.math3.random.RandomGenerator;
9
10 import java.util.ArrayList;
11 import java.util.List;
12
13 /**
14  * Uniform random bits mutation applied over vector of double values.
15  *
16  * @author Todor Balabanov
17  */
18 class UniformBinaryMutation implements MutationPolicy {
19
20     /**
21     * Pseudo random number generator.
22     */
23     private static final RandomGenerator PRNG =
24         GeneticAlgorithm.getRandomGenerator();
25
26     /**
27     * {@inheritDoc}
28     */
29     @Override
30     public Chromosome mutate(Chromosome original)
31         throws MathIllegalArgumentException {
32         if (original instanceof WeightsChromosome == false) {
33             throw new MathIllegalArgumentException(
34                 LocalizedFormats.INVALID_BINARY_CHROMOSOME);
35         }
36
37     /**

```

Фигура 7.21: Клас извършващ мутацията

Предложената операция за мутация е удачно да се прилага само върху използваното кодиране в хромозомите. Поради тази причина функцията за мутация извършва нужната проверка на входните данни (Фиг. 7.22).

```

26  /**
27   * {@inheritDoc}
28   */
29  @Override
30  public Chromosome mutate(Chromosome original)
31      throws MathIllegalArgumentException {
32      if (original instanceof WeightsChromosome == false) {
33          throw new MathIllegalArgumentException(
34              LocalizedFormats.INVALID_BINARY_CHROMOSOME);
35      }
36
37      /**
38       * Deep copy of list values.
39       */
40      List<Double> representation = new ArrayList<Double>(
41          ((WeightsChromosome) original).getRepresentation());
42
43      /**
44       * Mutate a single bit in each chromosome value.
45       */
46      for (int i = 0; i < representation.size(); i++) {
47          double value = representation.get(i);
48          if (PRNG.nextBoolean() == true) {
49              value -= Double.MIN_VALUE;
50          } else {
51              value += Double.MIN_VALUE;
52          }
53          representation.set(i, value);
54      }
55
56  }

```

Фигура 7.22: Контрол на входните данни

За разлика от класическата мутация в генетичните алгоритми, тук е направена модификация, наподобяваща метода за еволюция на разликите. Всеки ген в хромозомата бива променен с малка дробна стойност (Фиг. 7.23).

```

35     }
36
37     /*
38     * Deep copy of list values.
39     */
40     List<Double> representation = new ArrayList<Double>(
41         ((WeightsChromosome) original).getRepresentation());
42
43     /*
44     * Mutate a single bit in each chromosome value.
45     */
46     for (int i = 0; i < representation.size(); i++) {
47         double value = representation.get(i);
48         if (PRNG.nextBoolean() == true) {
49             value -= Double.MIN_VALUE;
50         } else {
51             value += Double.MIN_VALUE;
52         }
53         representation.set(i, value);
54     }
55
56     /*
57     * Construct new chromosome after mutation.
58     */
59     return ((WeightsChromosome) original).
60         newFixedLengthChromosome(representation);
61 }
62 }
63

```

Фигура 7.23: Случайна промяна на всички стойности в хромозомата

Мутацията приключва със създаване и връщане на мутирал екземпляр.

## Заклучение

Основна цел на настоящото учебно помагало е представянето на възможностите, които съвременните Android мобилни устройства могат да предложат за извършване на научни изчисления в разпределена среда. Без да претендира за изчерпателност изложеният материал има за цел да провокира творческото мислене у читателя и да го вдъхнови за създаването на собствени авторски проекти с разгледаните технологии и засегнатите научни области.

Авторите са благодарни на своите читатели за отделеното време и внимание като горещо насърчават подаването на обратна връзка и споделянето на интересни мисли, идеи или предложения, на посочените за връзка контакти.

## Библиография

- [1] Distributed Computing Info  
<http://www.distributedcomputing.info/>
- [2] Paul Shuch, H. *Searching for Extraterrestrial Intelligence*. Springer-Verlag Berlin Heidelberg, 2011.
- [3] Bohn, A., Guting, T., Mansmann, T. *MoneyBee: A new product to predict stock market developments using artificial intelligence and increased calculation capacity* German. et al. *Wirtschaftsinf*, vol. 45/3, 325–333, 2003.
- [4] VitoshaTrade Project  
<https://github.com/VelbazhdSoftwareLLC/VitoshaTrade>
- [5] Activation Function - Wikipedia  
[https://en.wikipedia.org/wiki/Activation\\_function](https://en.wikipedia.org/wiki/Activation_function)

## Списък на фигурите

1.1	Сравнение между последователни пресмятания и паралелни пресмятания . . . . .	3
1.2	Сравнение между конкурентни пресмятания и паралелни пресмятания . . . . .	4
2.1	Трислойна изкуствена невронна мрежа . . . . .	7
2.2	Трите основни операции в генетичните алгоритми . . . . .	9
2.3	Отношението евро/долар в рамките на един час . . . . .	10
2.4	Продуктовата линия TradingSolutions използваща изкуствени невронни мрежи и генетични алгоритми за прогнозиране на Forex финансови инструменти . . . . .	11
2.5	Системата MoneyVee за финансово прогнозиране в разпределена среда . . . . .	12
3.1	Уеб базирана трислойна софтуерна архитектура . . . . .	15
3.2	Подходи за софтуерна разработка . . . . .	16
3.3	Публикуване на проект в GitHub . . . . .	17
4.1	Определяне на приложението като активен тапет . . . . .	19
4.2	Модул от тип “услуга” . . . . .	20
4.3	Флагове за достъп до ресурса активен тапет . . . . .	21
4.4	Регистрация на услугата за прослушване на съобщения от операционната система. . . . .	22
4.5	Референция към описателния файл на активния тапет . . . . .	23
4.6	Прозорец за установяване на активния тапет . . . . .	24
4.7	XML описателен файл на тапета . . . . .	25
4.8	Графични файлове, съдържащи фотографии на планината Витоша до град София, България . . . . .	26
4.9	Визуално представяне на информация от изчисленията . . . . .	27
4.10	Първоначален набор от настройки . . . . .	28
4.11	Визуален компонент за включване и изключване на тапета . . . . .	29
4.12	Визуален компонент за посочване на сървър URL . . . . .	30
4.13	Визуален компонент за регулиране на натоварването . . . . .	31
4.14	Стойности за натоварване на системата . . . . .	32
4.15	Позициониране на областите за визуално представяне . . . . .	33
4.16	Стойности за позициониране на областите за визуално представяне . . . . .	34
4.17	Размер на областите за визуално представяне . . . . .	35
4.18	Стойности за размер на областите за визуално представяне . . . . .	36
4.19	Събитие за създаване на прозореца . . . . .	37
4.20	Събитие за пауза на прозореца . . . . .	38
4.21	Наследяване на WallpaperService . . . . .	39
4.22	Спомагателни константи . . . . .	40
4.23	Константи за цветовете . . . . .	41
4.24	Променливи отразяващи състоянието на активния тапет . . . . .	42
4.25	Променливи отговарящи за изкуствената невронна мрежа . . . . .	43
4.26	Променливи отговарящи за изкуствената невронна мрежа . . . . .	44
4.27	Определяне на броя и типовете неврони . . . . .	45
4.28	Структура на трислойна изкуствена невронна мрежа . . . . .	46
4.29	Избор на стойности за прогнозиране . . . . .	47
4.30	Определяне на границите във времевия ред . . . . .	48

4.31	Диапазон на активационната функция в изходния слой . . . . .	49
4.32	Тренировъчни примери . . . . .	50
4.33	Входни данни към изкуствената неверонна мрежа за получаване на прогноза . . . . .	51
4.34	Изследване на група от стойности за определяне на диапазона . . . . .	52
4.35	Активационни функции [5] . . . . .	53
4.36	Собствена реализация на наследения Engine клас и създаване на обект . . . . .	54
4.37	Вътрешни променливи за двигателя на услугата . . . . .	55
4.38	Единичен цикъл във фонов режим . . . . .	56
4.39	Изчисляване на прогноза . . . . .	57
4.40	Тренировъчен цикъл на изкуствената невронна мрежа . . . . .	58
4.41	Основна процедура по изрисване на информацията от тренировъчния процес . . . . .	59
4.42	Разбиване на задачата за визуално представяне . . . . .	60
4.43	Изрисване на фона . . . . .	61
4.44	Изрисване на областите за служебна информация . . . . .	62
4.45	Изрисване на информацията за валутната двойка . . . . .	63
4.46	Позиция и размери за визуално представяне на прогнозата . . . . .	64
4.47	Обхват на данните по ширина и височина . . . . .	65
4.48	Цикли за визуално представяне на стълбовете от времевия реди и прогнозата . . . . .	66
4.49	Топология на изкуствената невронна мрежа, която се визуализира . . . . .	67
4.50	Мащабиране на входната и изходната информация . . . . .	68
4.51	Стойности на теглата между слоевете . . . . .	69
4.52	Мащабиране на стойностите в скрития слой . . . . .	70
4.53	Нормализиране на стойностите за теглата . . . . .	71
4.54	Изрисване на топологията . . . . .	72
4.55	Конструктор на двигателя за услугата . . . . .	73
4.56	Промяна във видимостта на активния тапет . . . . .	74
4.57	Спиране на пресмятанията при разрушаване на повърхността за изрисване . . . . .	75
4.58	Размер на зоните за изрисване . . . . .	76
4.59	Координати и размери на областите за визуално представяне - горе-ляво . . . . .	77
4.60	Координати и размери на областите за визуално представяне - горе-център . . . . .	78
4.61	Координати и размери на областите за визуално представяне - горе-дясно . . . . .	79
4.62	Координати и размери на областите за визуално представяне - център-ляво . . . . .	80
4.63	Координати и размери на областите за визуално представяне - център-център . . . . .	81
4.64	Координати и размери на областите за визуално представяне - център-дясно . . . . .	82
4.65	Координати и размери на областите за визуално представяне - долу-ляво . . . . .	83
4.66	Координати и размери на областите за визуално представяне - долу-център . . . . .	84
4.67	Координати и размери на областите за визуално представяне - долу-дясно . . . . .	85
4.68	Натоварване на системата за фонов пресмятания . . . . .	86
4.69	Константи за интервалите на времевия ред (0 до 15) . . . . .	87
4.70	Константи за интервалите на времевия ред (30 до 43200) . . . . .	88
4.71	Описание на времеви интервал . . . . .	89
4.72	Определяне на константа за интервал по брой минути . . . . .	90
4.73	Определяне на константа за интервал по текстово описание . . . . .	91
4.74	Конструктор на изброения тип за интервал . . . . .	92
4.75	Структура на локалната таблица за котировките . . . . .	93
4.76	Структура на локалната таблица за изкуствените невронни мрежи . . . . .	94
4.77	Име и версията на базата данни . . . . .	95
4.78	Код за създаване на таблицата за котировки . . . . .	96
4.79	Код за създаване на таблицата за изкуствените невронни мрежи . . . . .	97
4.80	Код за изтриване на таблиците . . . . .	98
4.81	Конструктор и виртуални методи на спомагателния клас за базата данни . . . . .	99
4.82	Изброен тип за вида на невроните . . . . .	100
4.83	Целочислена константа за типа на неврона, която може да се ползва и като битово поле . . . . .	101
4.84	Определяне на изброимата константа по числена стойност . . . . .	102

4.85	Методи за достъп до вътрешните стойности на константите . . . . .	103
5.1	Публично хранилище за кода на сървър приложението . . . . .	105
5.2	Физическа структура на базата данни . . . . .	106
5.3	Таблица с времевите интервали за редовете . . . . .	107
5.4	Таблица за представяне на валутните двойки . . . . .	108
5.5	Таблица за представяне на типовете мрежи . . . . .	109
5.6	Таблица за представяне на типовете мрежи . . . . .	110
5.7	Таблица с опции за протичане на обучението . . . . .	111
5.8	Таблица с опции за протичане на обучението . . . . .	112
5.9	Таблица с опции за протичане на обучението . . . . .	113
5.10	Процедура за добавяне на валутна двойка . . . . .	114
5.11	Процедура за запис на екземпляр от изкуствена невронна мрежа . . . . .	115
5.12	Процедура за запис на тренировъчни примери . . . . .	116
5.13	Функция за определяне на типа мрежа . . . . .	117
5.14	Функция за определяне на типа валутна двойка . . . . .	118
5.15	Функция за определяне на най-добрата жизненост . . . . .	119
5.16	Функция за определяне на броя неврони за определен еземпляр мрежа . . . . .	120
5.17	Глобални променливи за достъп до базата данни . . . . .	121
5.18	Отваряне на връзка към базата данни . . . . .	122
5.19	Затваряне на връзката към базата данни . . . . .	123
5.20	Изпълнение на заявки към базата данни . . . . .	124
5.21	Проверка на входните данни . . . . .	125
5.22	Заявка за извличане на екземпляр по идентификатор . . . . .	126
5.23	Проверка за наличност на екземпляра . . . . .	127
5.24	Информация за невроните . . . . .	128
5.25	Информация за връзките . . . . .	129
5.26	Информация за теглата . . . . .	130
5.27	Приключване на процедурата по изпращане на отговор . . . . .	131
5.28	Заявка за зареждане на случайна мрежа . . . . .	132
5.29	Входни параметри за конкретна топология на невронна мрежа . . . . .	133
5.30	Флагове на невроните и връзките между тях . . . . .	134
5.31	Заявка за най-добра жизненост . . . . .	135
5.32	Определяне на броя неврони по идентификатор на мрежа . . . . .	136
5.33	Отговор към клиента . . . . .	137
5.34	Проверка на входните параметри . . . . .	138
5.35	Заявка за тренировъчно множество . . . . .	139
5.36	Брой примери, времеви маркери и нива на отваряне . . . . .	140
5.37	Най-ниска и най-висока постигната стойност . . . . .	141
5.38	Нива на затваряне и търгуван обем . . . . .	142
5.39	Изпращане на отговор до клиента . . . . .	143
5.40	Проверка на входните аргументи . . . . .	144
5.41	Еземпляри от типа на подадения идентификатор . . . . .	145
5.42	Пакетиране на получените стойности в JSON съобщение . . . . .	146
5.43	Проверка за екземпляри по название на валутна двойка и период . . . . .	147
5.44	Пакетиране на получените стойности в JSON съобщение . . . . .	148
5.45	Изпращане на резултата . . . . .	149
5.46	Брой тренировъчни примери по валутна информация . . . . .	150
5.47	Заявка за брой тренировъчни примери . . . . .	151
5.48	Изпращане на отговор до клиента . . . . .	152
5.49	Проверка на входните аргументи за валутна двойка, период, жизненост и брой неврони . . . . .	153
5.50	Проверка на входните аргументи за флагове на невроните, връзки и тегла между невроните . . . . .	154
5.51	Проверка за типа на всеки неврон . . . . .	155
5.52	Проверка за стойността на всяко тегло . . . . .	156

5.53	Проверка за стойността на всяка връзка . . . . .	157
5.54	Съхраняване на екземпляр изкуствена невронна мрежа . . . . .	158
5.55	Проверка на жизнеността преди съхраняване на информацията . . . . .	159
5.56	Проверка на аргументите за валутна двойка, период, брой примери и времеви маркери . . . . .	160
5.57	Проверка на аргументите за отваряне, най-ниска стойност, най-висока стойност, затваряне и изтъргуван обем . . . . .	161
5.58	Проверка на стойностите за времеви маркери и нива на отваряне . . . . .	162
5.59	Проверка на стойностите за най-ниско и най-високо постигнато ниво . . . . .	163
5.60	Проверка на стойностите за затваряне и изтъргуван обем . . . . .	164
5.61	Съхраняване на тренировъчно множество . . . . .	165
6.1	Инициализация на вътрешните променливи . . . . .	167
6.2	Зареждане на данните с помощта на HTTP комуникация . . . . .	168
6.3	HTTP комуникационен файл . . . . .	169
6.4	Конструктор на спомагателния клас . . . . .	170
6.5	Функция за зареждане на случайно избрана от сървъра мрежа . . . . .	171
6.6	Зареждане на тренировъчно множество . . . . .	172
6.7	Функция за запазване на допълнително обучавана мрежа . . . . .	173
6.8	Подготвяне на POST заявка . . . . .	174
6.9	Изпълнение на POST заявката . . . . .	175
6.10	Ключови стойности за описване на мрежата . . . . .	176
6.11	Ключови стойности за описване на тренировъчното множество . . . . .	177
6.12	Стойности за времевия ред и невроните . . . . .	178
6.13	Стойности за теглата и силата на връзките . . . . .	179
6.14	Стойности за време, отваряне и най-ниска постигната стойност . . . . .	180
6.15	Стойности за най-висока постигната стойност, затваряне и изтъргуван обем . . . . .	181
6.16	Записване на данните в глобалната структура . . . . .	182
7.1	Единична стъпка за обучение на мрежата . . . . .	184
7.2	Класическо обратно разпространение на грешката . . . . .	185
7.3	Бързо разпространение на грешката . . . . .	186
7.4	Мащабирано съединение на градиента . . . . .	187
7.5	Корекция на теглата по Манхатън правило . . . . .	188
7.6	Превключване между обратно разпространение на грешката и генетичен алгоритъм . . . . .	189
7.7	Параметри на генетичния алгоритъм . . . . .	190
7.8	Извличане на теглата от мрежата . . . . .	191
7.9	Начална популация . . . . .	192
7.10	Изпълнение на генетичния алгоритъм . . . . .	193
7.11	Обновяване на теглата в мрежата . . . . .	194
7.12	Кодиране на теглата в хромозомите . . . . .	195
7.13	Конструиране по зададен масив . . . . .	196
7.14	Конструиране по зададен списък . . . . .	197
7.15	Конструиране по зададен списък с възможност за дълбоко копиране . . . . .	198
7.16	Зареждане на стойностите като тегла в мрежата . . . . .	199
7.17	Изпълнение на правия пас . . . . .	200
7.18	Условие за валидност на хромозомата . . . . .	201
7.19	Конструиране на нова хромозома с фиксирана дължина . . . . .	202
7.20	Функция за извеждане на стойностите извън хромозомата . . . . .	203
7.21	Клас извършващ мутацията . . . . .	204
7.22	Контрол на входните данни . . . . .	205
7.23	Случайна промяна на всички стойности в хромозомата . . . . .	206

## Азбучен указател

дарени изчислителни ресурси, 5, 11, 15  
екстраполация, 10  
евристични алгоритми, 6  
евристики, 6  
финансови времеви редове, 9  
генетични алгоритми, 6, 8, 13  
глобална оптимизационна евристика, 8  
грид изчисленията, 4  
хранилища за програмен код, 16  
изчисления в разпределена среда, 1, 5, 11, 15, 18, 104, 207  
изкуствени невронни мрежи, 6, 7, 10, 13, 26, 56–58, 60, 67  
конкурентни пресмятания, 4  
кръстосване в генетичен алгоритъм, 8  
мобилна среда за разпределени изчисления, 5  
мобилни устройства, 1  
мутация в генетичен алгоритъм, 8  
обратно разпространение на грешката, 7, 58  
паралелни пресмятания, 4  
последователното програмиране, 3  
развойни средства, 13  
селекция в генетичен алгоритъм, 8  
системи за подпомагане вземането на решения, 10  
софтуерни лицензи, 16  
супер компютърни изчисления, 4, 6  
времеви редове, 6, 9, 10  
жизнена функция, 8

Настоящото учебно помагало съдържа серия от примери по дисциплините „Java за напреднали“, която се изучава в магистърската програма „Софтуерни технологии в Интернет“ на „Нов български университет“ и „Мобилни приложения“, която се изучава в бакалавърската програма „Компютърни науки“ на „Университета по библиотекознание и информационни технологии“. Насочено е към аудитория със сериозни познания в областта на програмирането.



Помагалото е съставено от практически примери, обхващащи цялостния цикъл за разработка на клиент-сървър базирани системи, подходящи за реализацията на изчисления в разпределена среда. Структурата на учебното помагало позволява отделни части от него да послужат при разработването на курсови задачи и/или дипломни работи.



Свободният достъп до помагалото дава възможност то да бъде използвано в учебния процес на курсове с подобно съдържание и на други учебни заведения. Изложеният материал дава възможност за допълване на съществуващите магистърски програми, примерно в областта на изкуствения интелект.



Посочените в библиографията литературни източници са предимно със справочен характер, но дават възможност на заинтересуваните читатели да разширят познанията си в засегнатите области.