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, интересуващи 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

Вера Ангелова – ИИКТ-БАН
e-mail: vangelova@iit.bas.bg

Пенчо Маринов – ИИКТ-БАН
e-mail: pencho@bas.bg

eISSN: 2367-8666

Теми

Съдържание

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, съставящи 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 уеб 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 PHP 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.

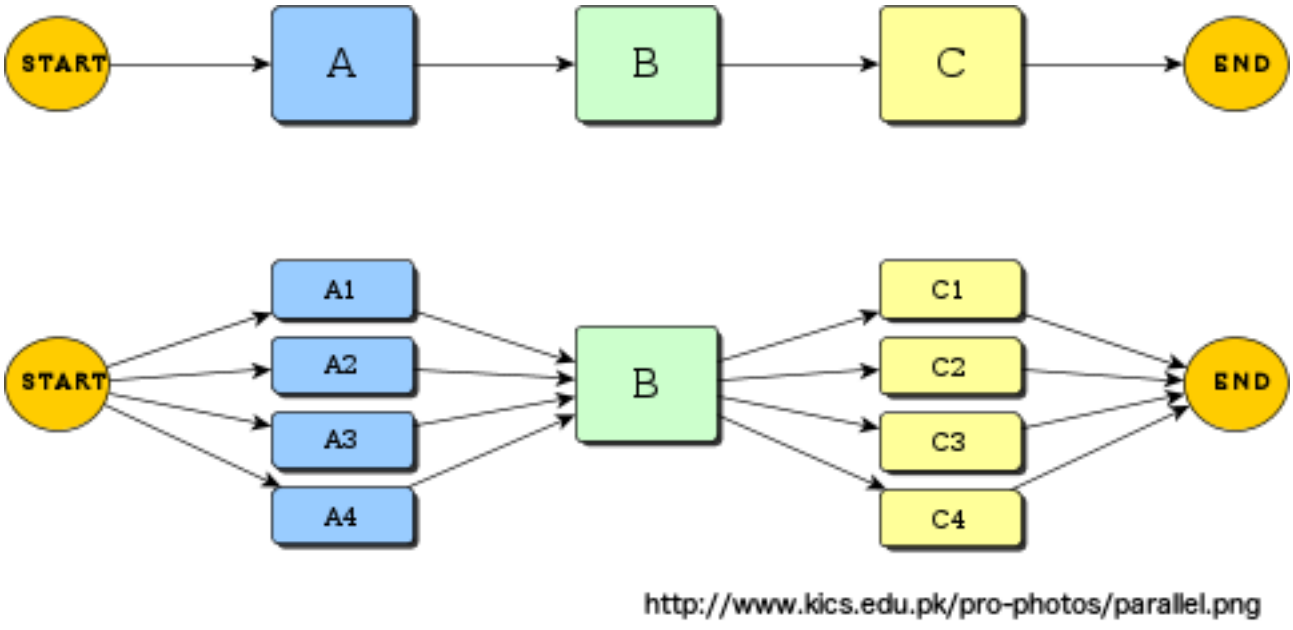
Глава 1

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

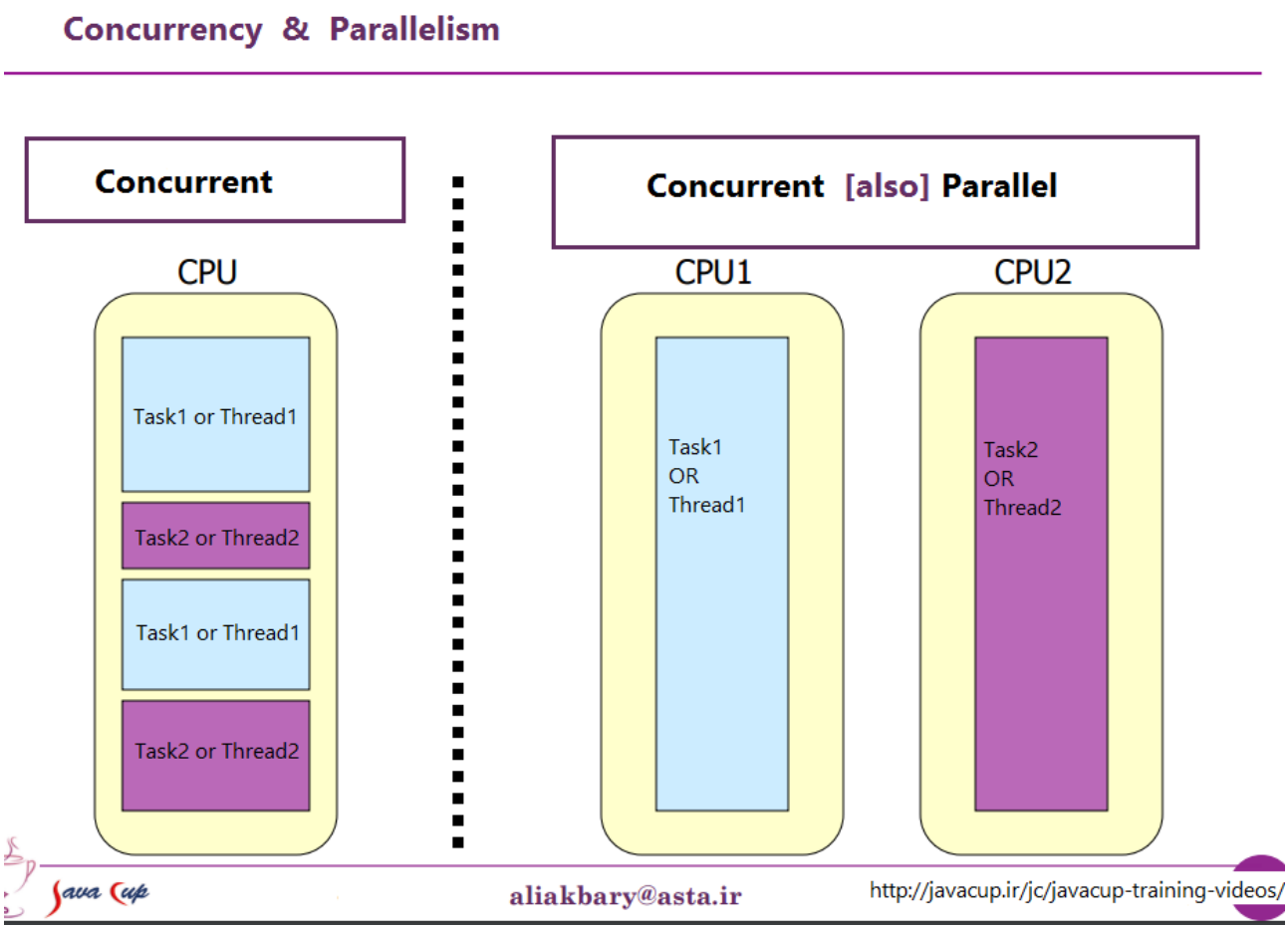
При последователното програмиране всяка изчислителна инструкция следва всички предходни. В зората на изчислителната техника пресмятанията са извършвани по този начин. Дори в наши дни значителна част от алгоритмите се изпълняват само последователно, тъй като входните данни за всяка инструкция зависят 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).



Фигура 1.2: Сравнение между конкуретни 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). Съществото 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

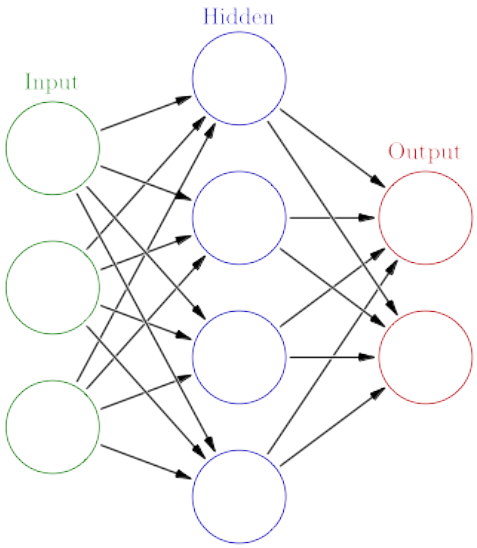
Евристики за прогнозиране

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).

$$z_j = \frac{1}{1 + e^{-y_j}} \tag{2.1}$$

$$z_j = \frac{e^{2y_j} - 1}{e^{2y_j} + 1} \tag{2.2}$$

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.

$$y_j = \sum_{i=1} x_i * w_{ij} \tag{2.3}$$

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.

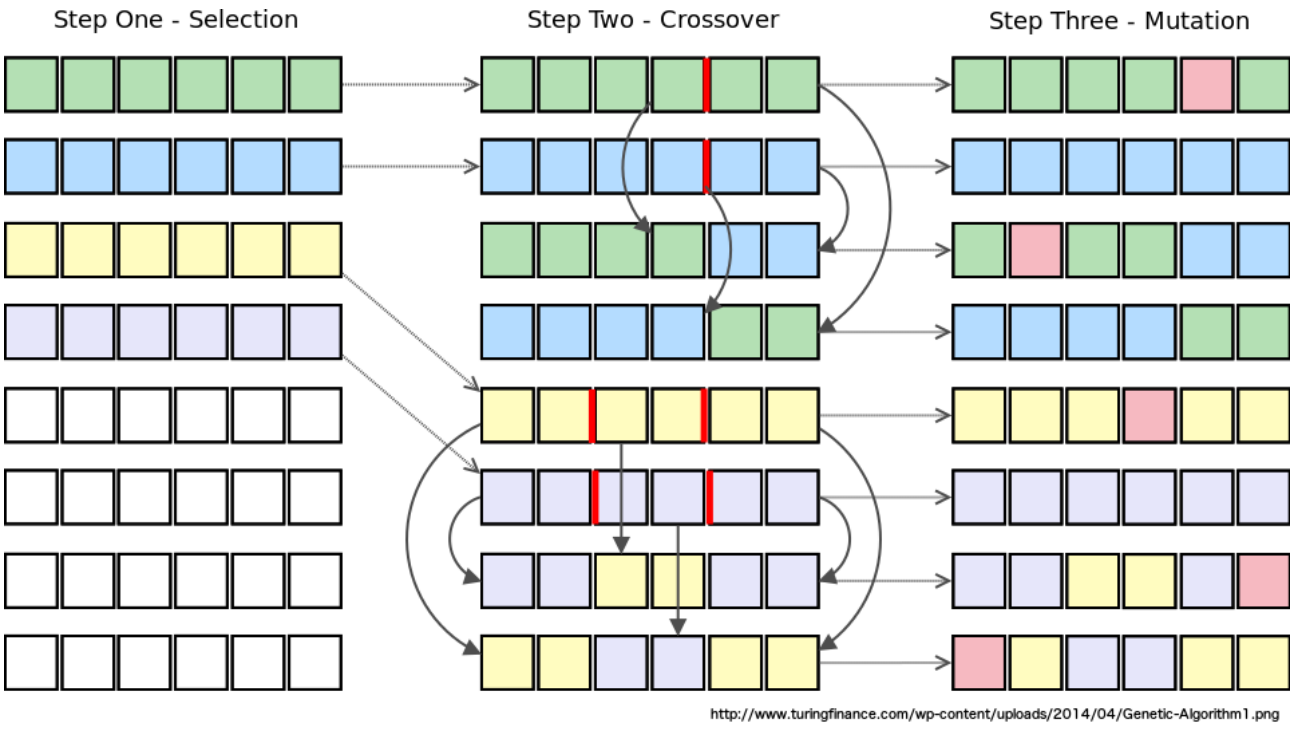
2.2 Генетични алгоритми

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 подобрение 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

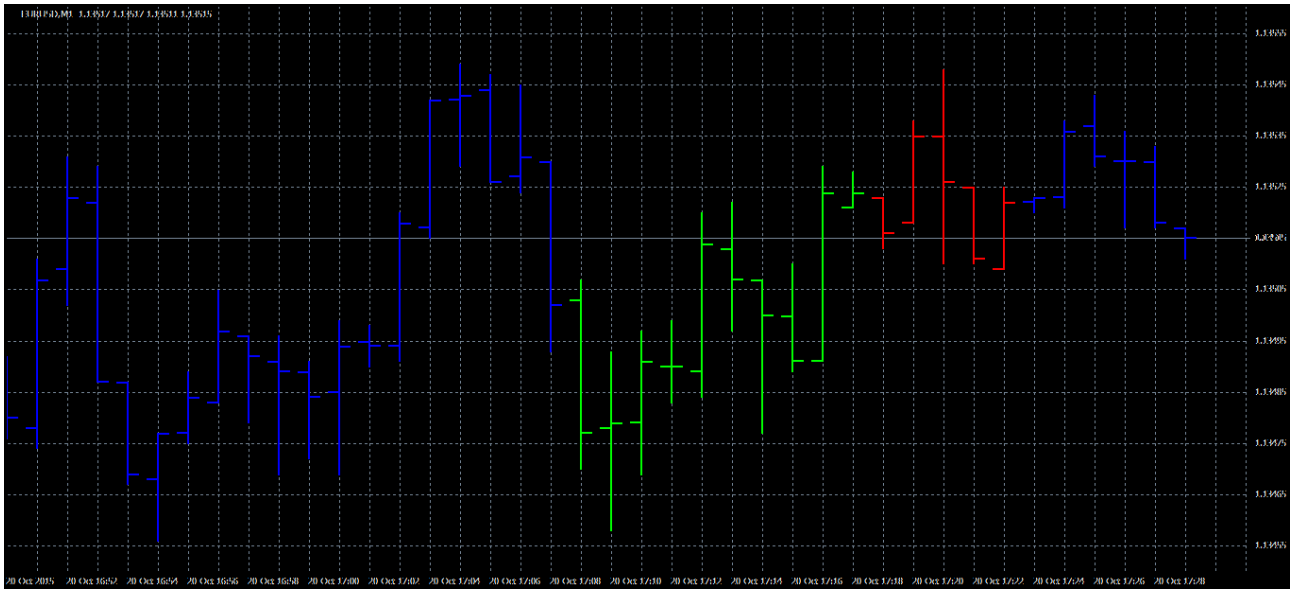
това се наблюдава при естествените биологични видове.

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, характеризиращи времевия 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 стройност 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 чертички 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

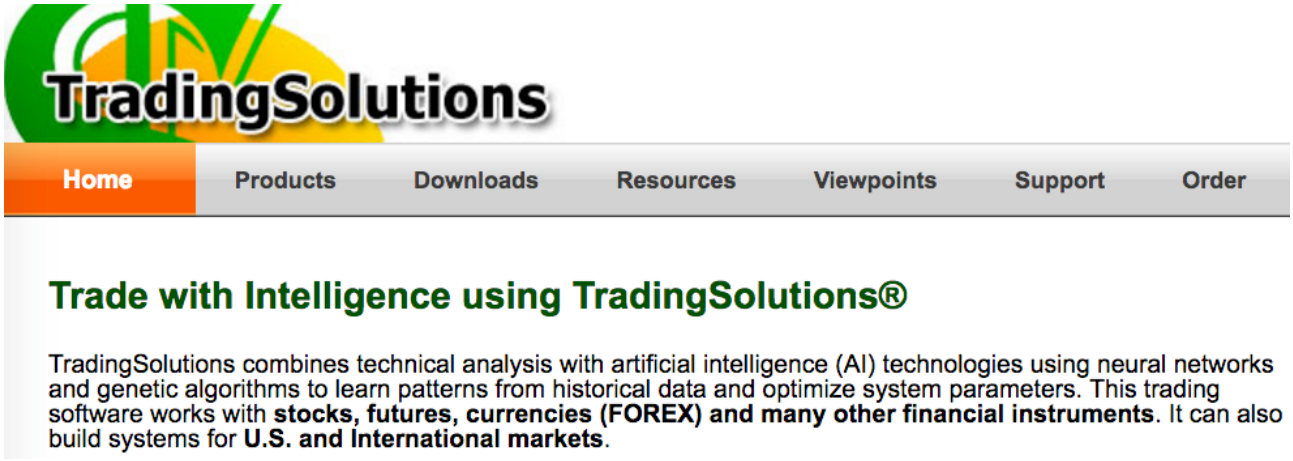
невронни мрежи. Мащабираната информация от миналия период (зелено на графиката) се подава към входния слой на изкуствената невронна мрежа, а прогнозата се получава в изходния слой на изкуствената невронна мрежа и се подлага на обратно мащабиране (червеното на графиката). Така декомпозиран времевият ред 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 изграждат функционална зависимост между входни и изходни данни ги прави идеален кандидат за прогнозираща система. Ако се приеме, че измерените стойности във времевия 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 й 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 времевия ред трябва да бъде мащабирана в съответния работен интервал на мрежата. Стойностите на времевия 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 времевия 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 прогнозират цените на финансови инструменти винаги е привличала не само индустрията, но и научните среди. Едно от най-забележителните постижения в тази насока е проектът MoneyBee (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 пресмятанията 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 релационна 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“.

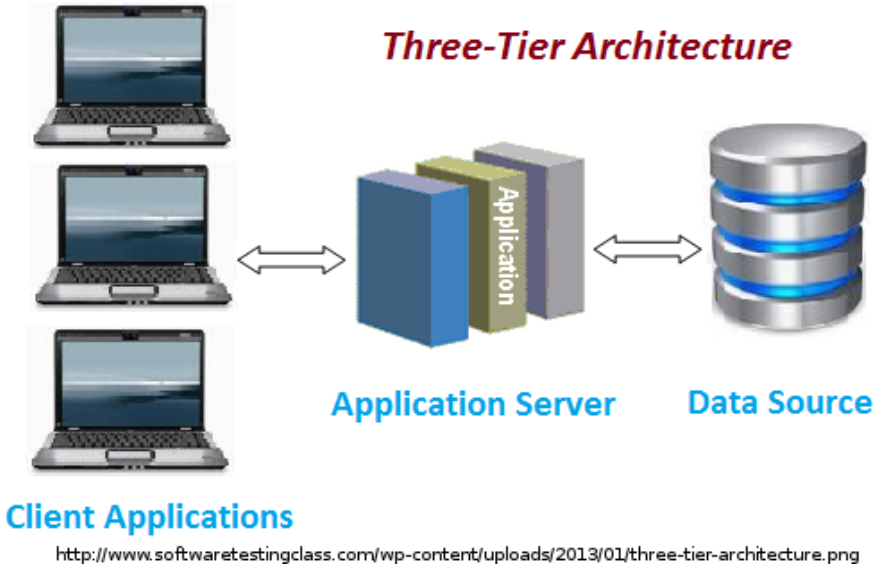
3.1.3 За комуникация между сървъра и клиента

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 поголяма 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-съществено 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.

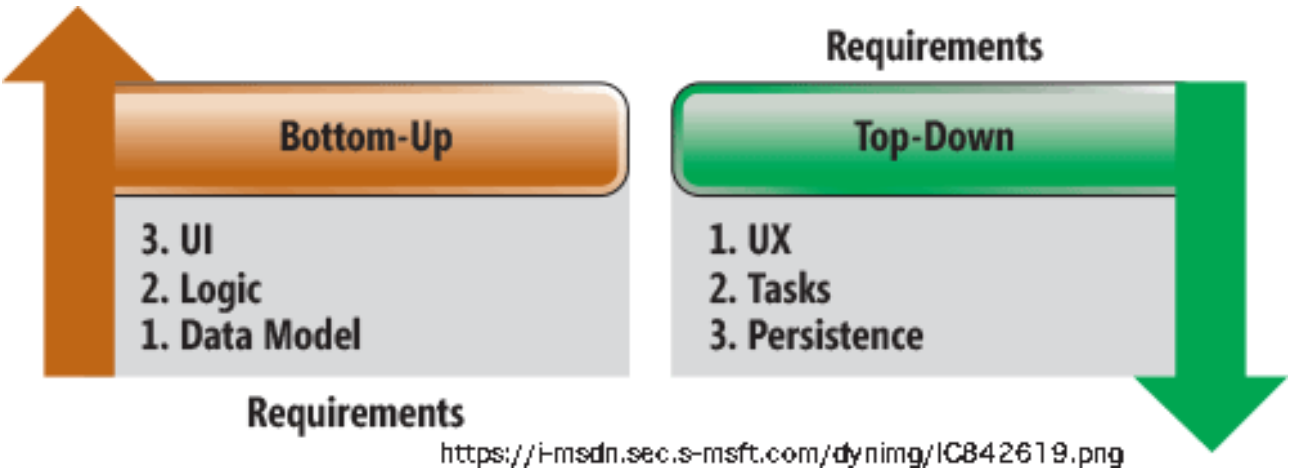


Фигура 3.1: Уеб базирана трислойна софтуерна архитектура

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.

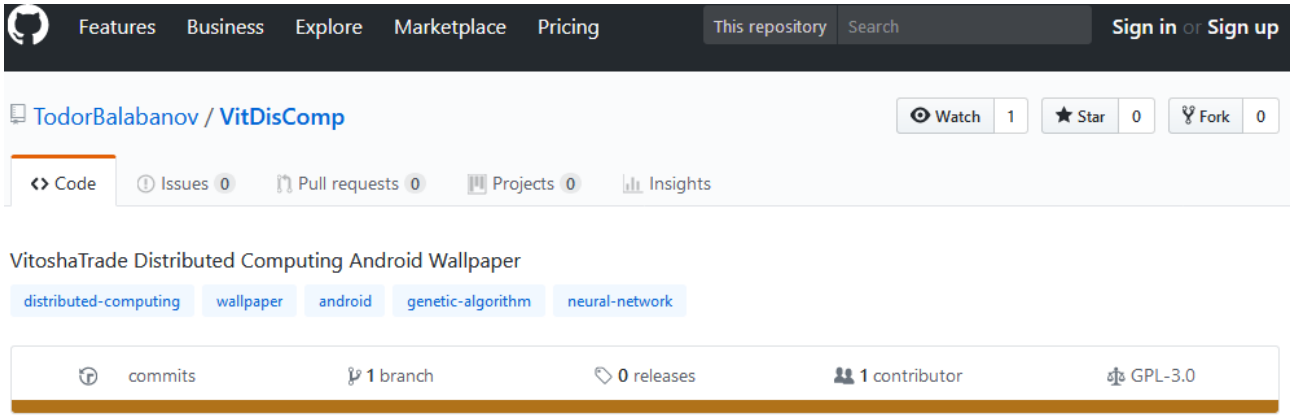
3.3.1 Лиценз

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

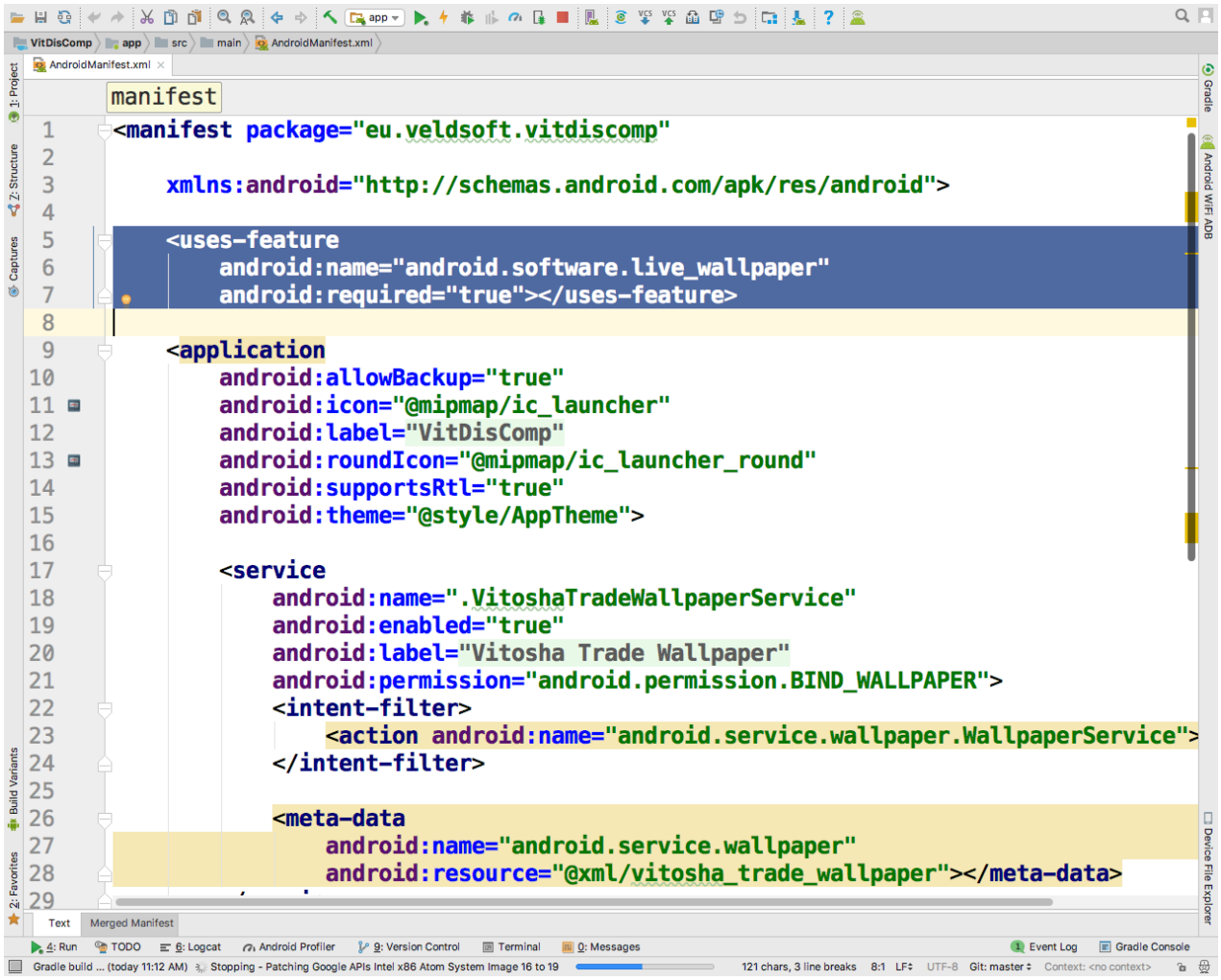
Глава 4

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.

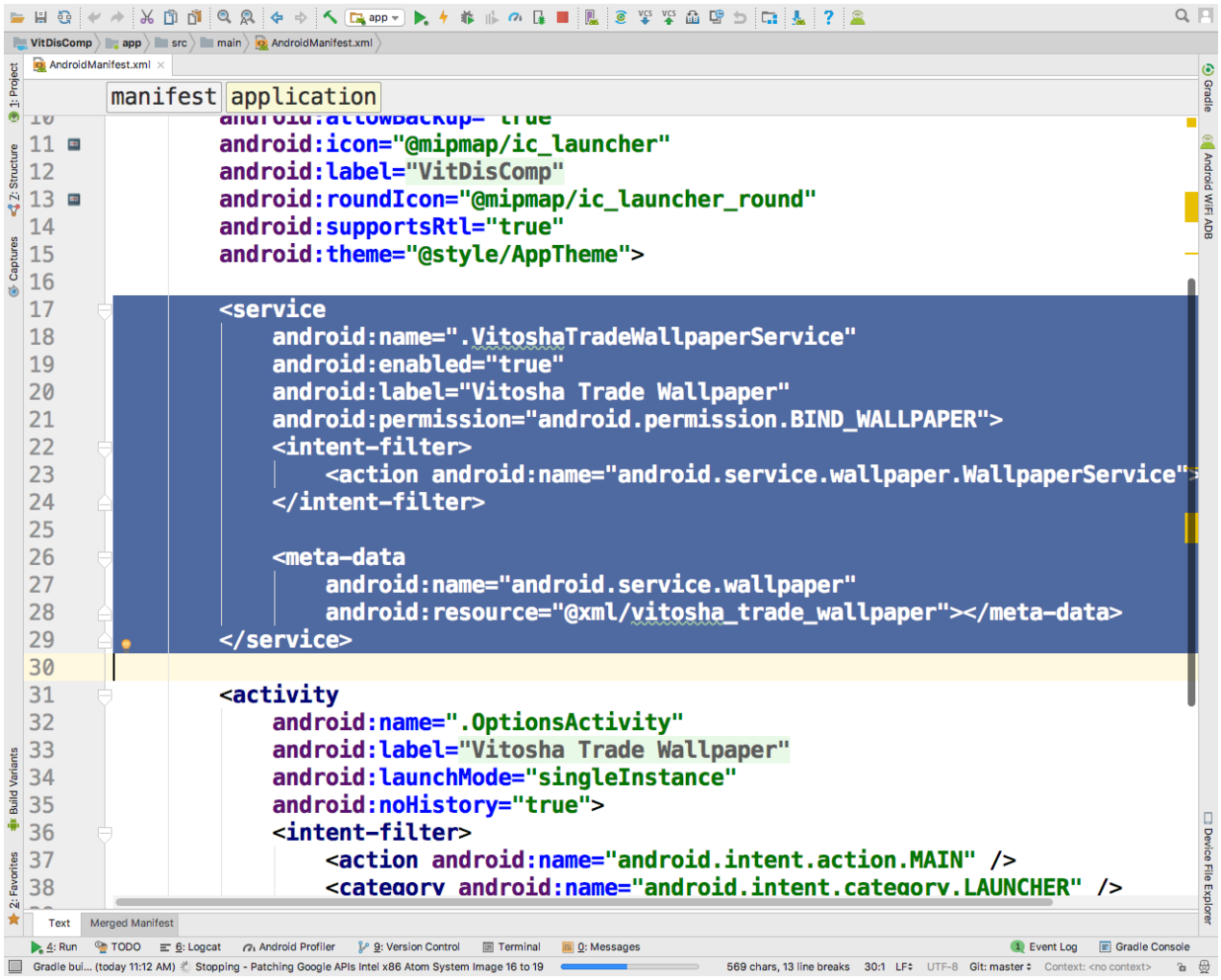
4.1 Манифест файл на мобилното приложение

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.



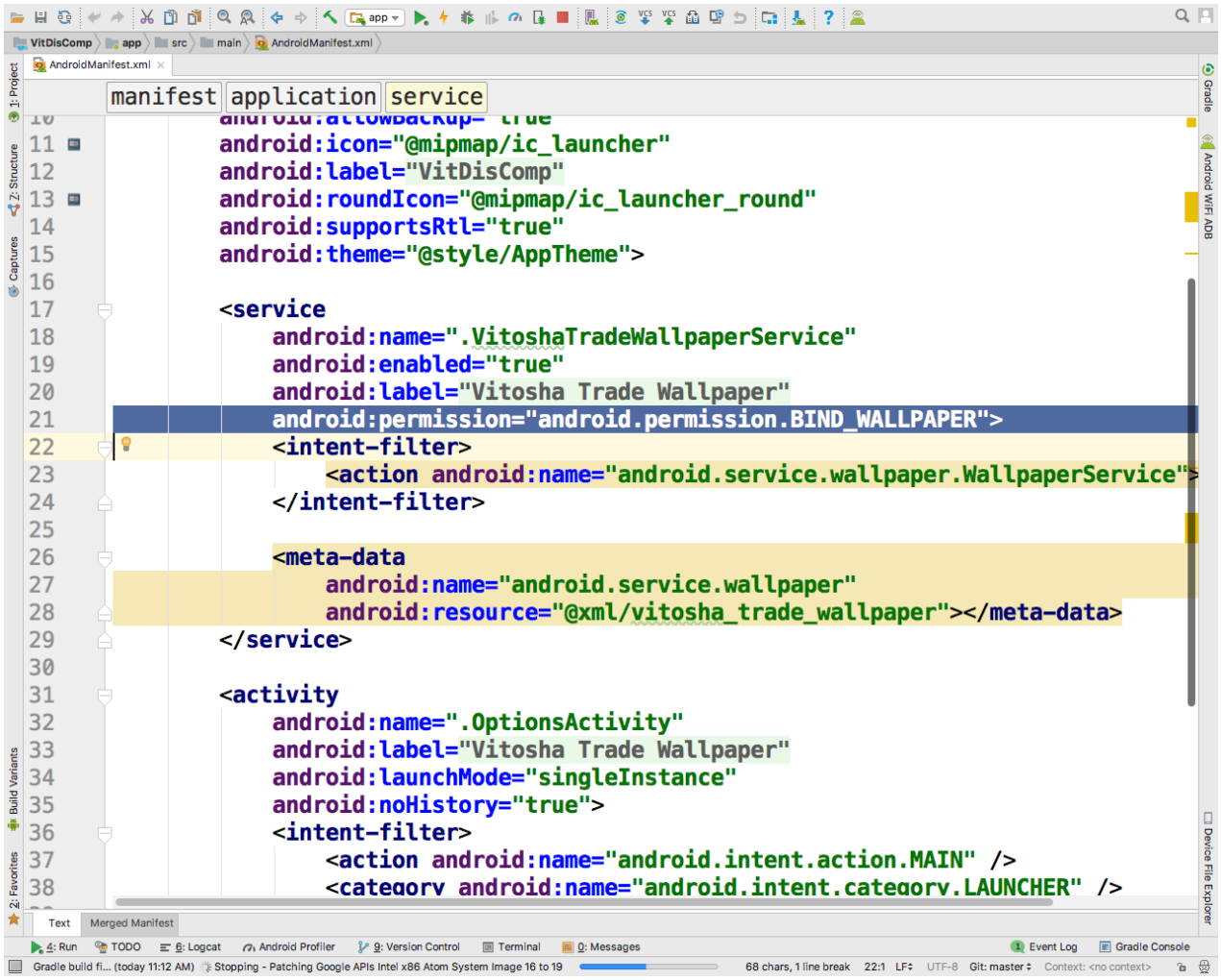
Фигура 4.1: Определяне на приложението като активен тапет

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.



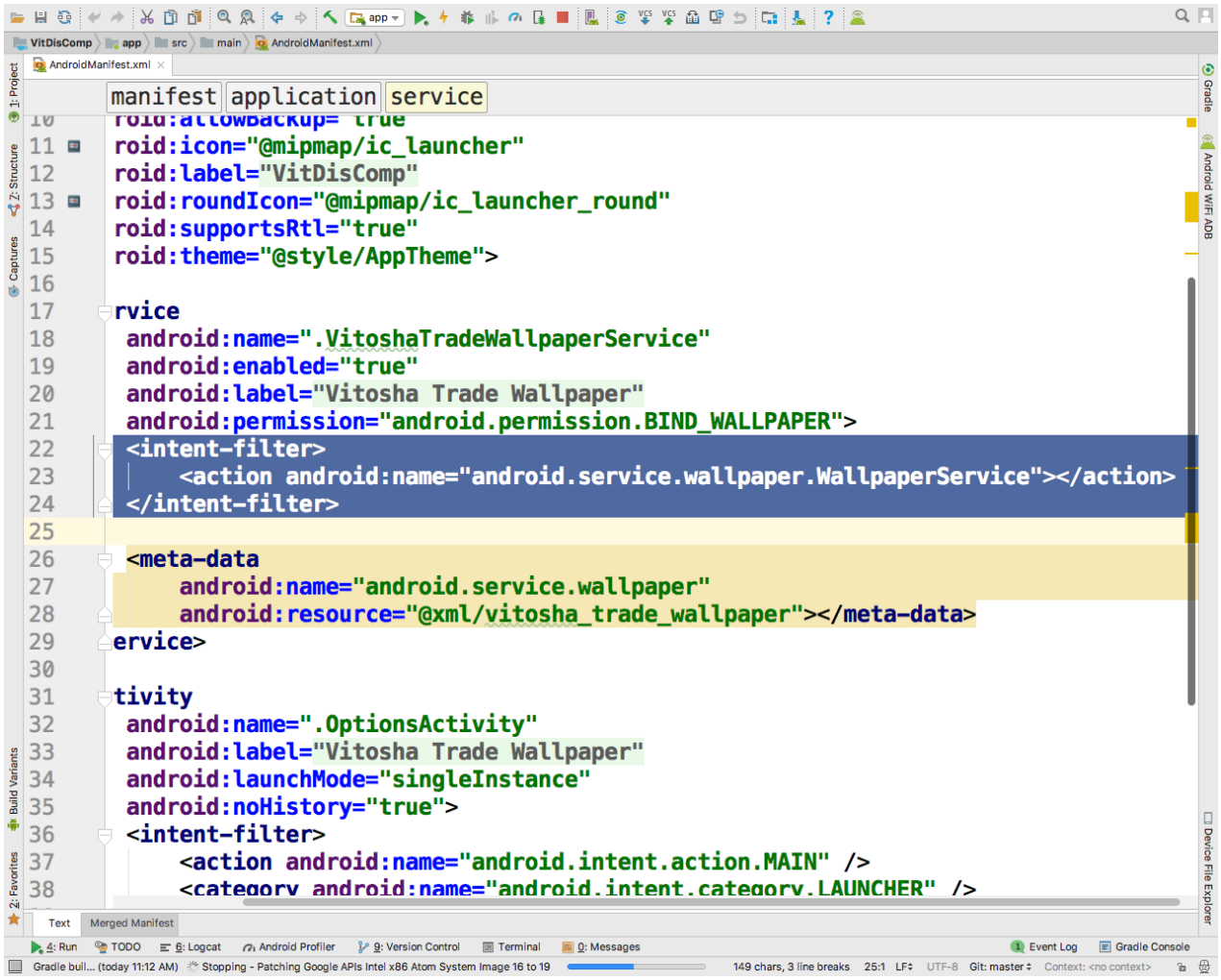
Фигура 4.2: Модул от тип “услуга”

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).



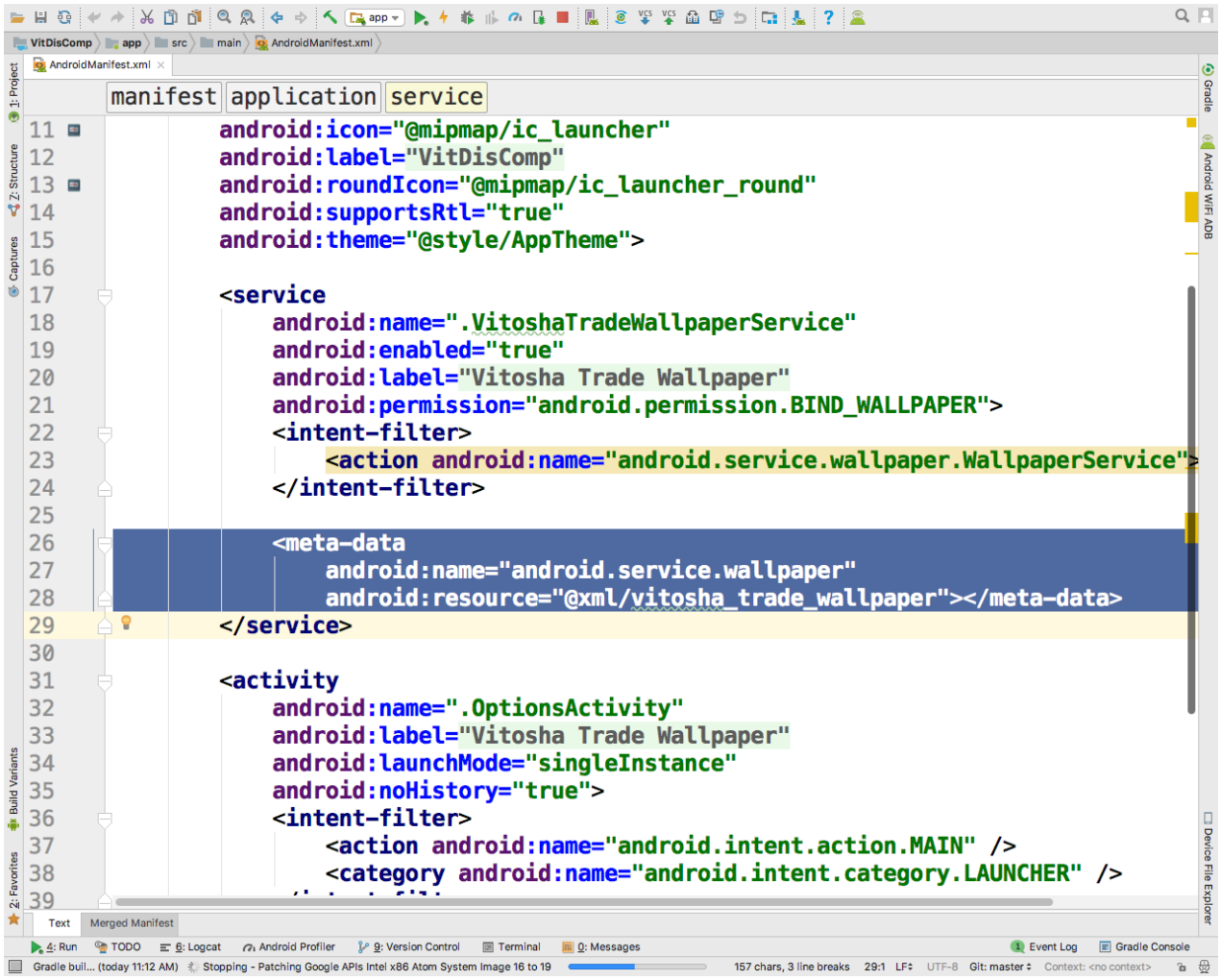
Фигура 4.3: Флагове за достъп до ресурса активен тапет

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).



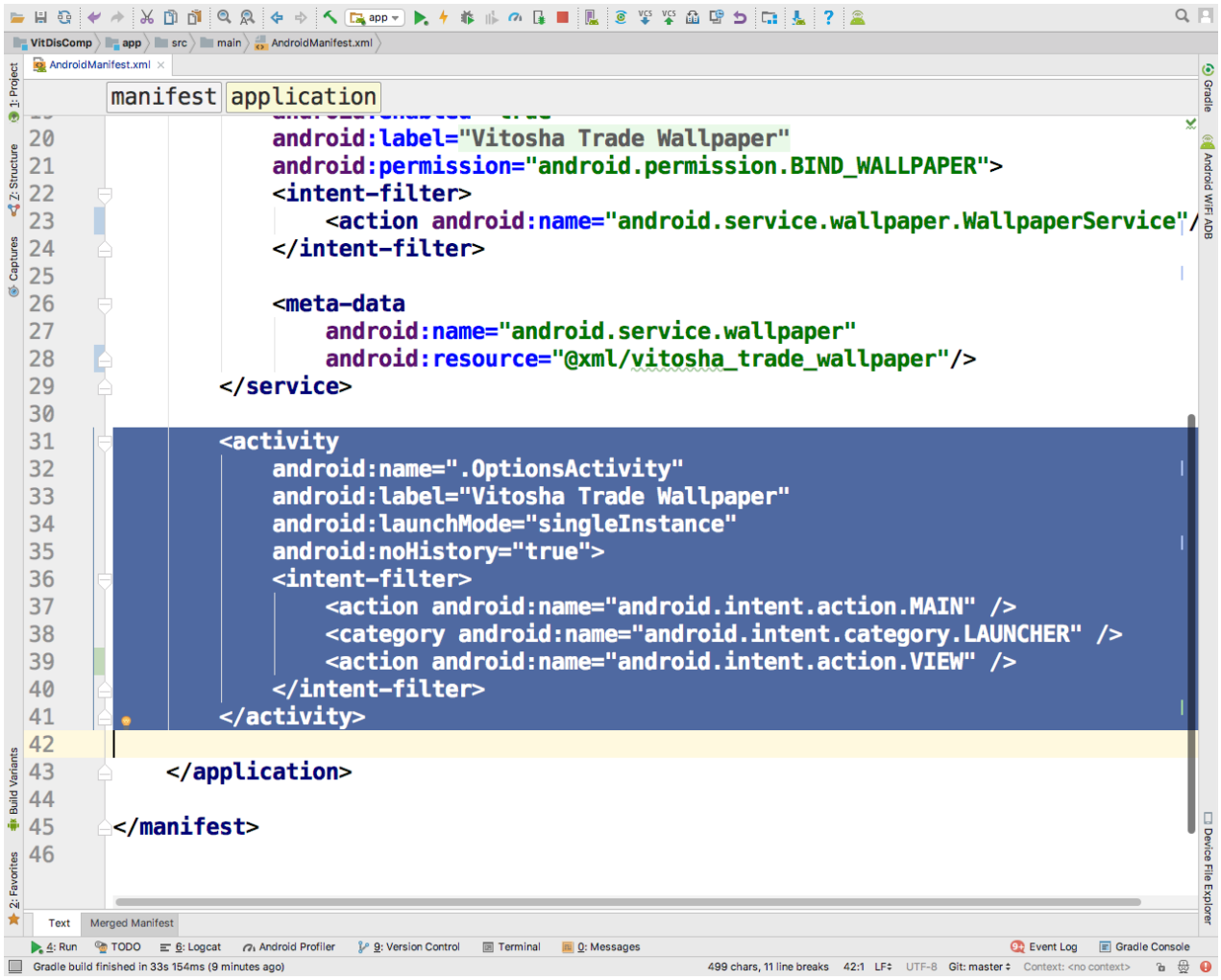
Фигура 4.4: Регистрация на услугата за прослушване на съобщения от операционната система.

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).



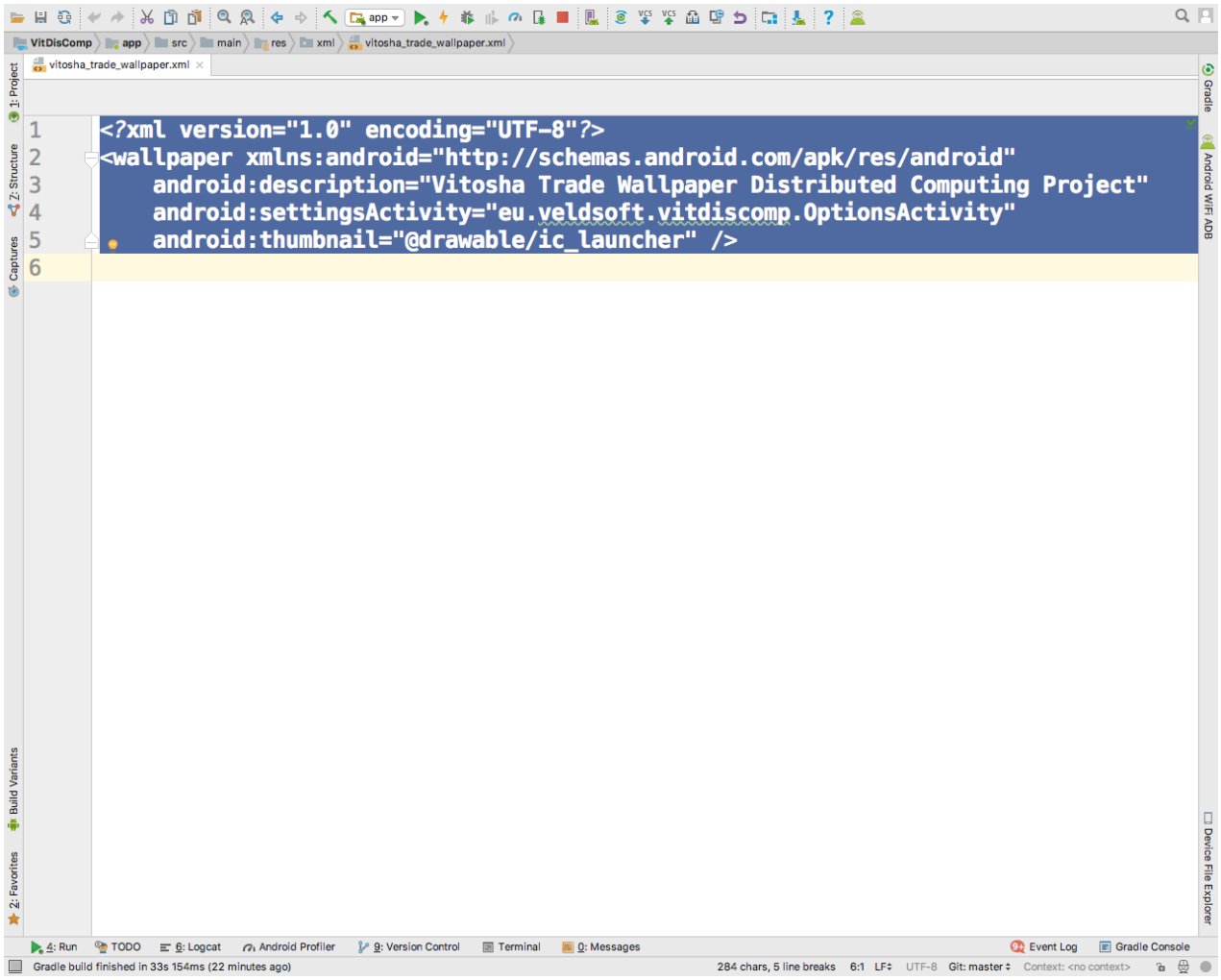
Фигура 4.5: Референция към описателния файл на активния тапет

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).

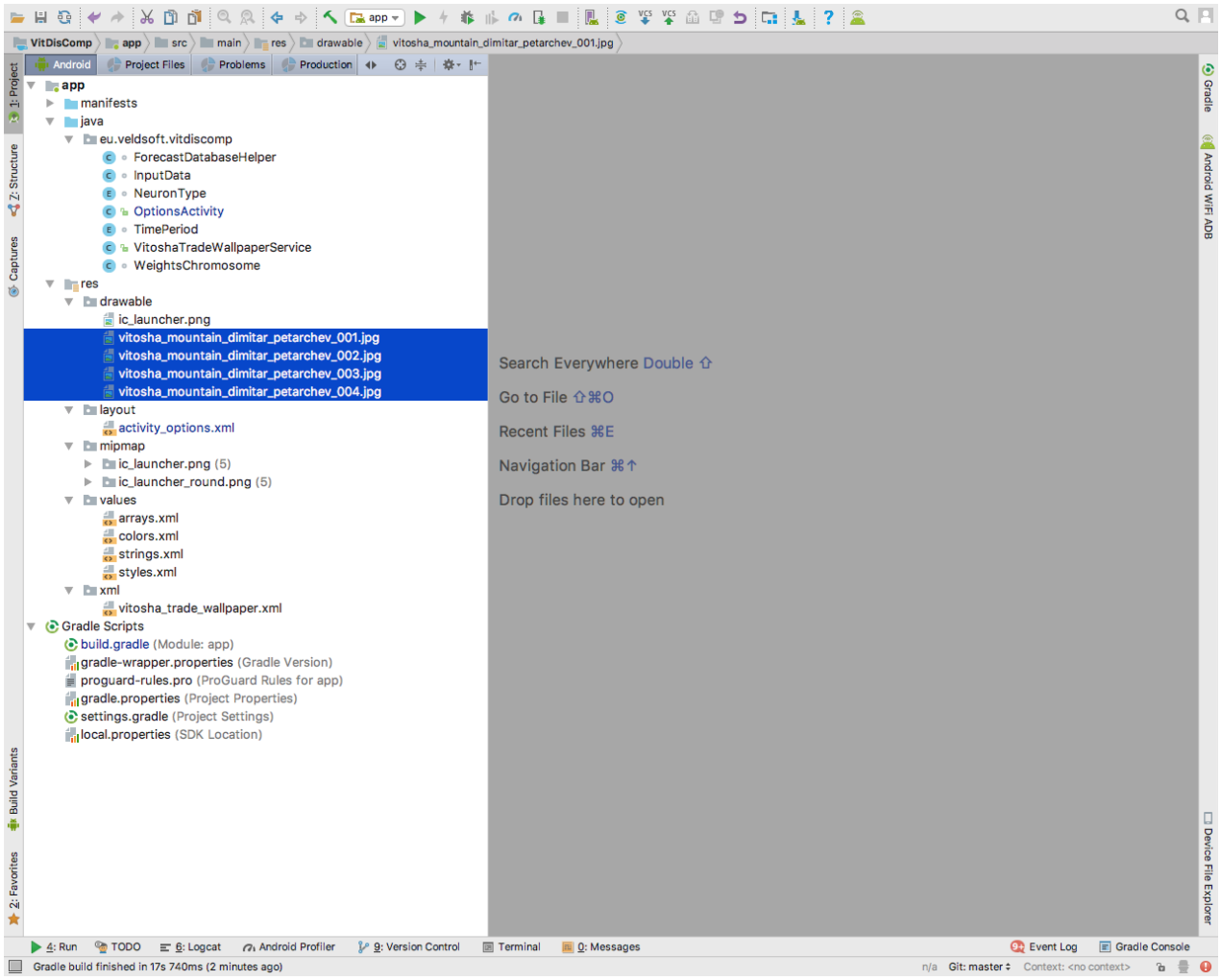


Фигура 4.7: XML описателен файл на тапета

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).

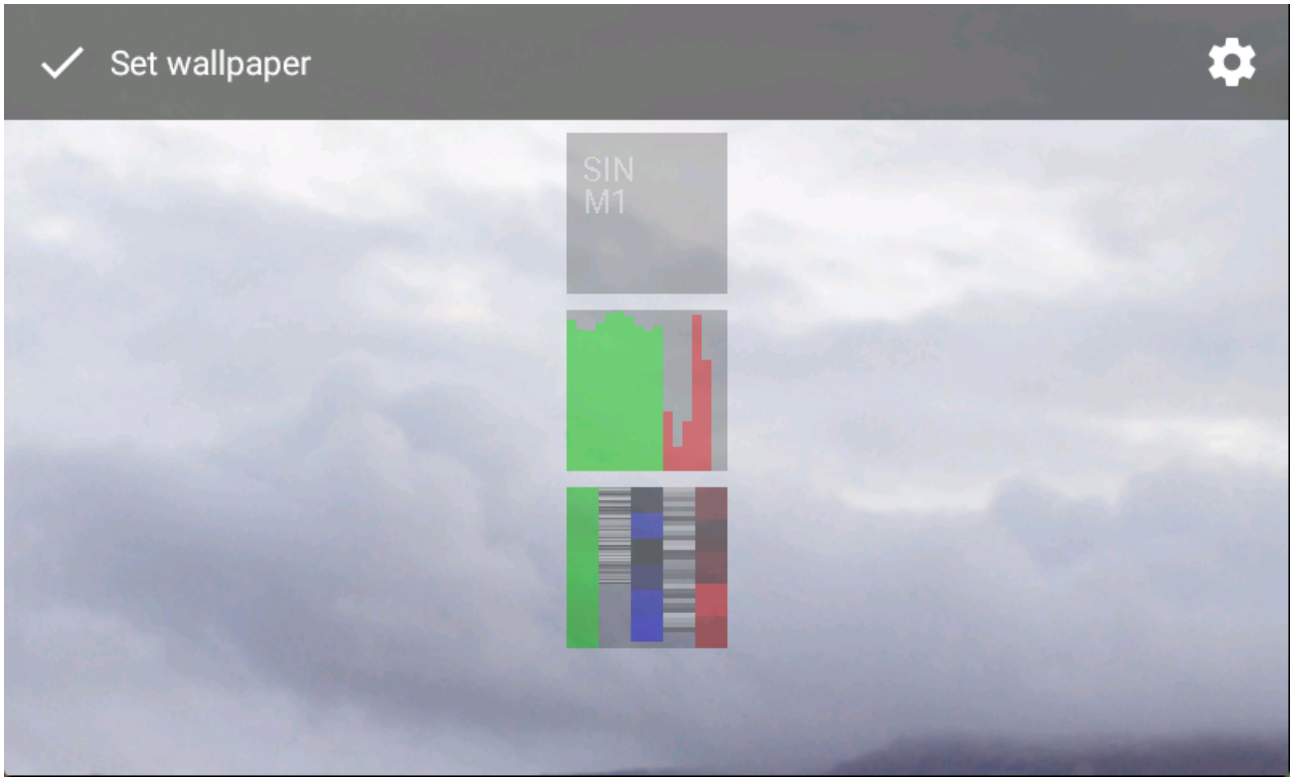
4.2 Екран с настройки

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.



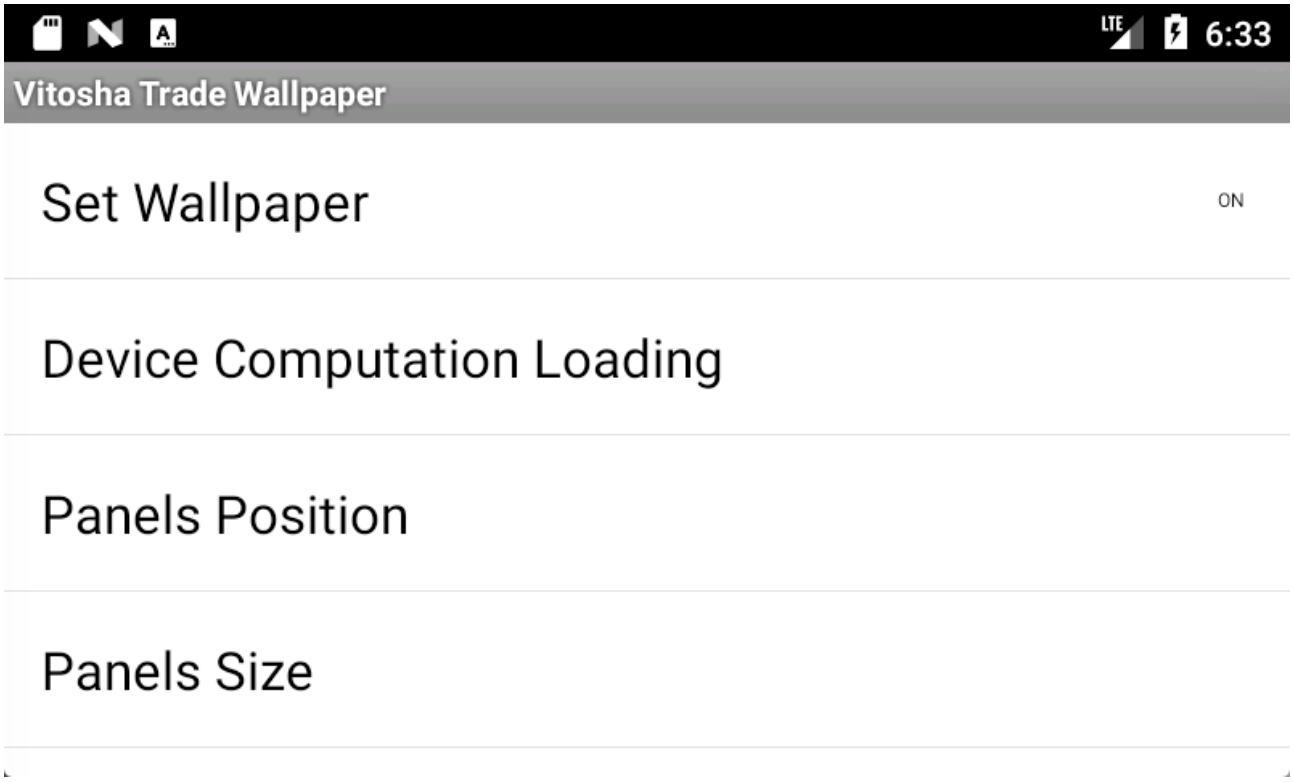
Фигура 4.8: Графични файлове, съдържащи фотографии на планината Витоша до град София, България

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

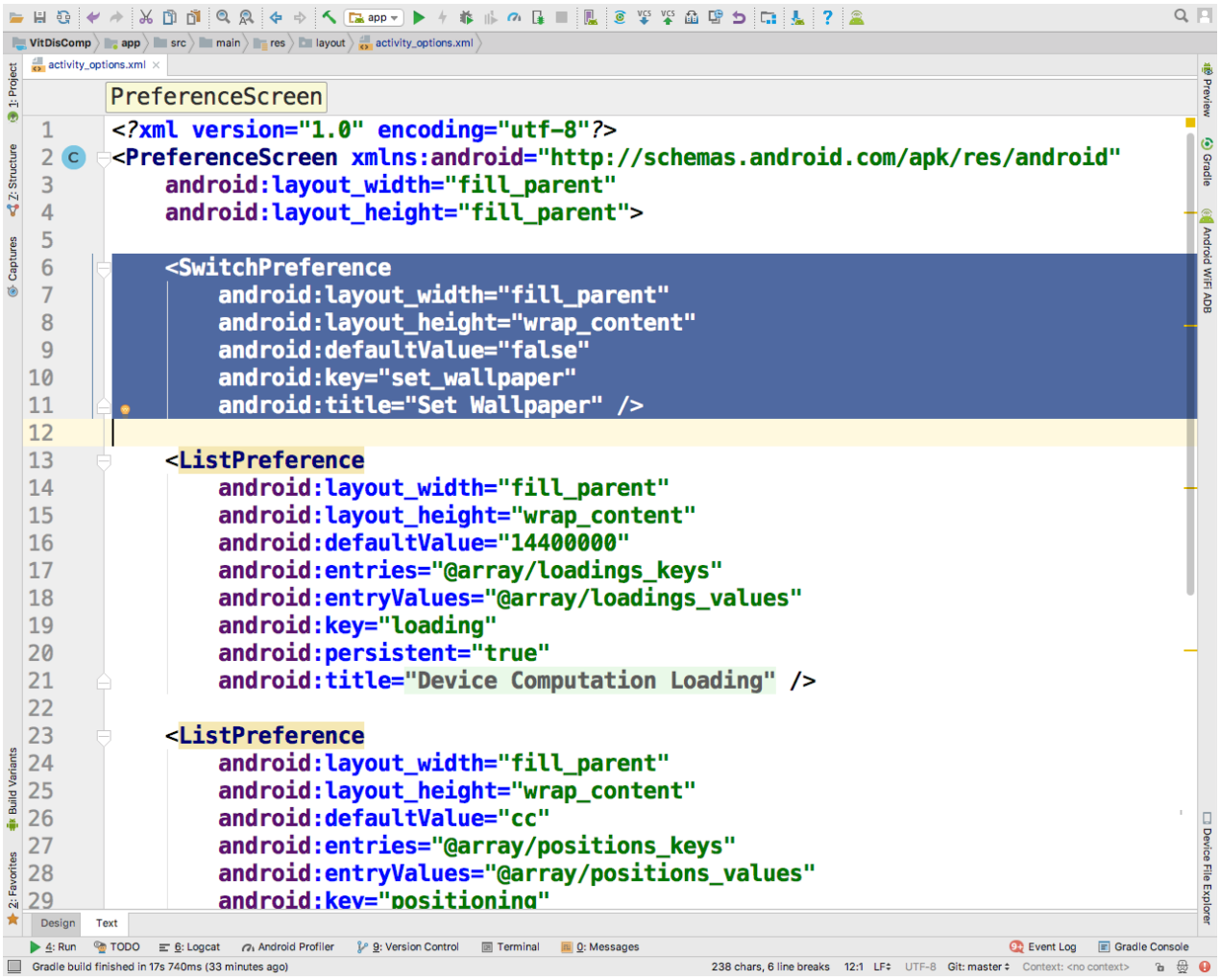
В настройките за активния тапет са предоставени възможности за управление на позицията и размерът за трите визуални области. Също така са добавени първоначални настройки за натоварване на устройството и дали активният тапет да бъде включен (Фиг. 4.10).



Фигура 4.10: Първоначален набор от настройки

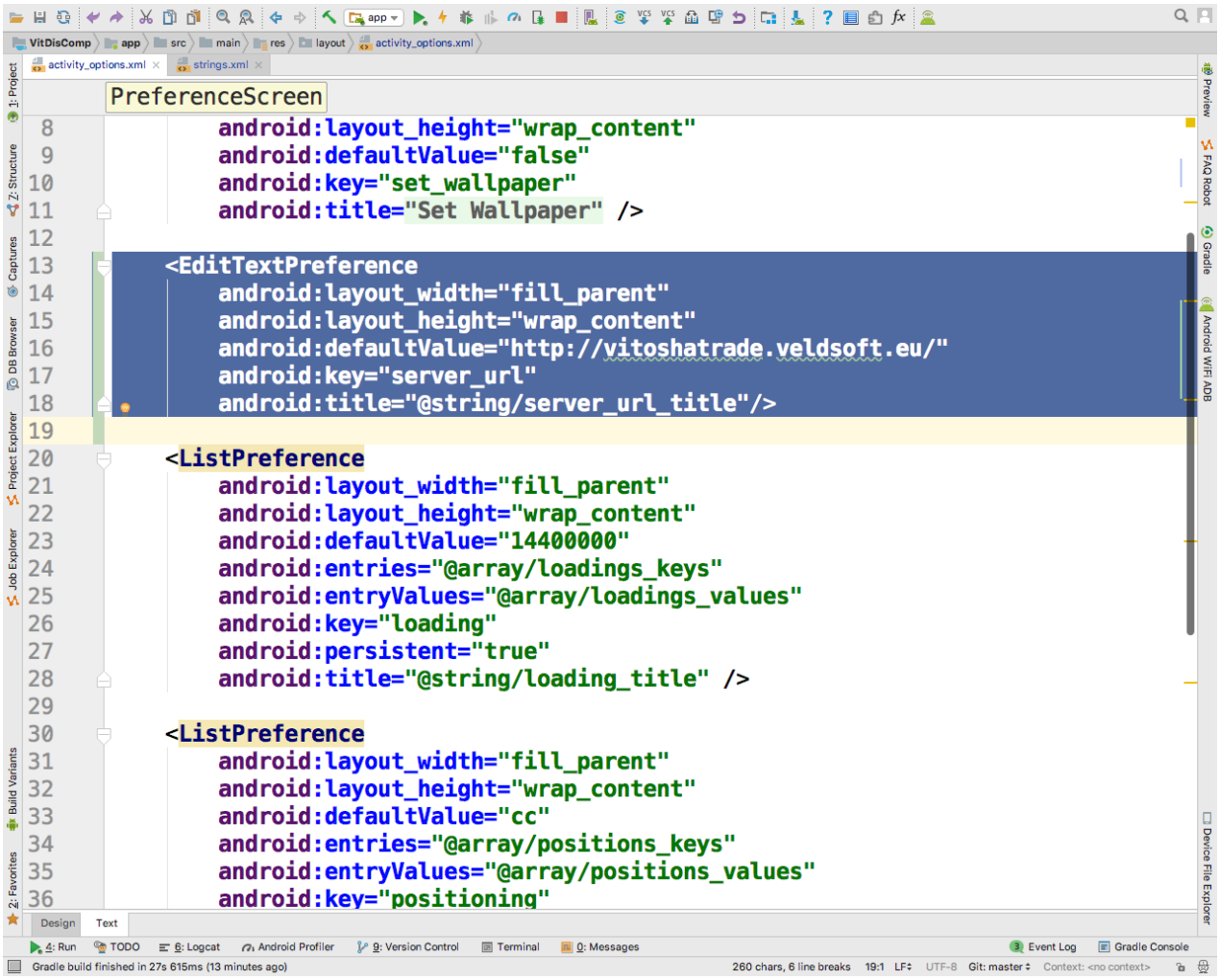
Всеки прозорец в Android се описва със свой файл за разполагане (layout) и файл с Java програмен код. Описателният файл за графичен потребителски интерфейс използва XML и много наподобява съставянето на уеб страница. Когато се изготвя екран за настройки един от най-полезните инструменти в операционната система Android са споделените преференции (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



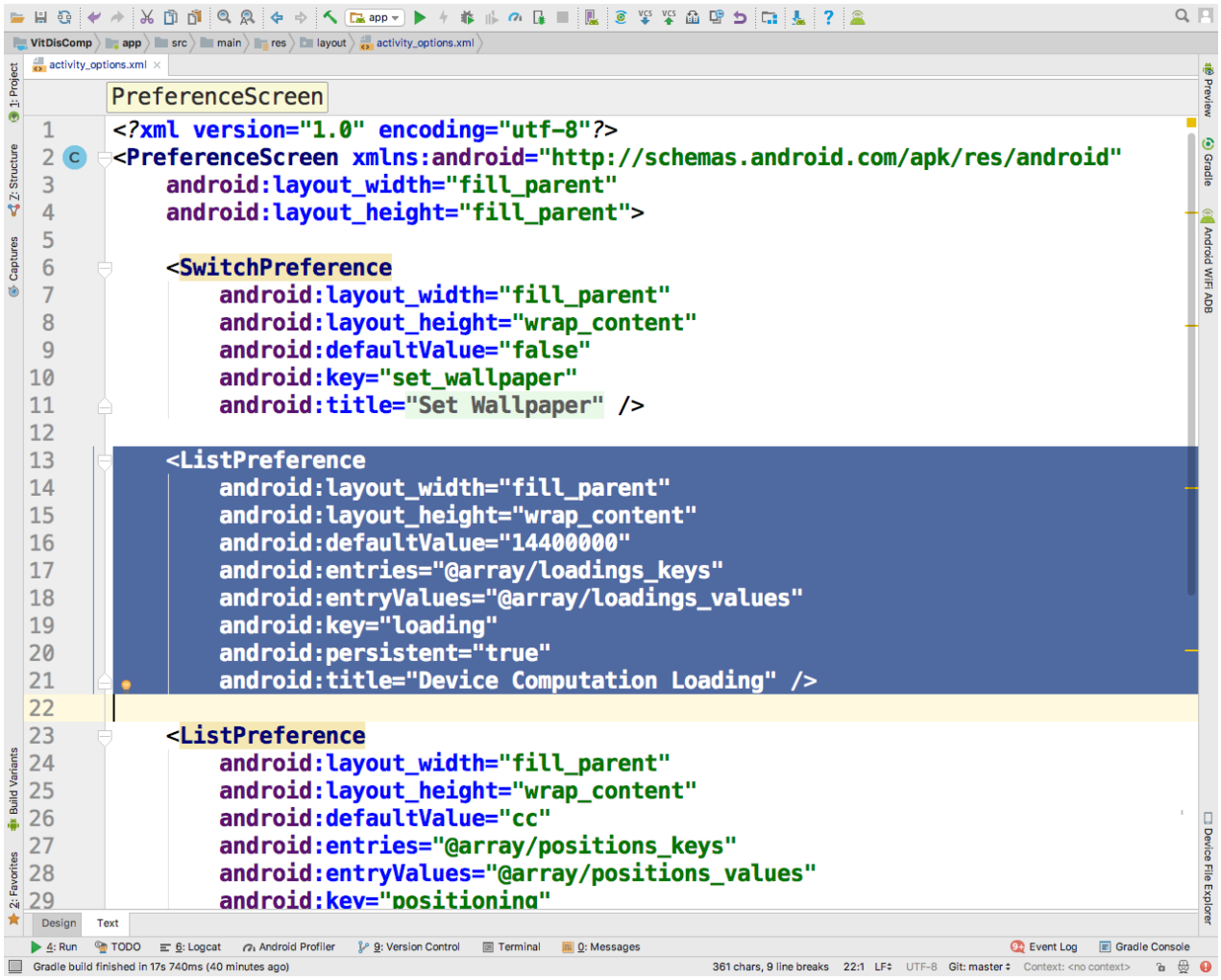
Фигура 4.11: Визуален компонент за включване и изключване на тапета

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



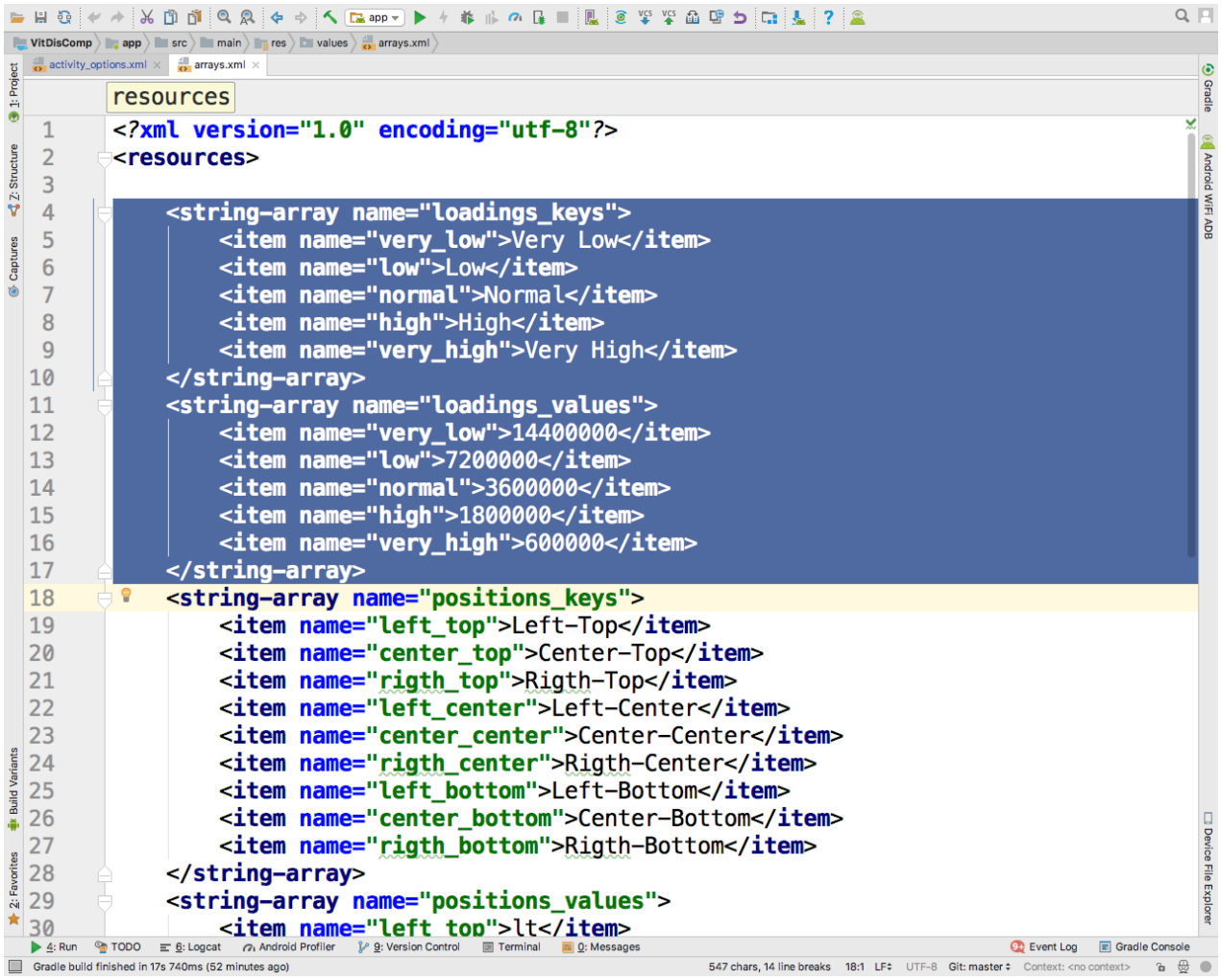
Фигура 4.12: Визуален компонент за посочване на сървър URL

Тъй като мобилното приложение тегли информацията за времевите редове от отдалечен уеб 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.



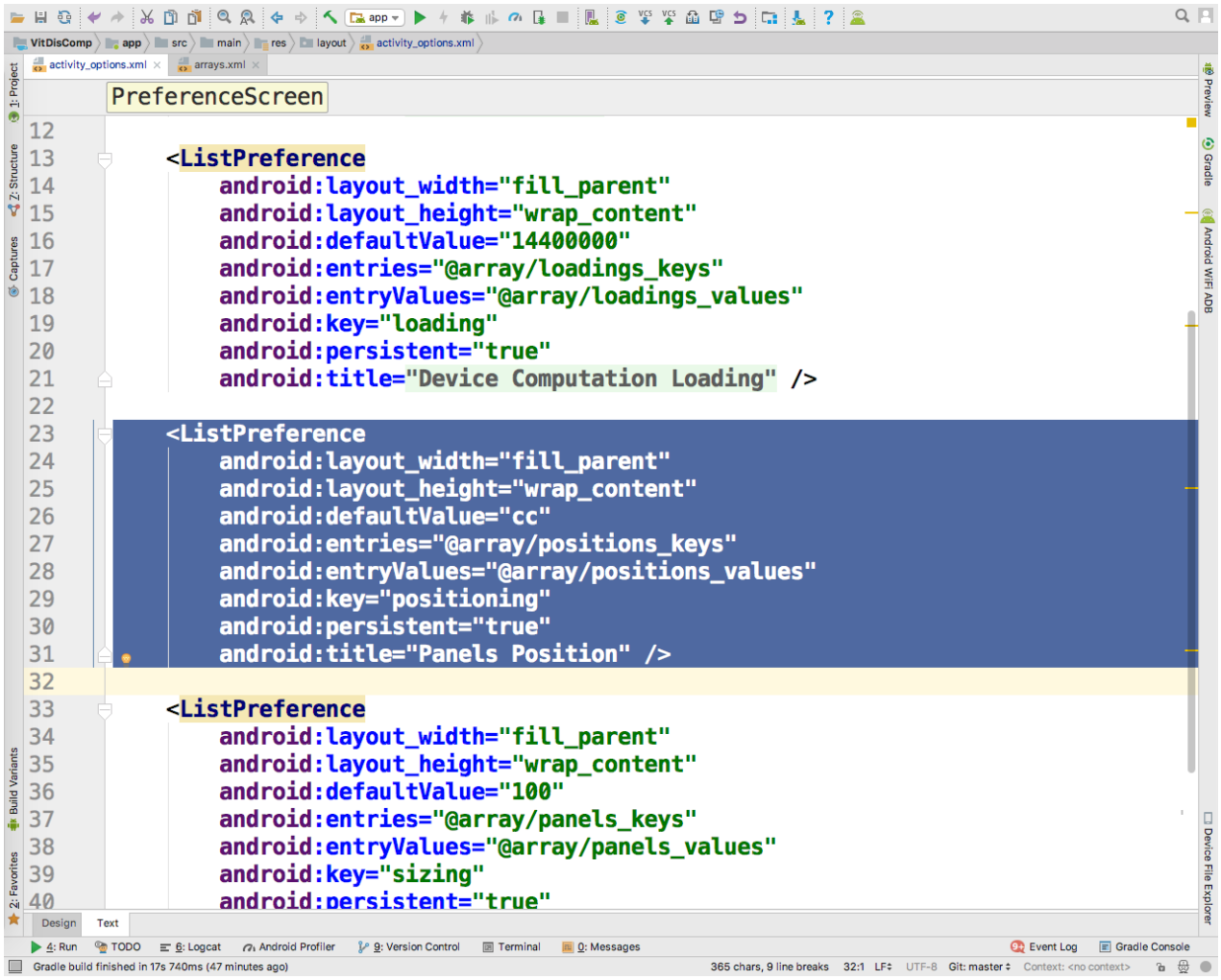
Фигура 4.13: Визуален компонент за регулиране на натоварването

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



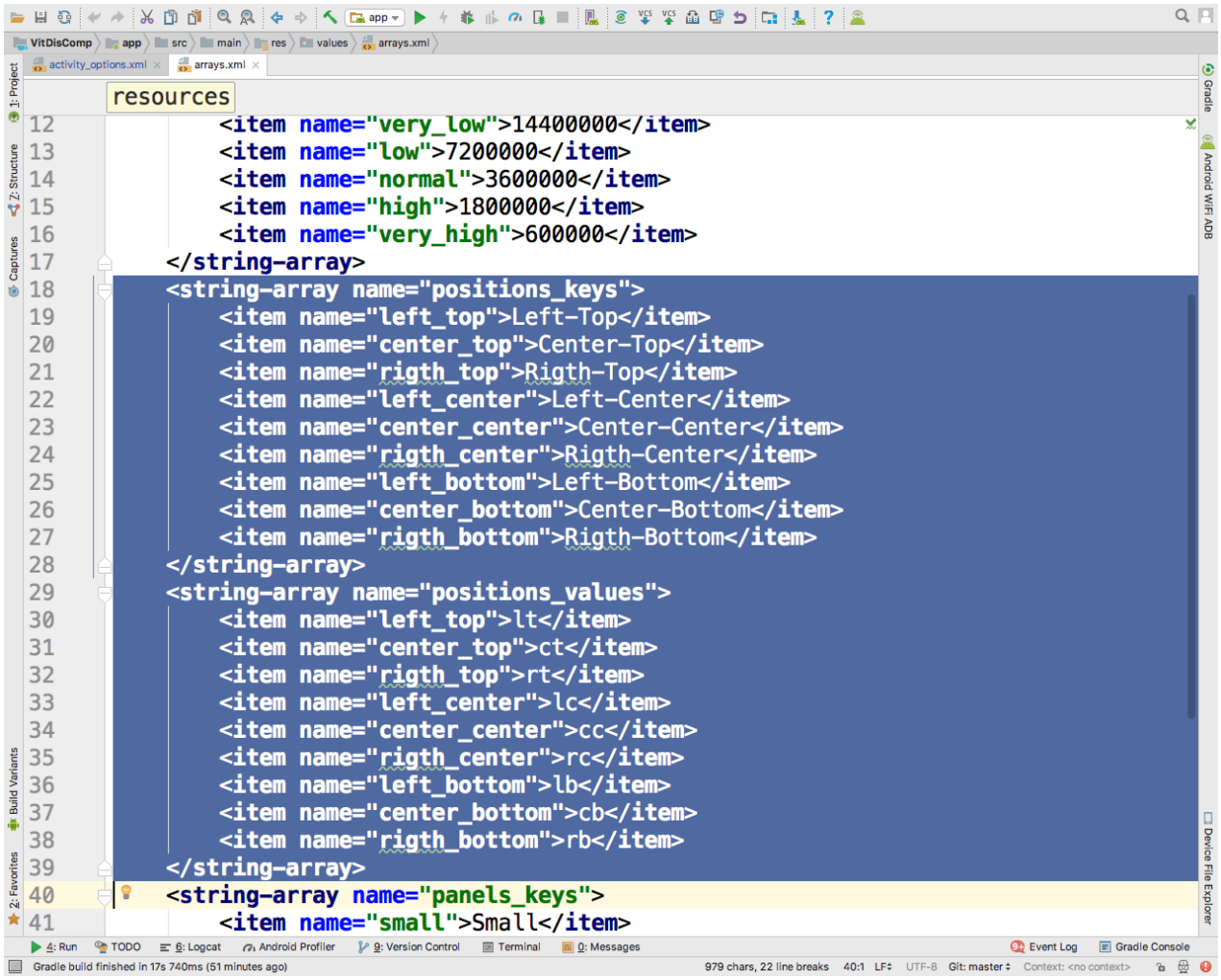
Фигура 4.14: Стойности за натоварване на системата

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).



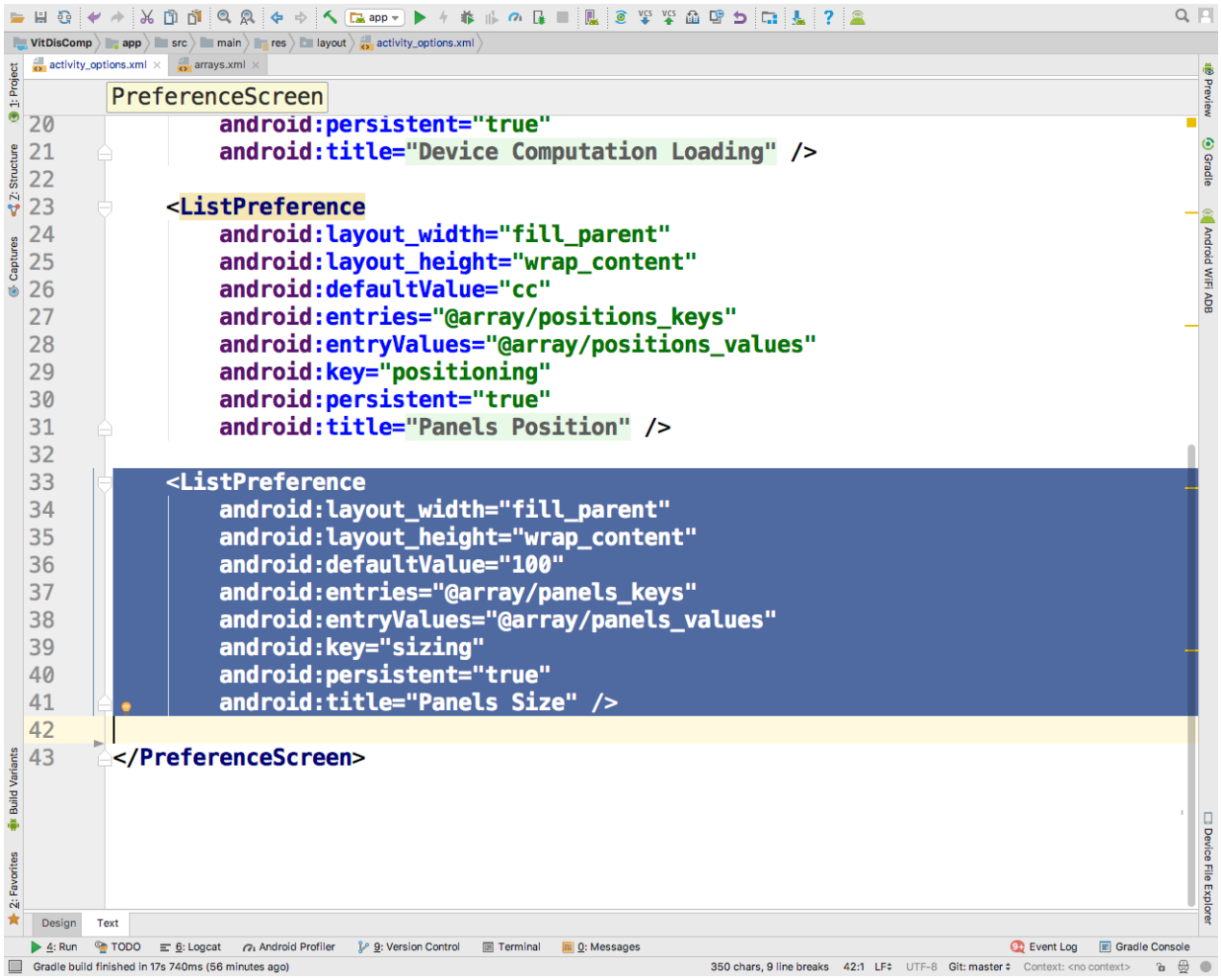
Фигура 4.15: Позициониране на областите за визуално представяне

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



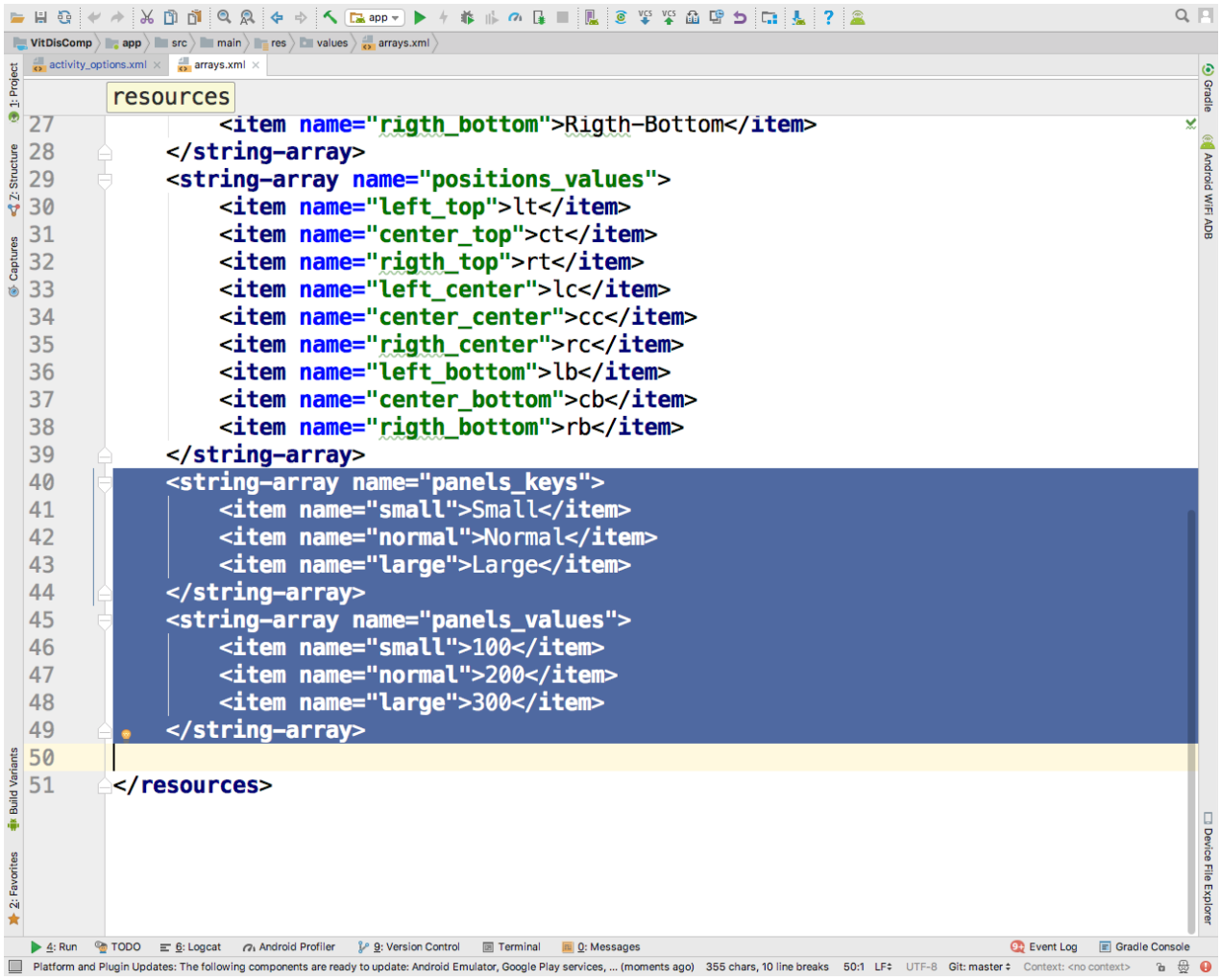
Фигура 4.16: Стойности за позициониране на областите за визуално представяне

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).

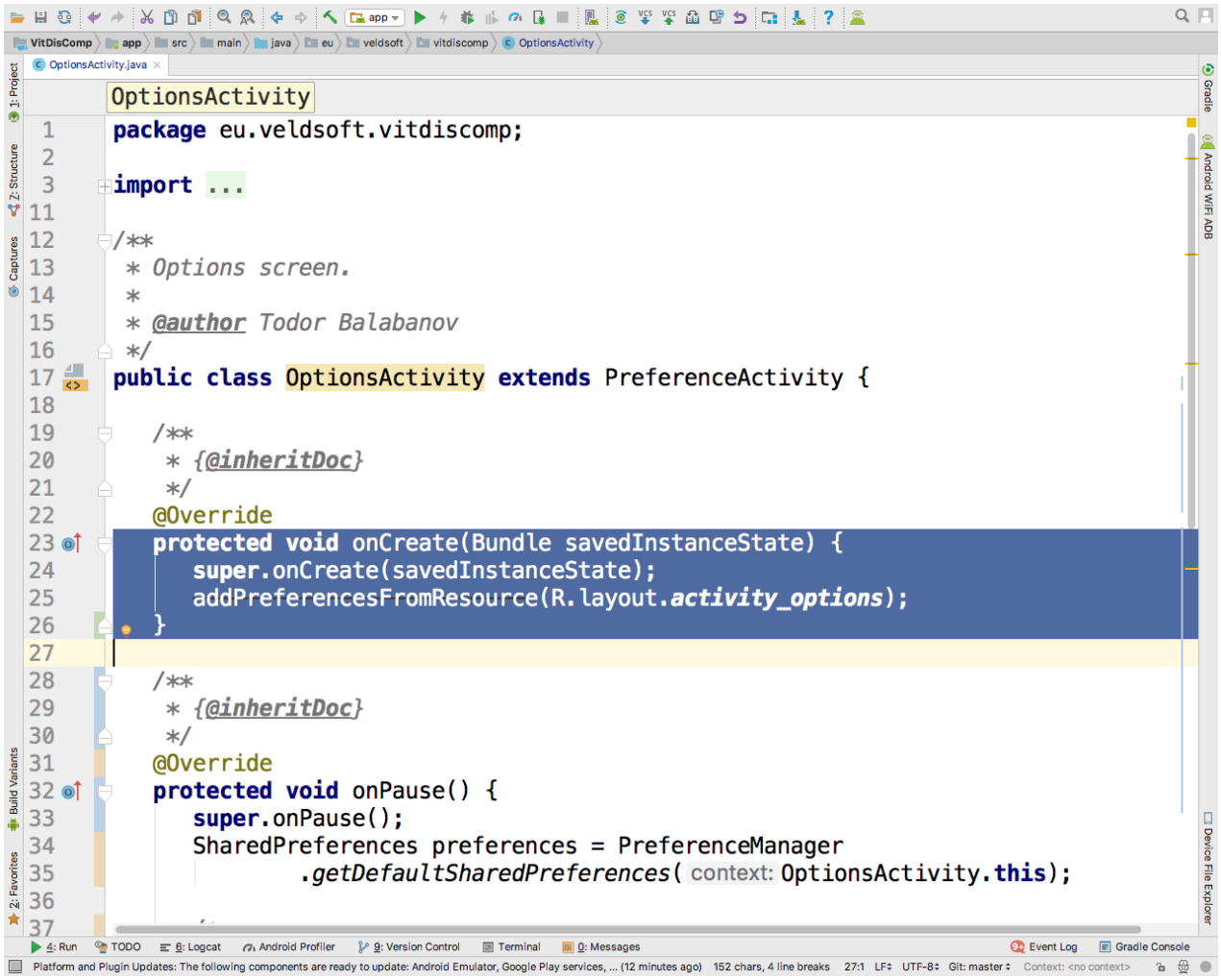


Фигура 4.18: Стойности за размер на областите за визуално представяне

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

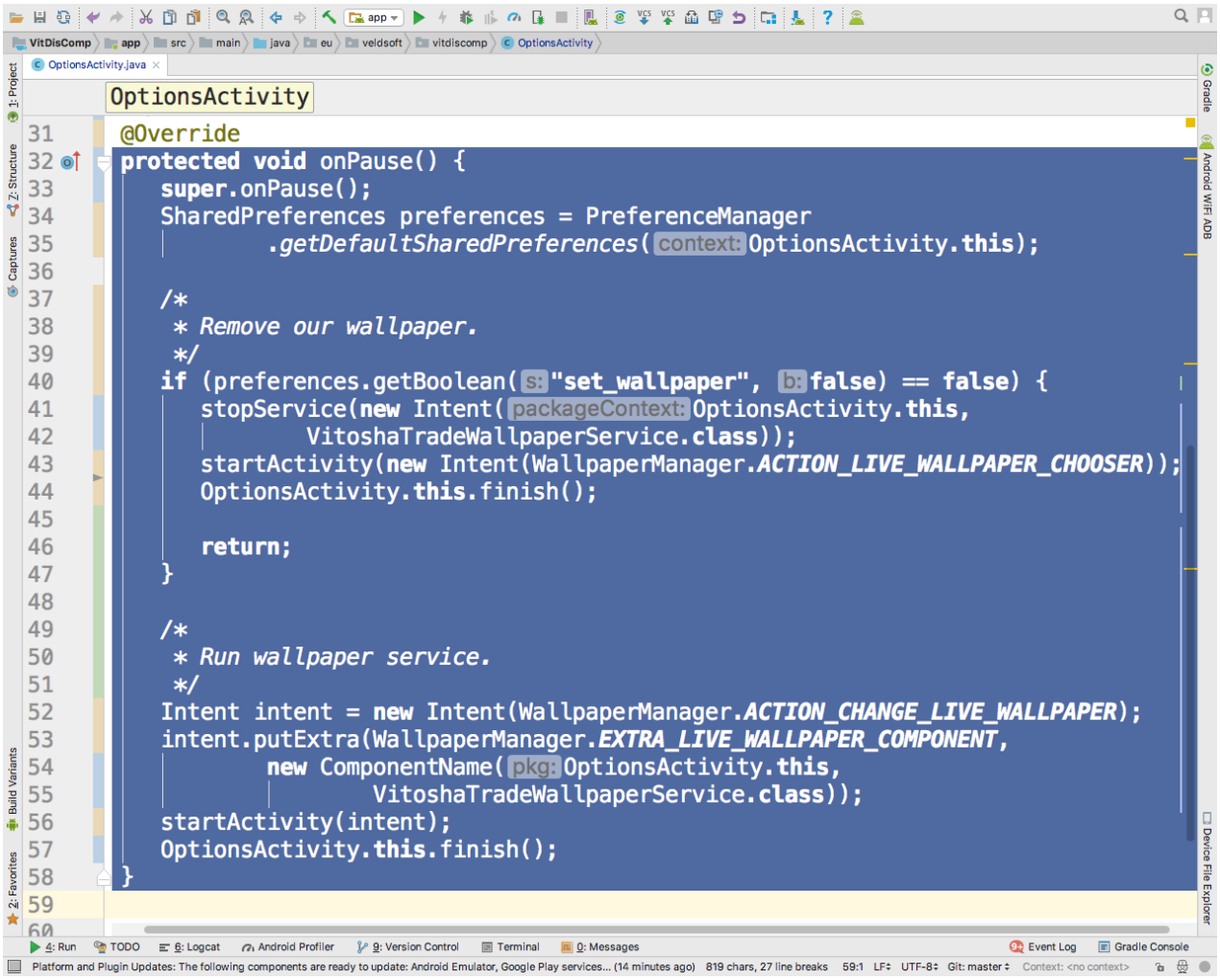
4.2.2 Програмен код за управление на интерфейса

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.



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).

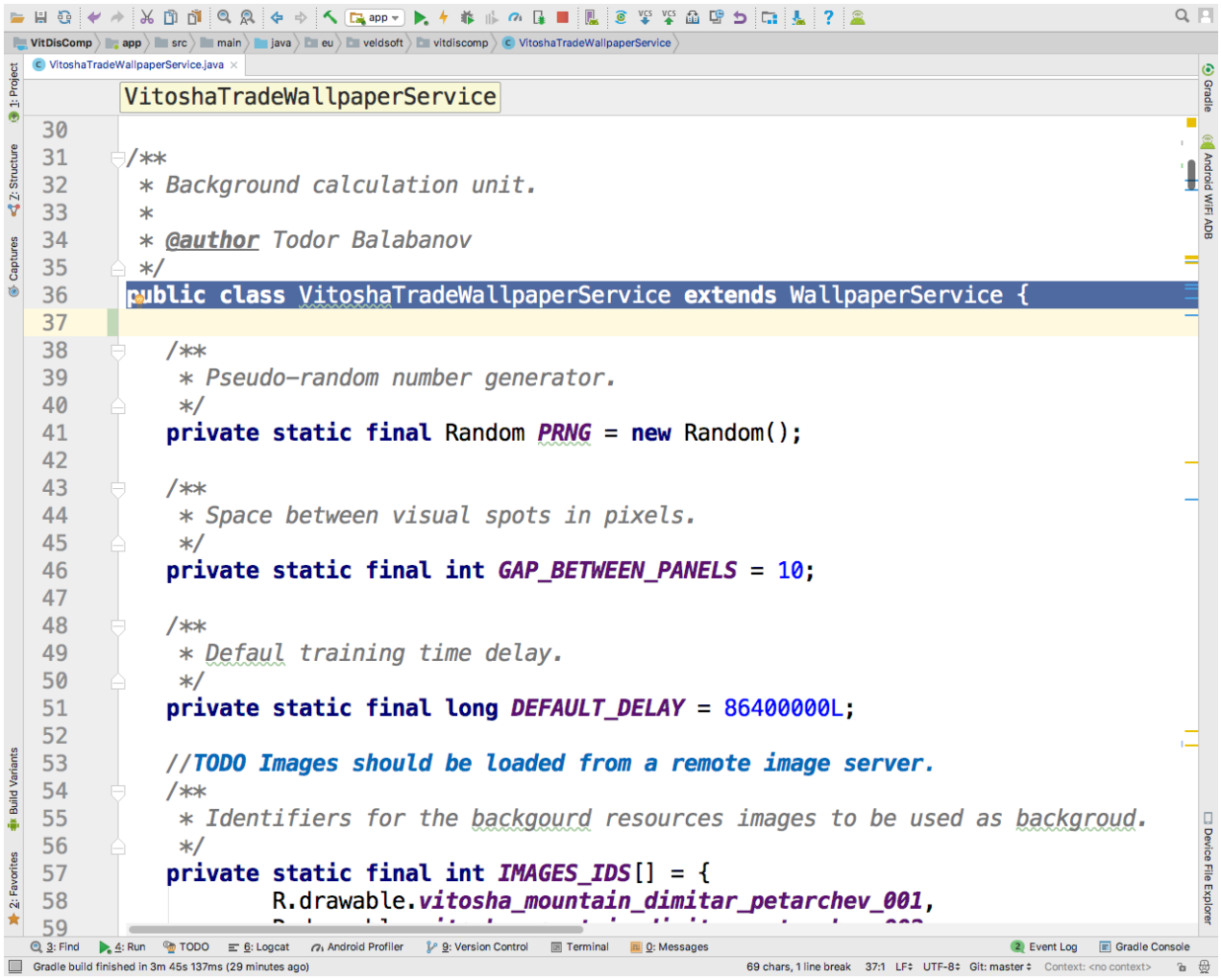


Фигура 4.20: Събитие за пауза на прозореца

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

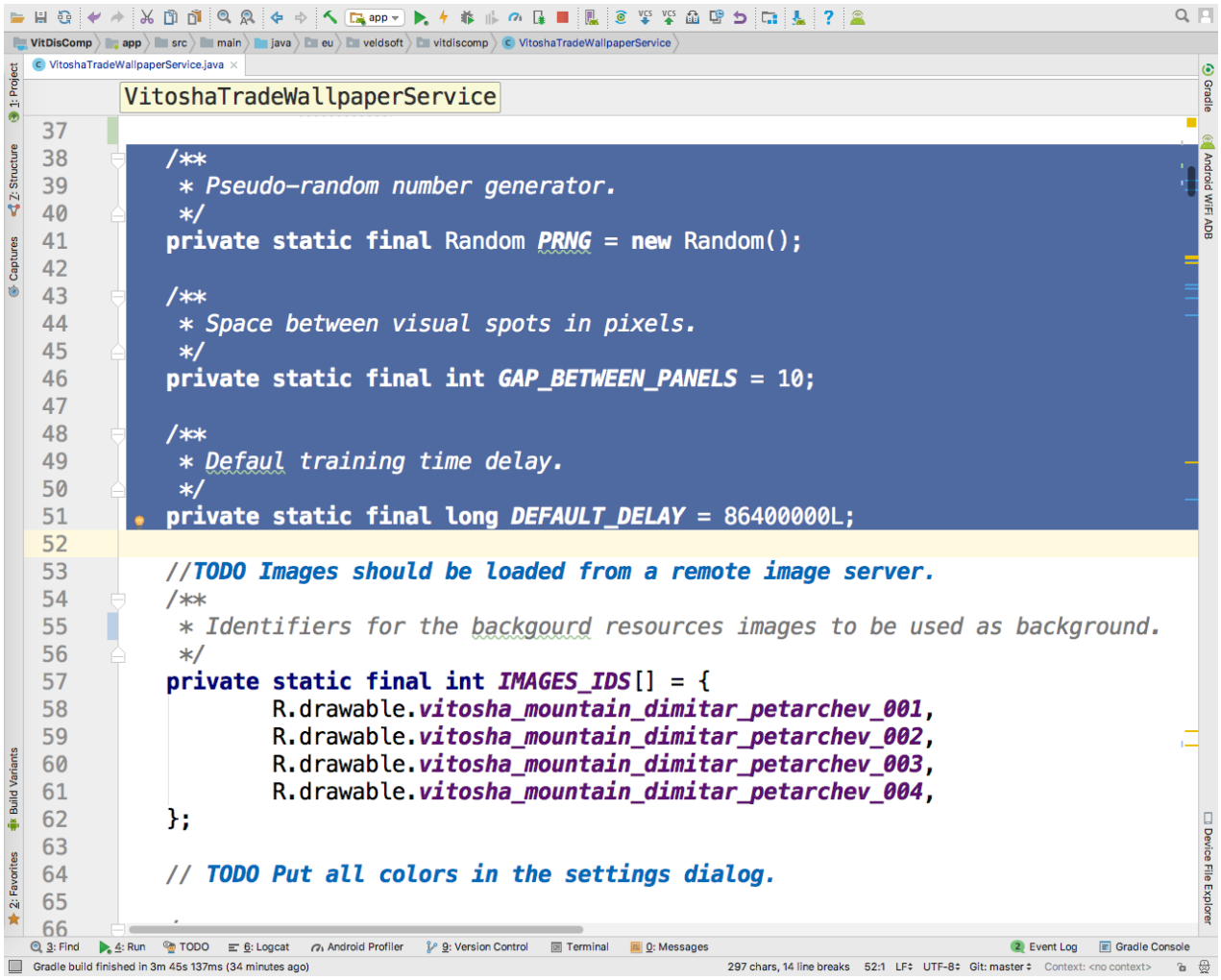
4.3 Пресмятане на фонов режим

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).



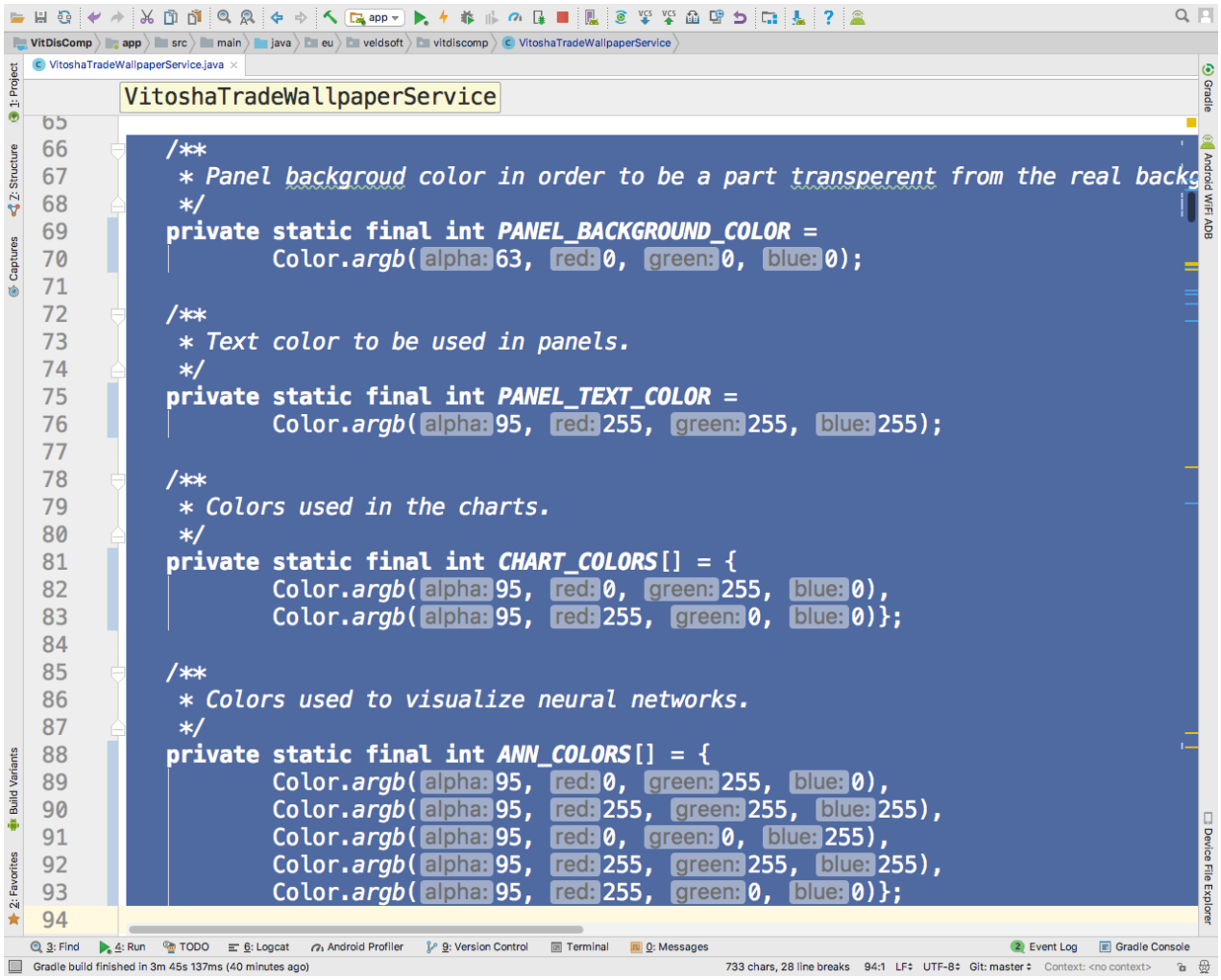
Фигура 4.21: Наследяване на WallpaperService

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).



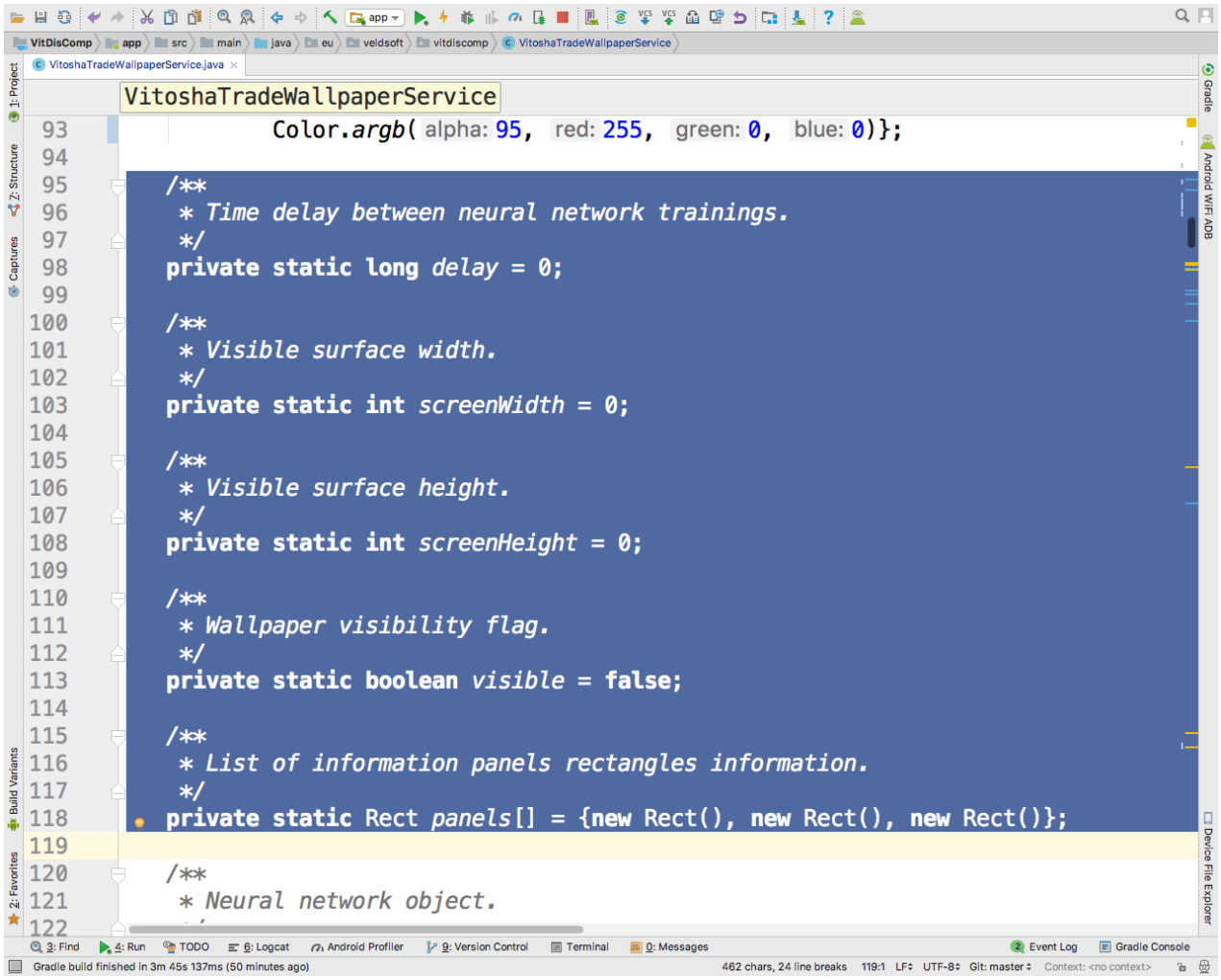
Фигура 4.22: Спомагателни константи

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).



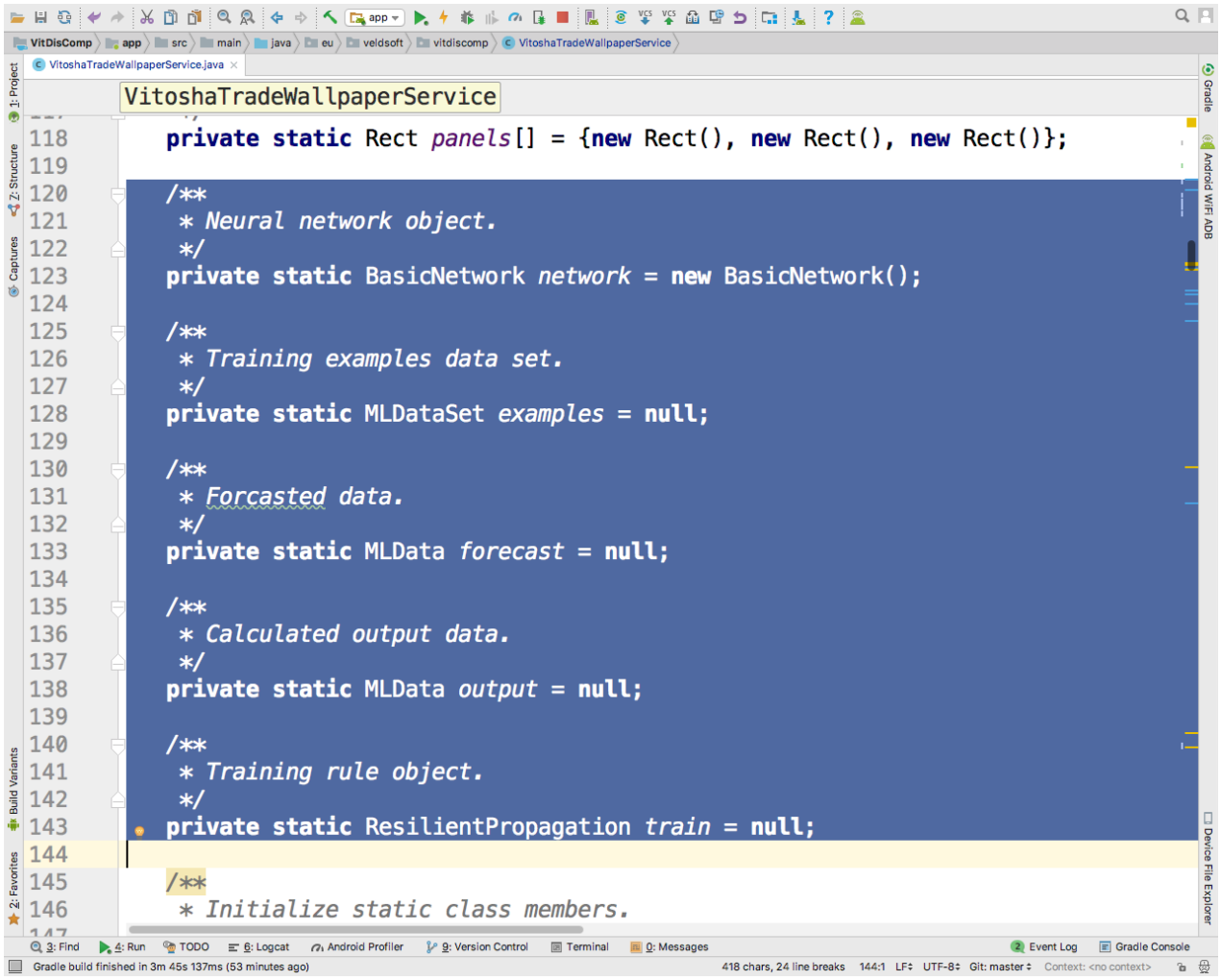
Фигура 4.23: Константи за цветовете

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).



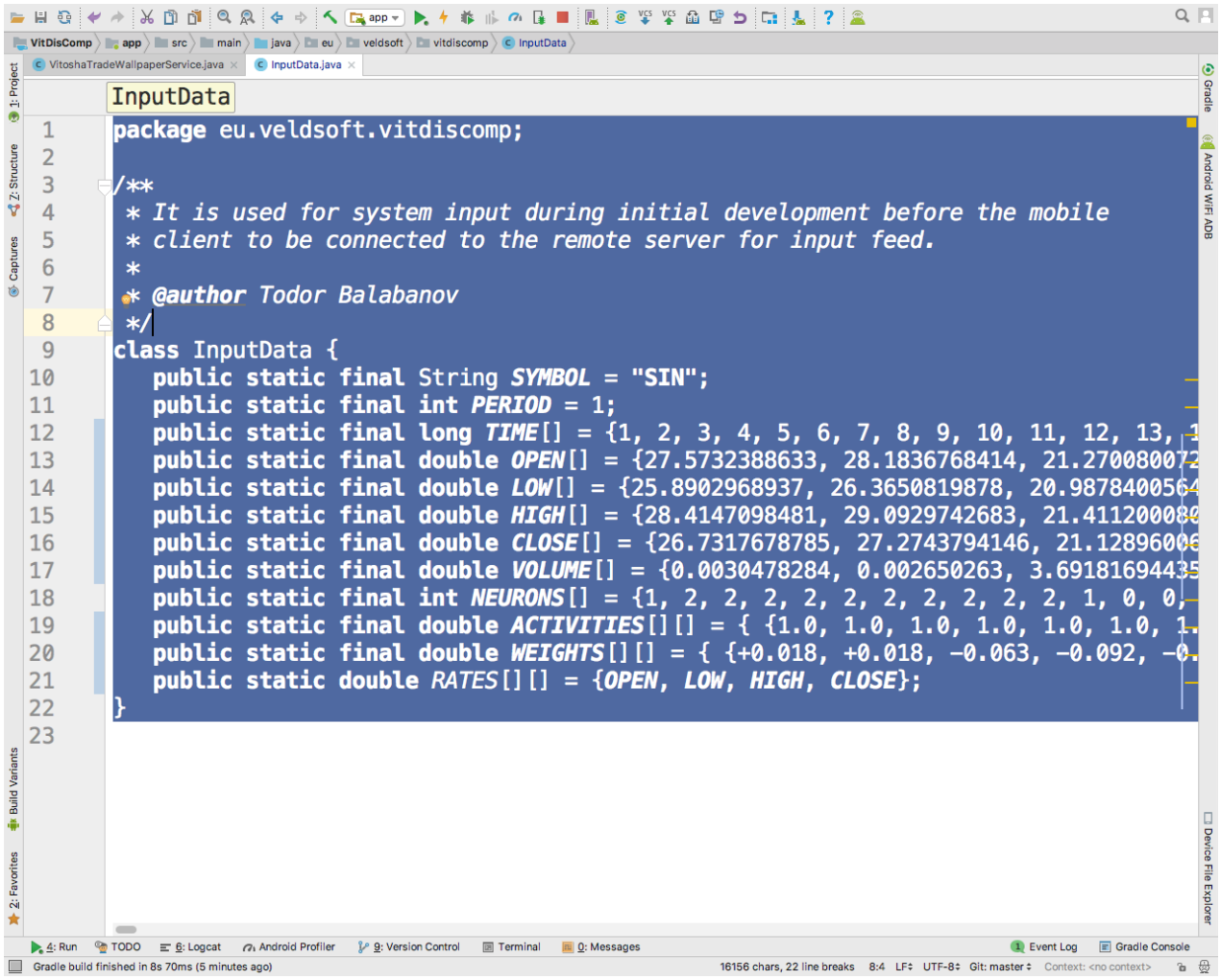
Фигура 4.24: Променливи отразяващи състоянието на активния тапет

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).



Фигура 4.25: Променливи отговарящи за изкуствената невронна мрежа

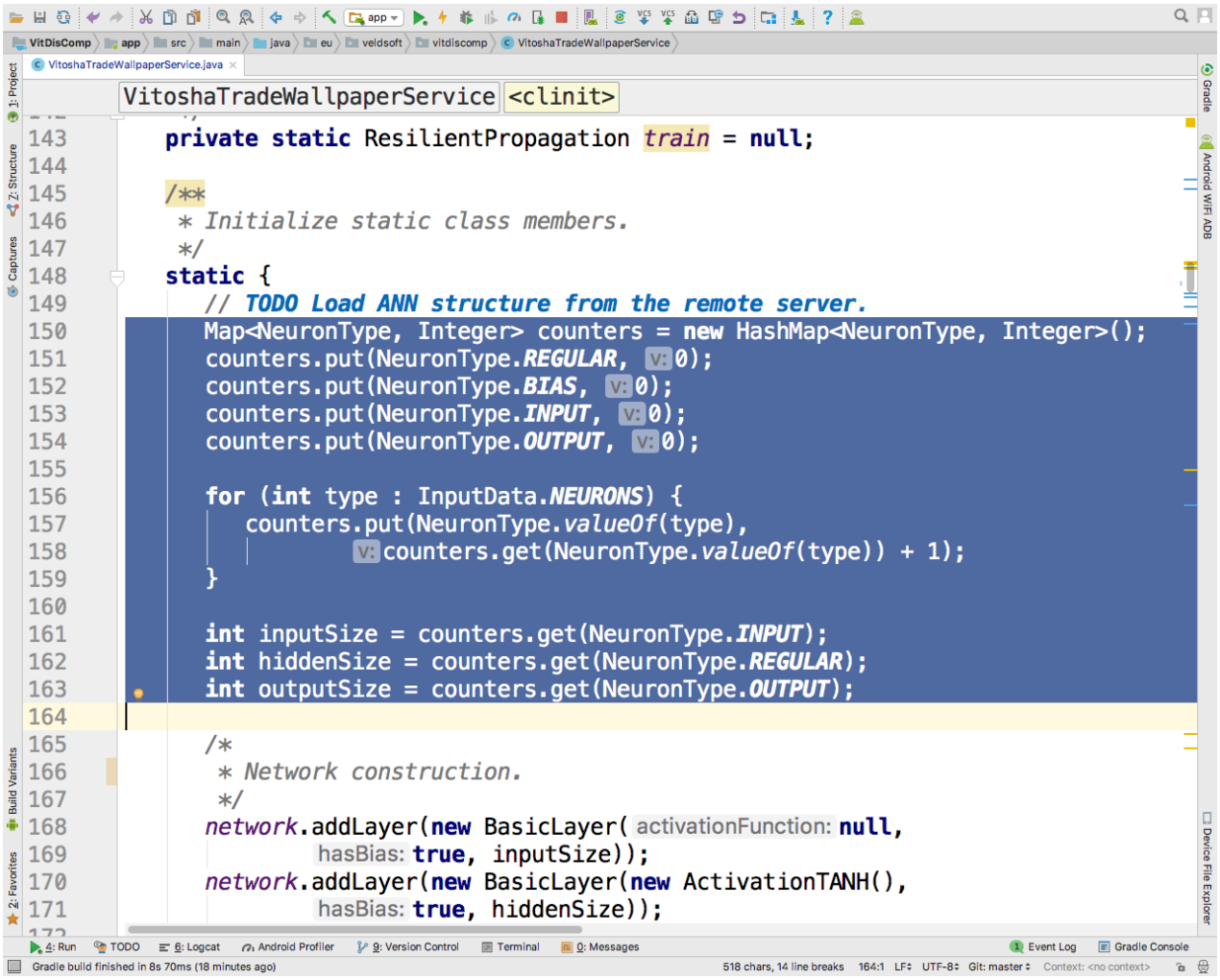
При комерсиалното производство на софтуер много често използван похват е създаването на „софтуерни тапи“. Това са парчета код, които служат за комуникация между обекти и части от системата, които още не са създадени. В настоящия случай точно такава софтуерна тапа представя липсата на сървър и източник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



Фигура 4.26: Променливи отговарящи за изкуствената невронна мрежа

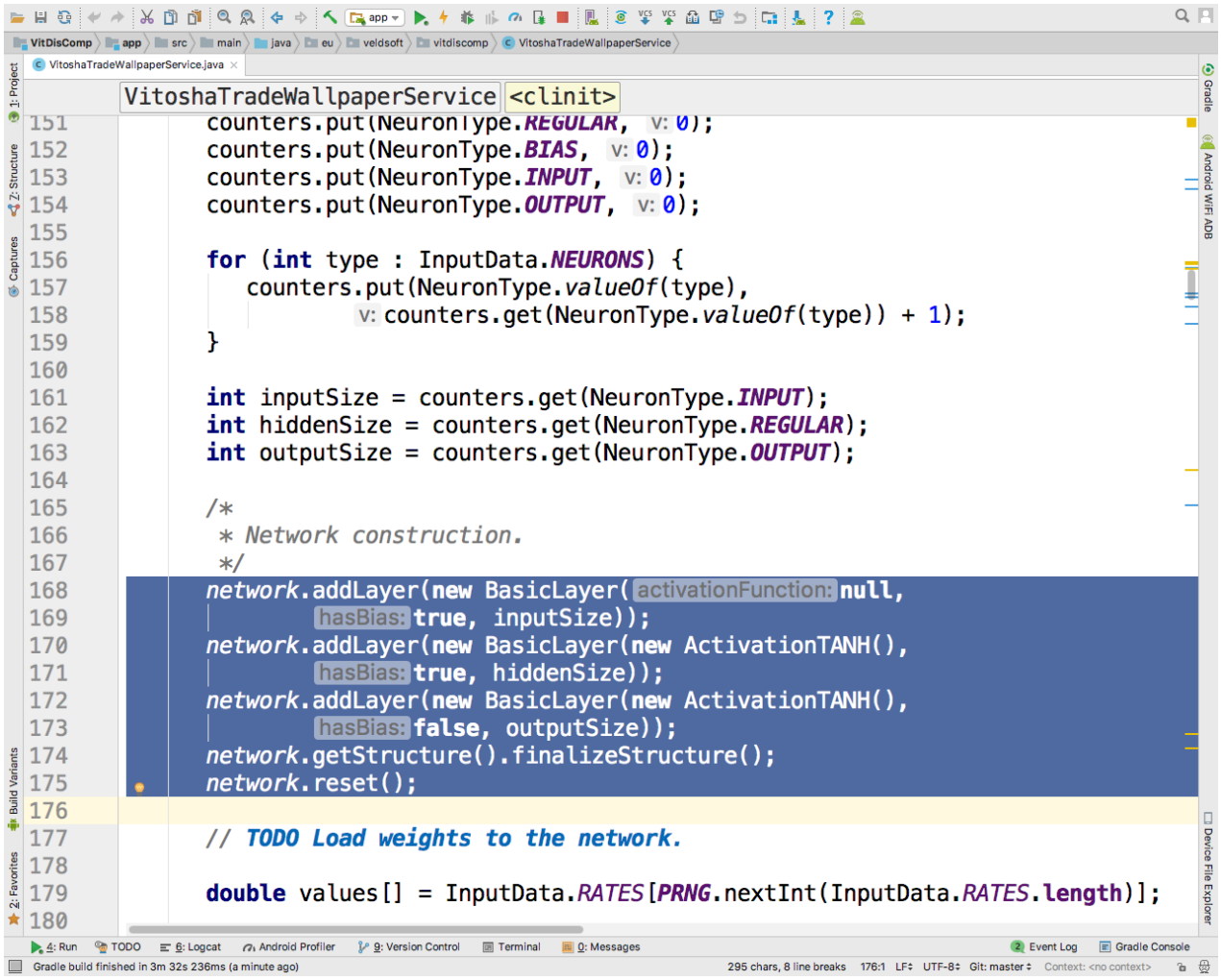
Финансовите времеви редове най-често се описват със следните характеристики: 1. Вид финансов инструмент (ticker symbol или stock symbol), в случая математическата функция синус; 2. Интервал между отделните измервания (отчитан в брой минути), в случая една минута; 3. Шест паралелни масива (дискретно време, нива на отваряне, най-ниско постигнато ниво, най-високо постигнато ниво, нива на затваряне и изтъргуван обем).

Освен финансовата информация, към клиентското приложение трябва да се подава и информация за топологията на изкуствената невронна мрежа. За описанието на изкуствената невронна мрежа са нужни: 1. Брой, подредба и тип на невроните (едномерен масив от константи); 2. Матрица на съседство между невроните (единица там, където има връзка и нула там, където няма връзка); 3. Текуща стойност на тегловния коефициент за всяка връзка между два неврона (включително и примките, ако има такива).



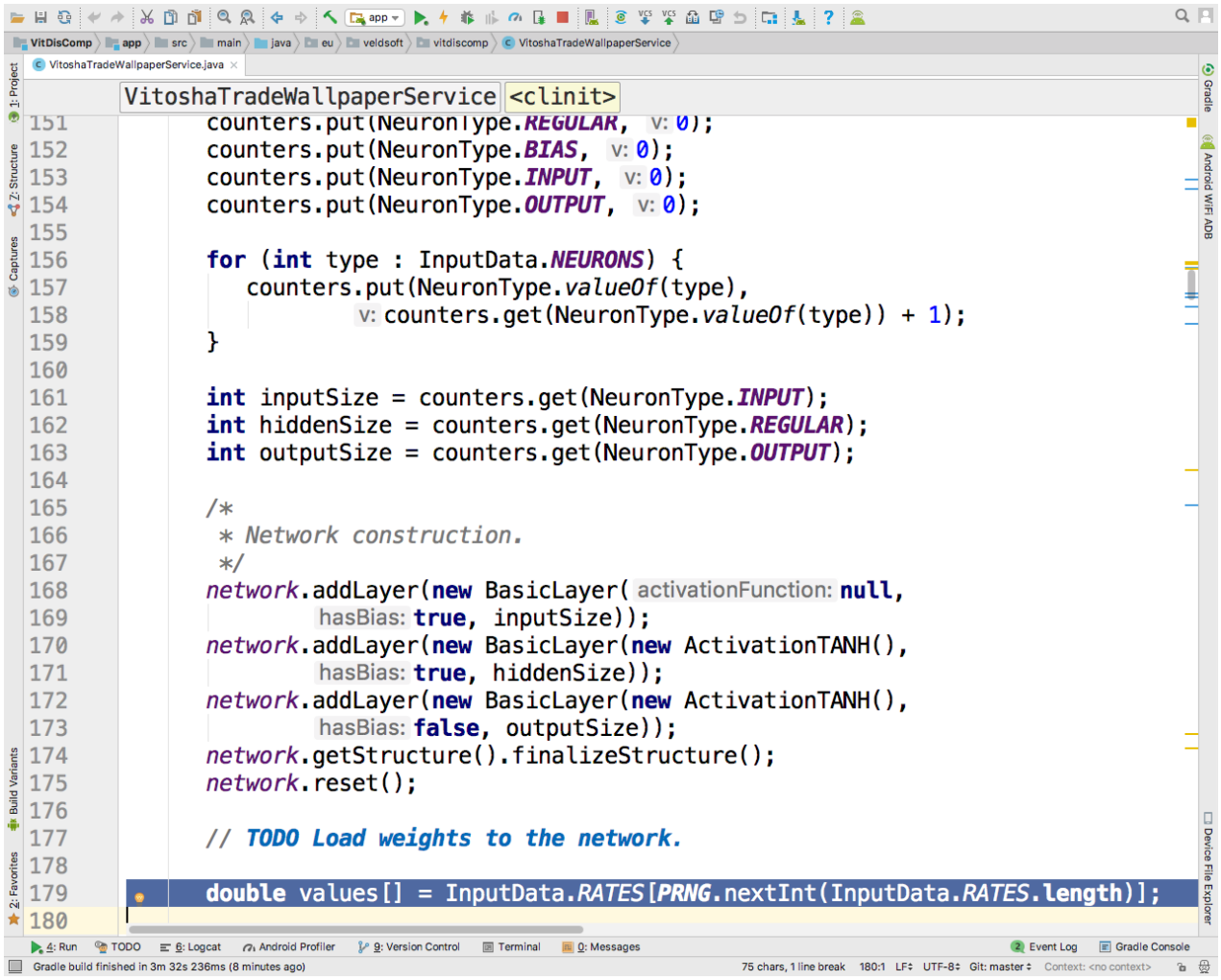
Фигура 4.27: Определяне на броя и типовете неврони

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



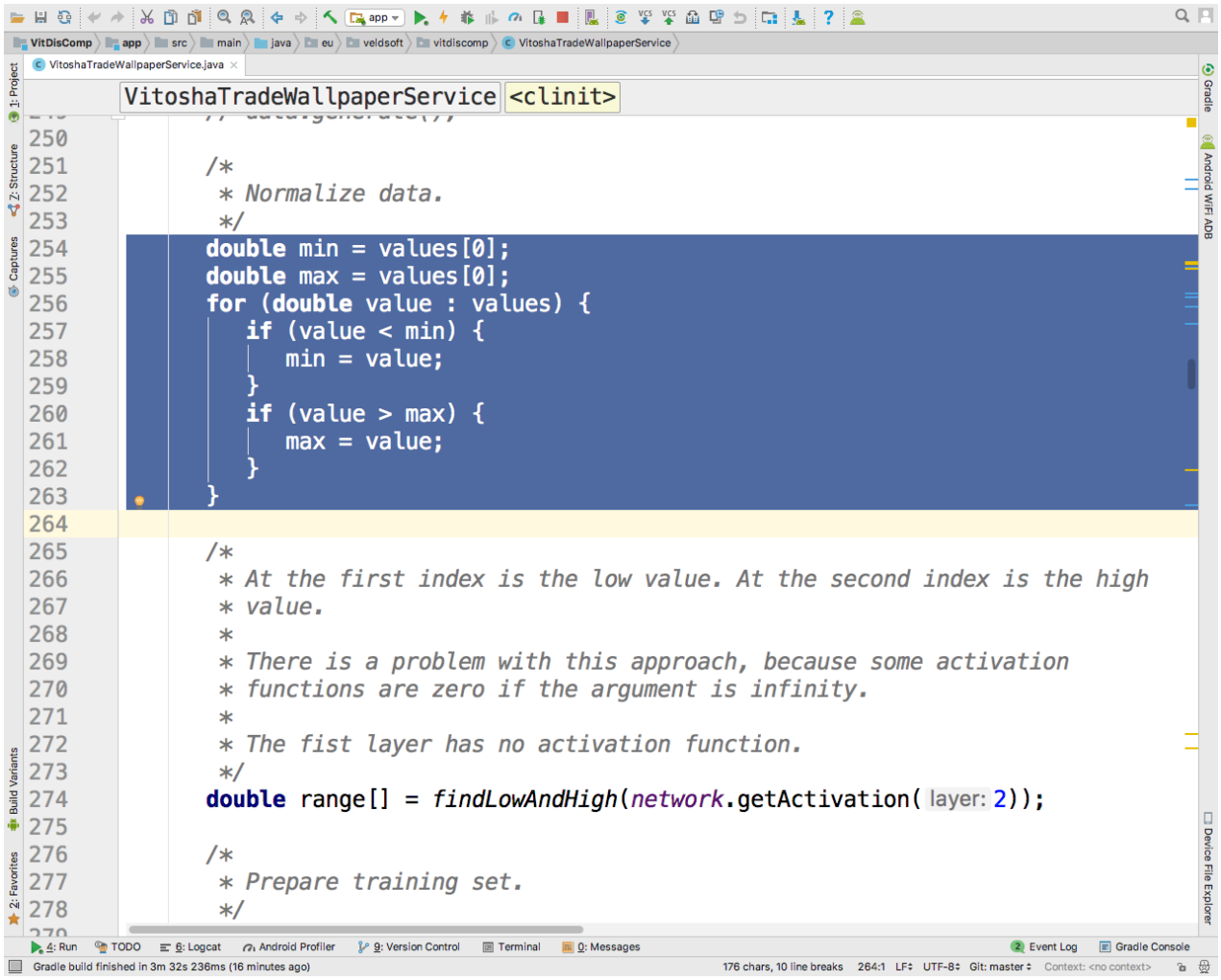
Фигура 4.28: Структура на трислойна изкуствена невронна мрежа

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 скритият 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 слои, тъй като отместващият неврон винаги емитира високото ниво на сигнализация.



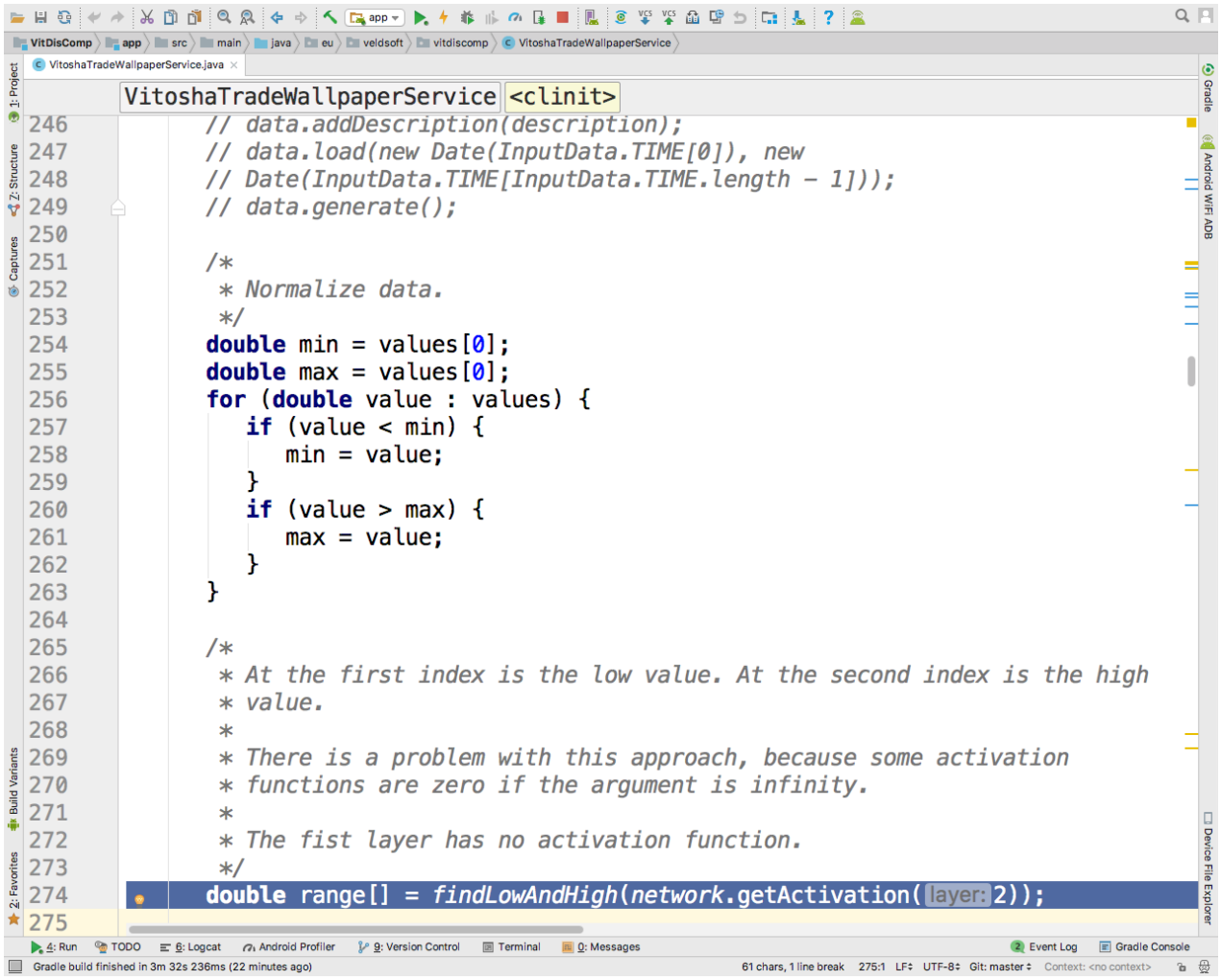
Фигура 4.29: Избор на стойности за прогнозиране

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).



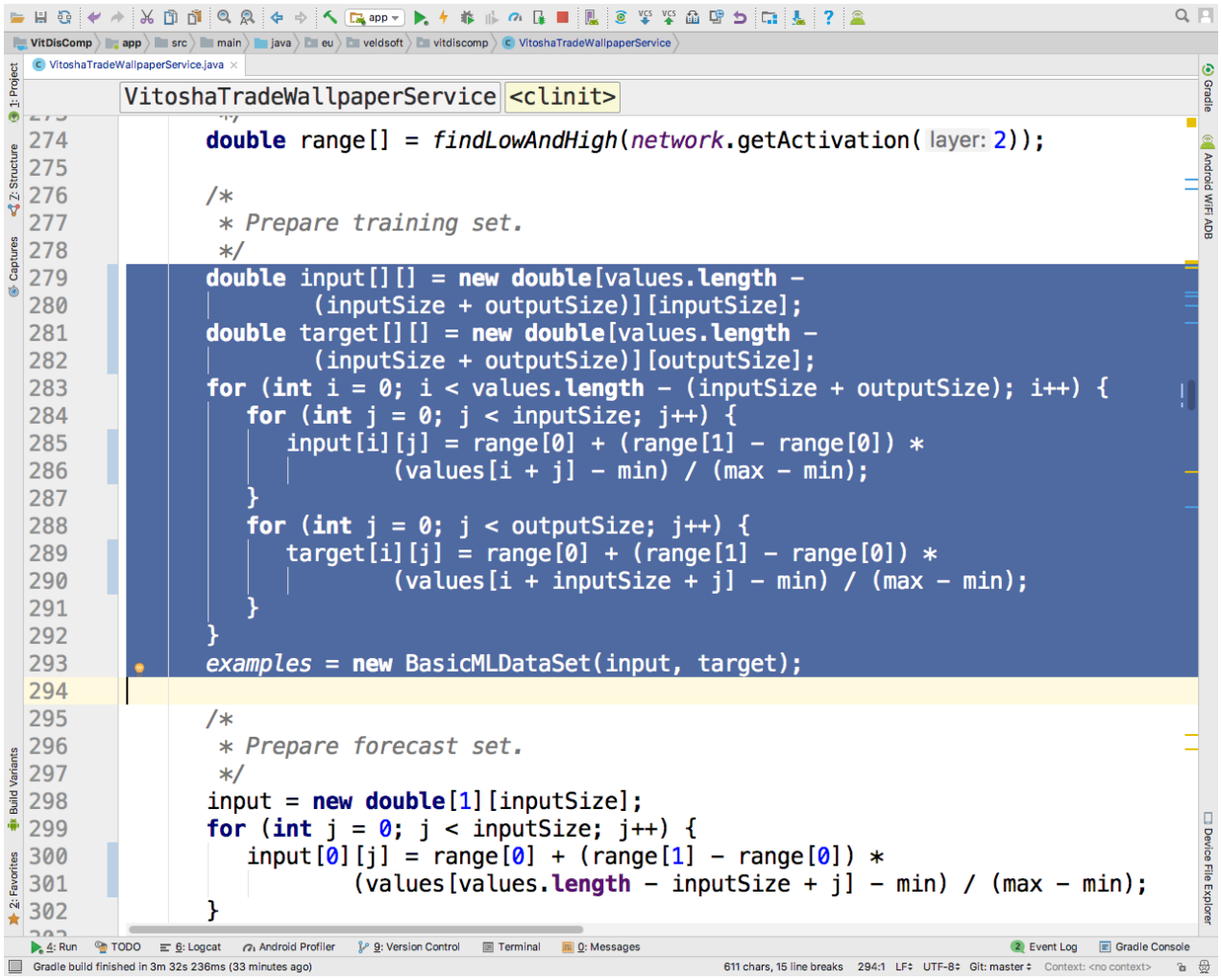
Фигура 4.30: Определяне на границите във времевия ред

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 постигане 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).



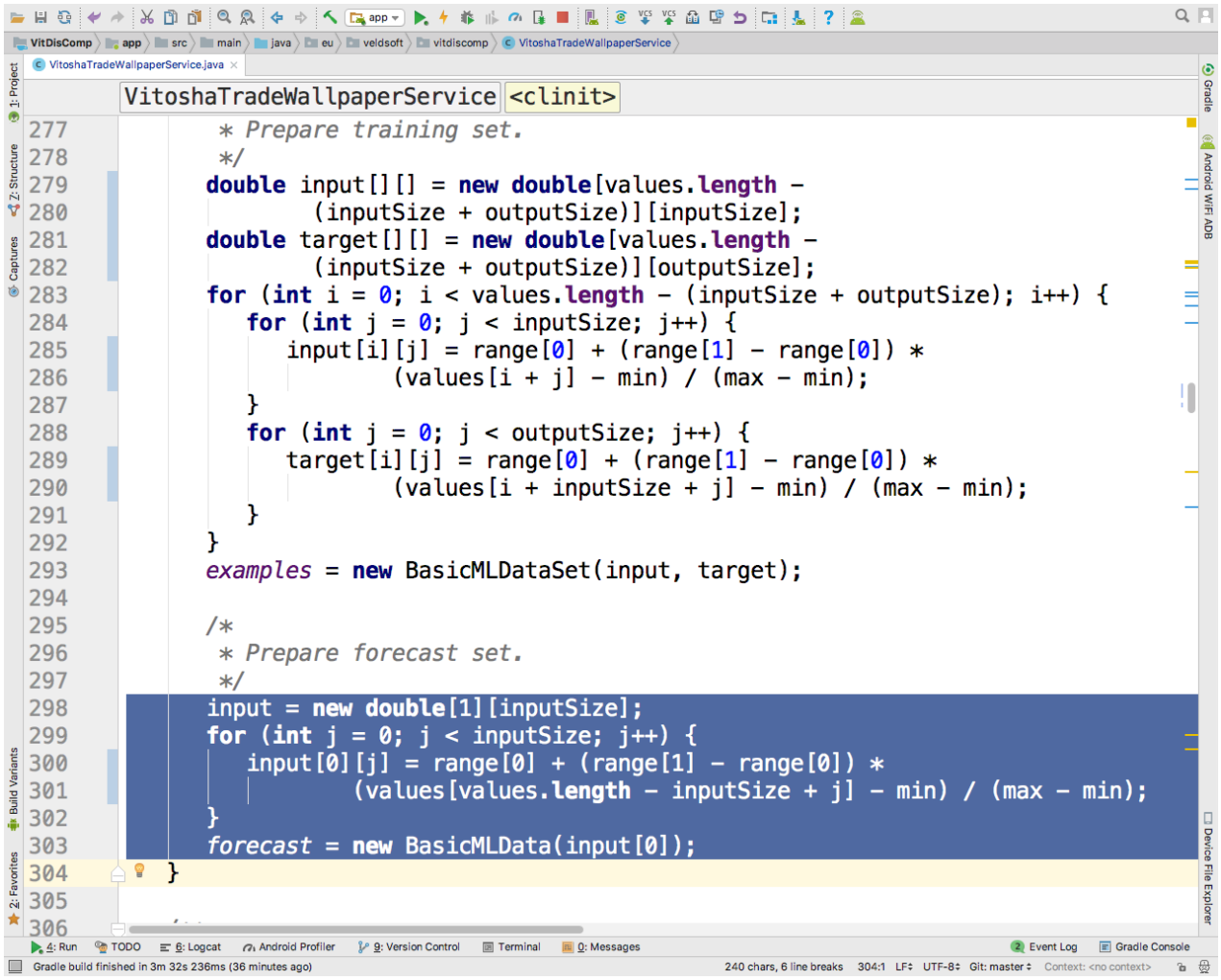
Фигура 4.31: Диапазон на активационната функция в изходния слой

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.



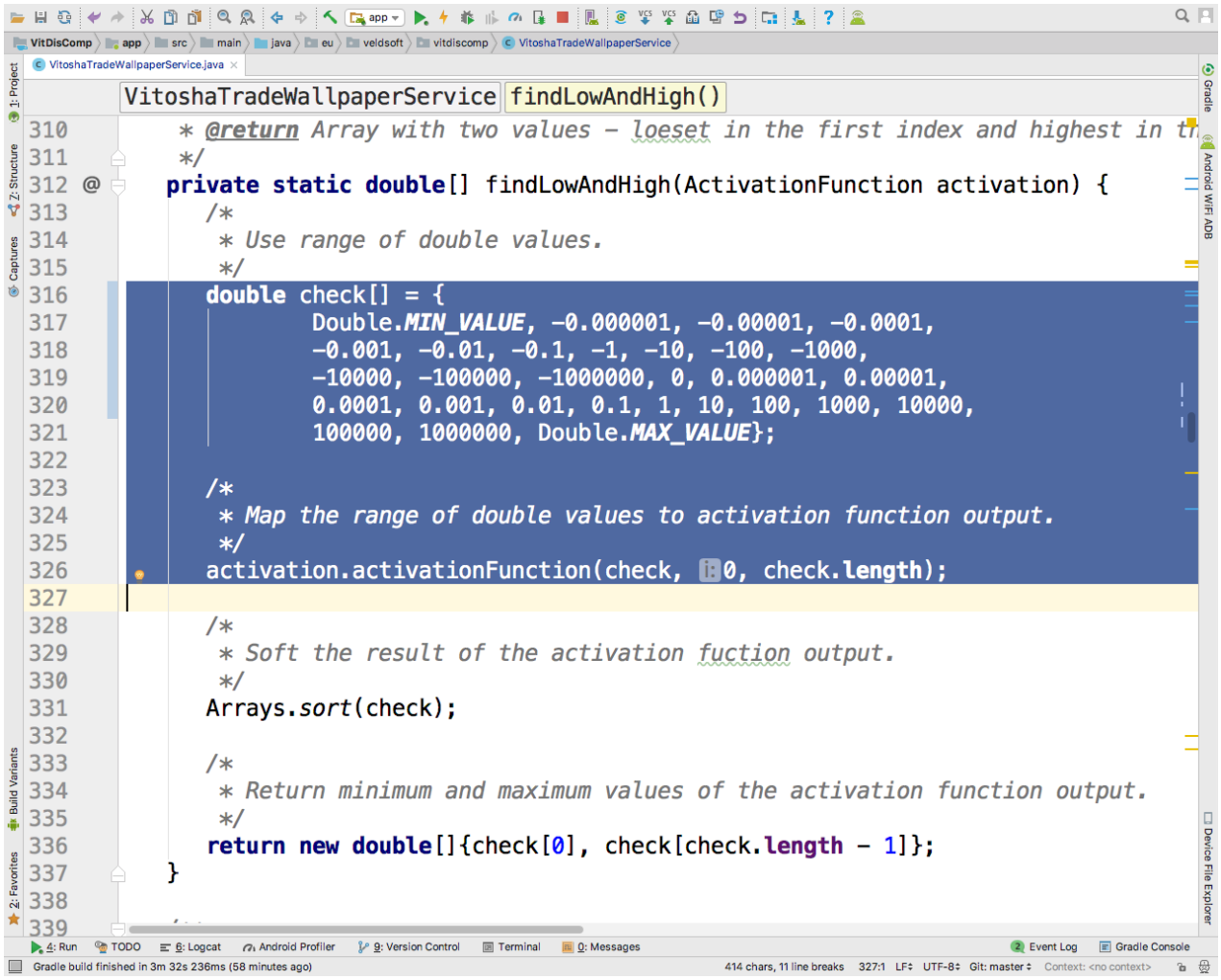
Фигура 4.32: Тренировъчни примери

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).



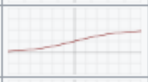
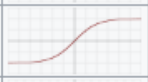
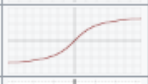
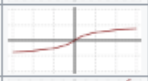
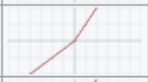
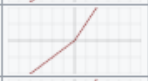
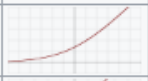
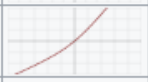
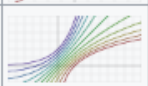
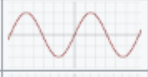
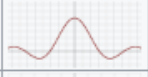
Фигура 4.33: Входни данни към изкуствената неверонна мрежа за получаване на прогноза

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



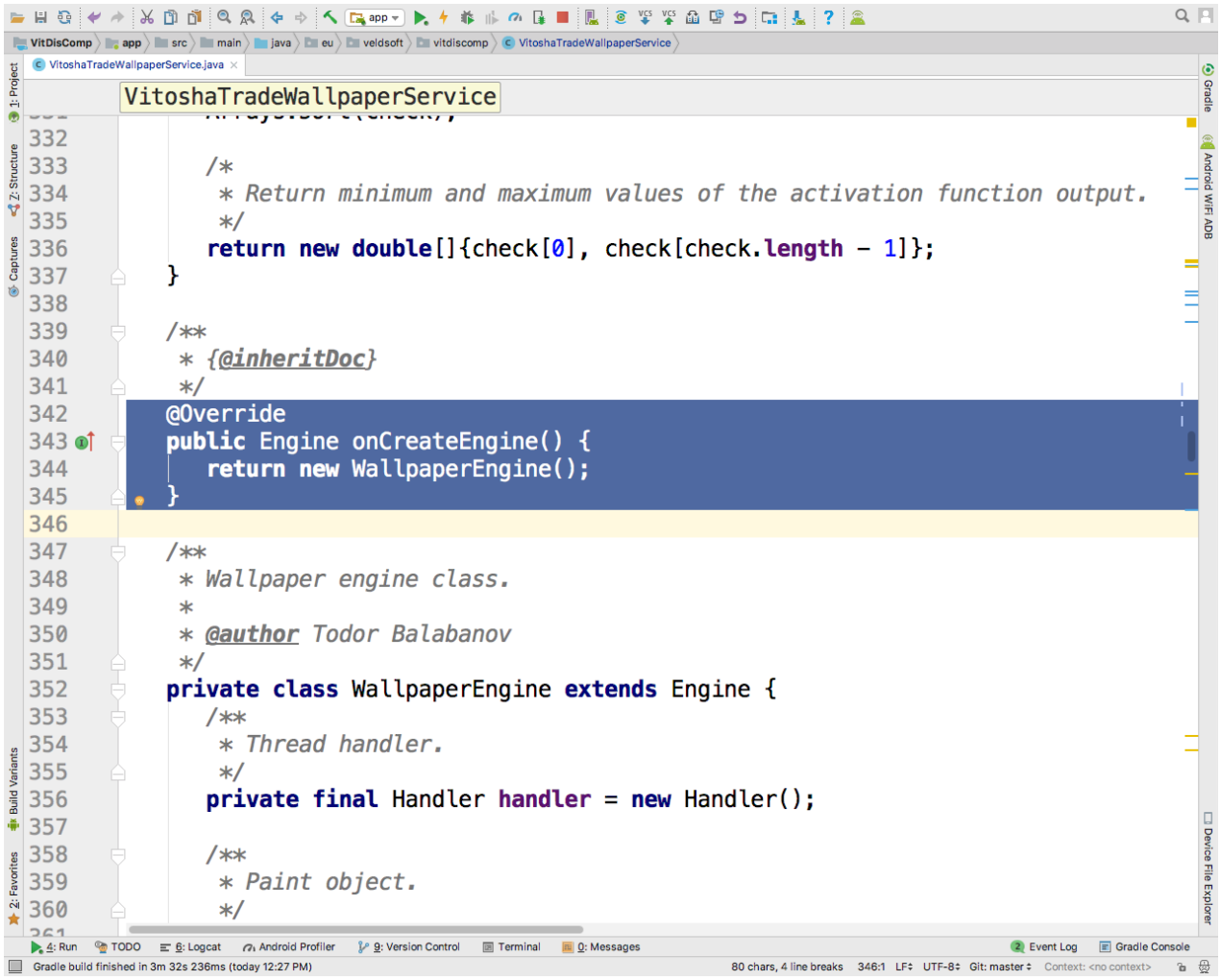
Фигура 4.34: Изследване на група от стойности за определяне на диапазона

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		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2 + 1}$	$\left(-\frac{\pi}{2}, \frac{\pi}{2}\right)$
Softsign [7][8]		$f(x) = \frac{x}{1 + \lvert x \rvert}$	$f'(x) = \frac{1}{(1 + \lvert x \rvert)^2}$	$(-1, 1)$
Rectified linear unit (ReLU)[9]		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$[0, \infty)$
Leaky rectified linear unit (Leaky ReLU)[10]		$f(x) = \begin{cases} 0.01x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0.01 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$(-\infty, \infty)$
Parameteric rectified linear unit (PReLU)[11]		$f(\alpha, x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(\alpha, x) = \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$(-\infty, \infty)$
Randomized leaky rectified linear unit (RReLU)[12]		$f(\alpha, x) = \begin{cases} \alpha x & \text{for } x < 0 \text{ [1]} \\ x & \text{for } x \geq 0 \end{cases}$	$f'(\alpha, x) = \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$(-\infty, \infty)$
Exponential linear unit (ELU)[13]		$f(\alpha, x) = \begin{cases} \alpha(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(\alpha, x) = \begin{cases} f(\alpha, x) + \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$(-\alpha, \infty)$
Scaled exponential linear unit (SELU)[14]		$f(\alpha, x) = \lambda \begin{cases} \alpha(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$ 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)[15]		$f_{t_l, a_l, t_r, a_r}(x) = \begin{cases} t_l + a_l(x - t_l) & \text{for } x \leq t_l \\ x & \text{for } t_l < x < t_r \\ t_r + a_r(x - t_r) & \text{for } x \geq t_r \end{cases}$ 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) [16]		$f(x) = \max(0, x) + \sum_{s=1}^{S} a_i^s \max(0, -x + b_i^s)$	$f'(x) = H(x) - \sum_{s=1}^{S} a_i^s H(-x + b_i^s)$ [2]	$(-\infty, \infty)$
SoftPlus[17]		$f(x) = \ln(1 + e^x)$	$f'(x) = \frac{1}{1 + e^{-x}}$	$(0, \infty)$
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 [18]		$f(\alpha, x) = \begin{cases} -\frac{\ln(1 - \alpha(x + \alpha))}{\alpha} & \text{for } \alpha < 0 \\ x & \text{for } \alpha = 0 \\ \frac{e^{\alpha x} - 1}{\alpha} + \alpha & \text{for } \alpha > 0 \end{cases}$	$f'(\alpha, x) = \begin{cases} \frac{1}{1 - \alpha(\alpha + x)} & \text{for } \alpha < 0 \\ e^{\alpha x} & \text{for } \alpha \geq 0 \end{cases}$	$(-\infty, \infty)$
Sinusoid[19]		$f(x) = \sin(x)$	$f'(x) = \cos(x)$	$[-1, 1]$
Sinc		$f(x) = \begin{cases} 1 & \text{for } x = 0 \\ \frac{\sin(x)}{x} & \text{for } x \neq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x = 0 \\ \frac{\cos(x)}{x} - \frac{\sin(x)}{x^2} & \text{for } x \neq 0 \end{cases}$	$[\approx -.217234, 1]$
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).

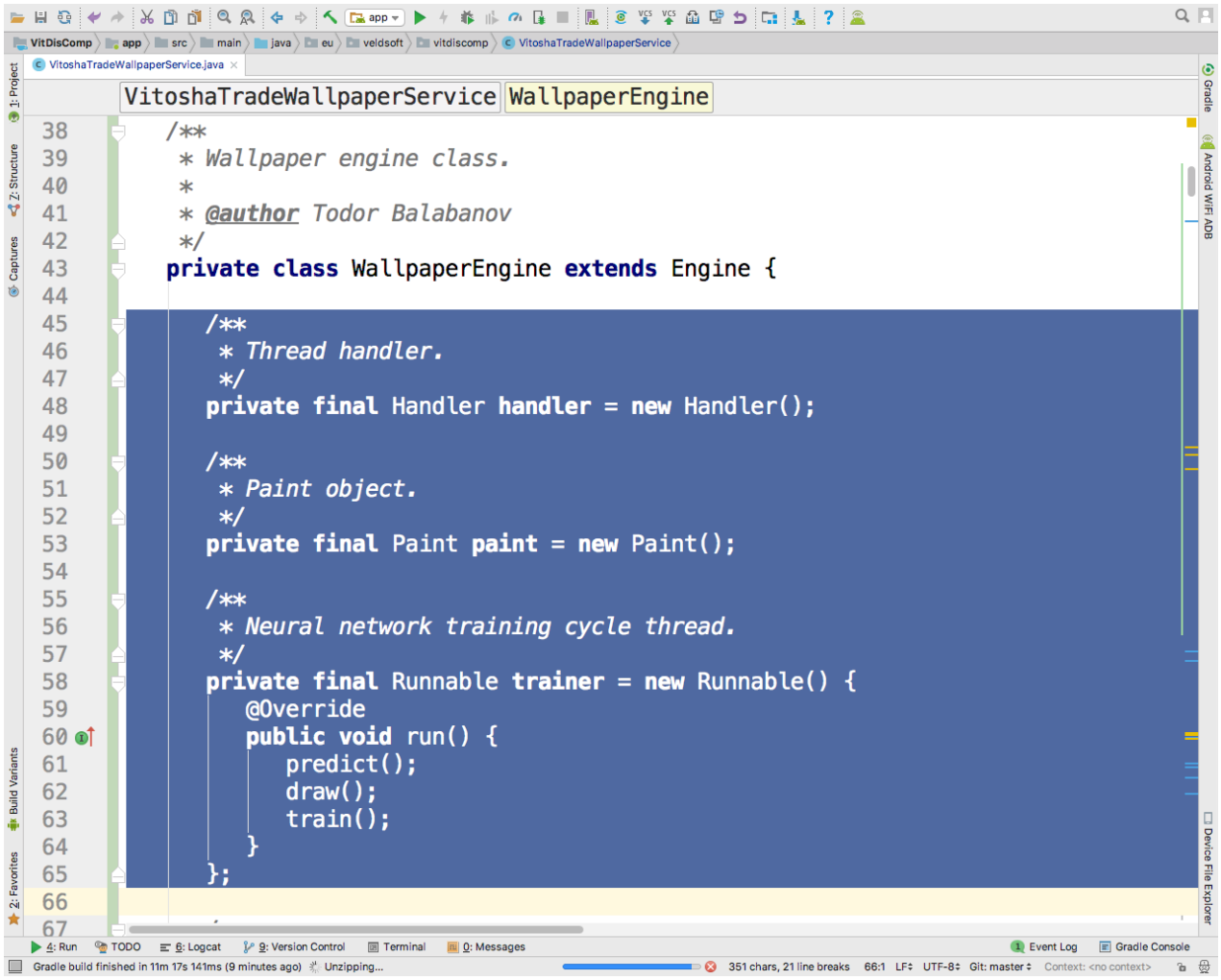


Фигура 4.36: Собствена реализация на наследения Engine клас и създаване на обект

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

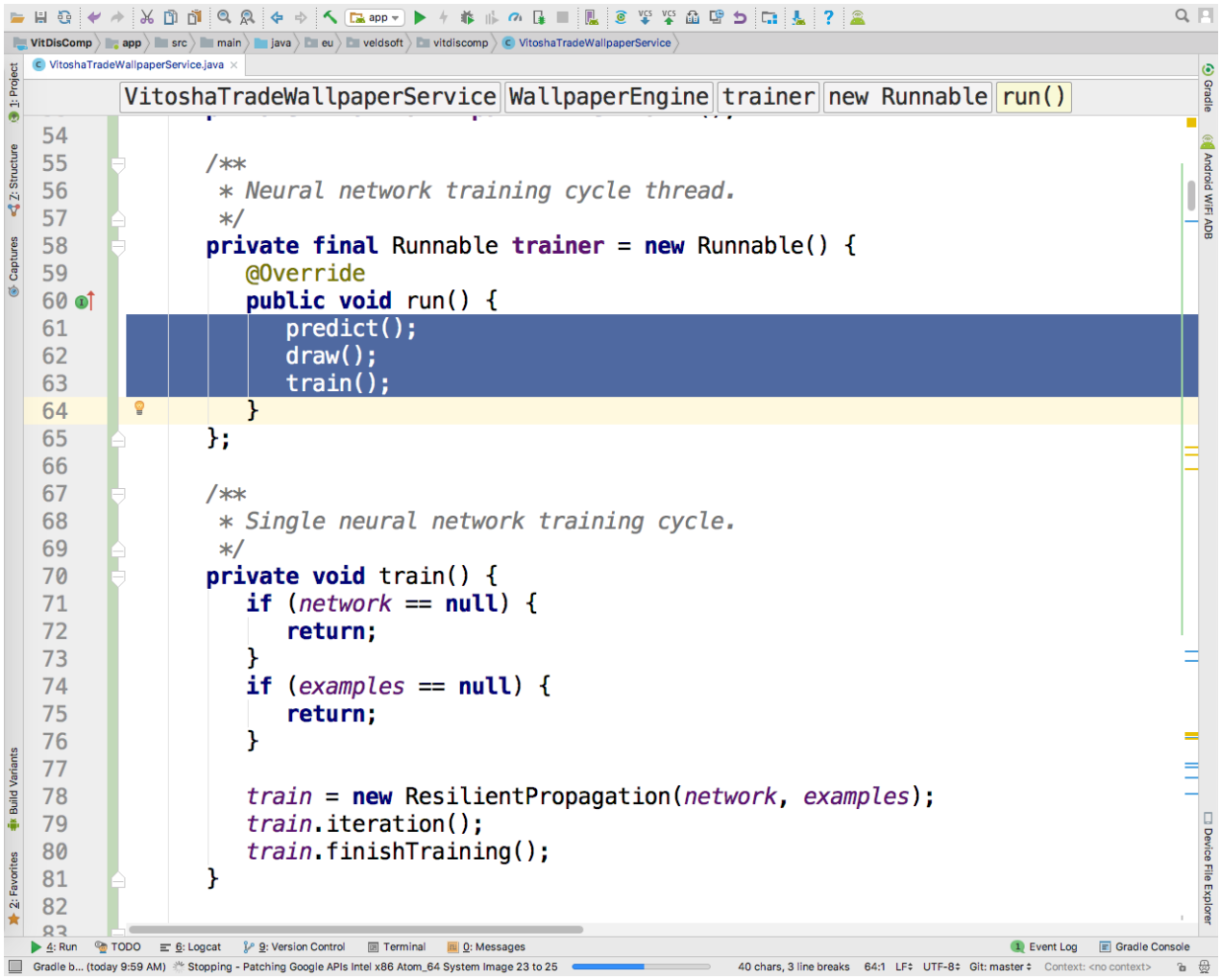
4.4 Двигател на услугата

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



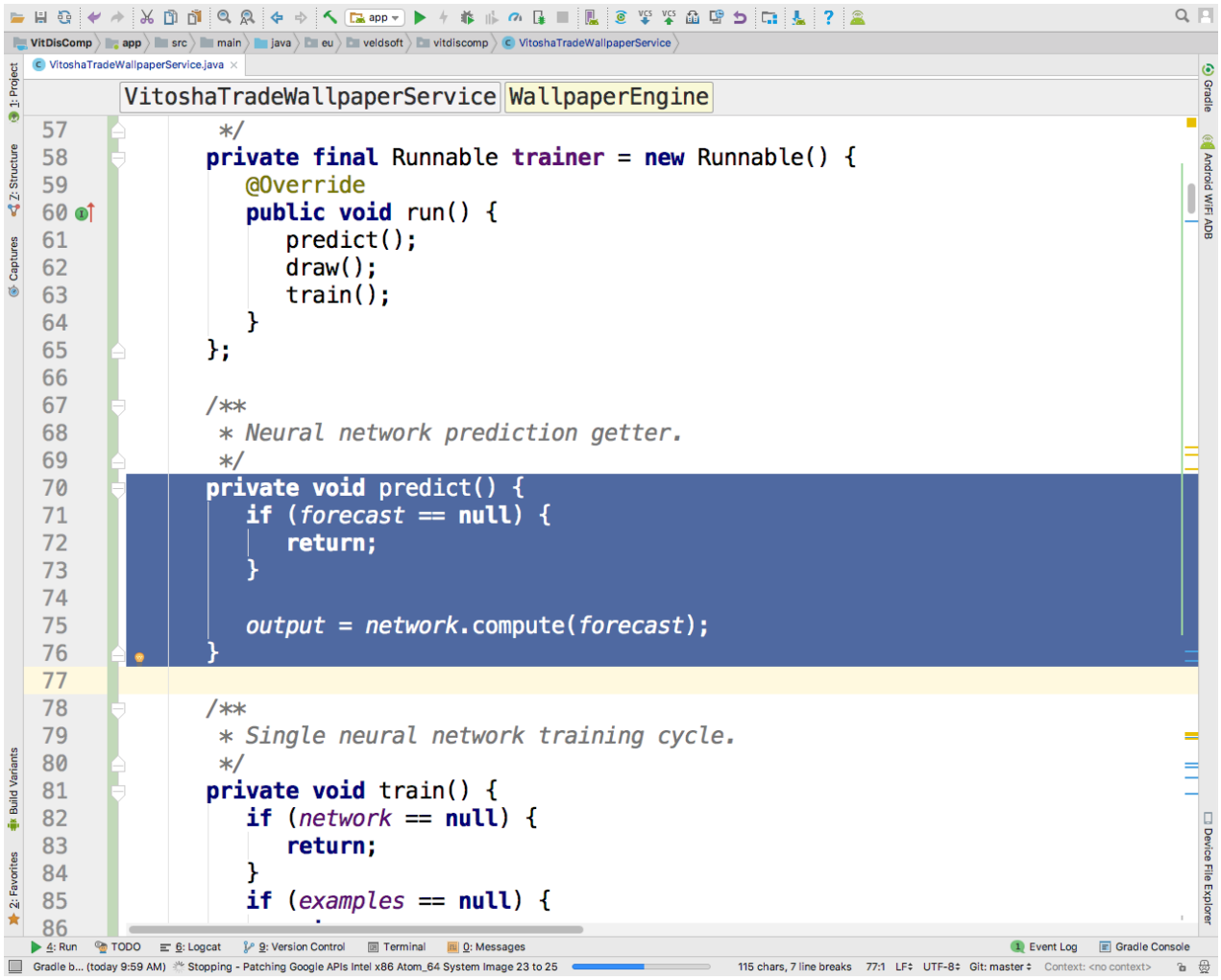
Фигура 4.37: Вътрешни променливи за двигателя на услугата

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 изрисуване. Ползата е, че обектът бива създаден веднъж и след това се ползва през целия жизнен цикъл на услугата.



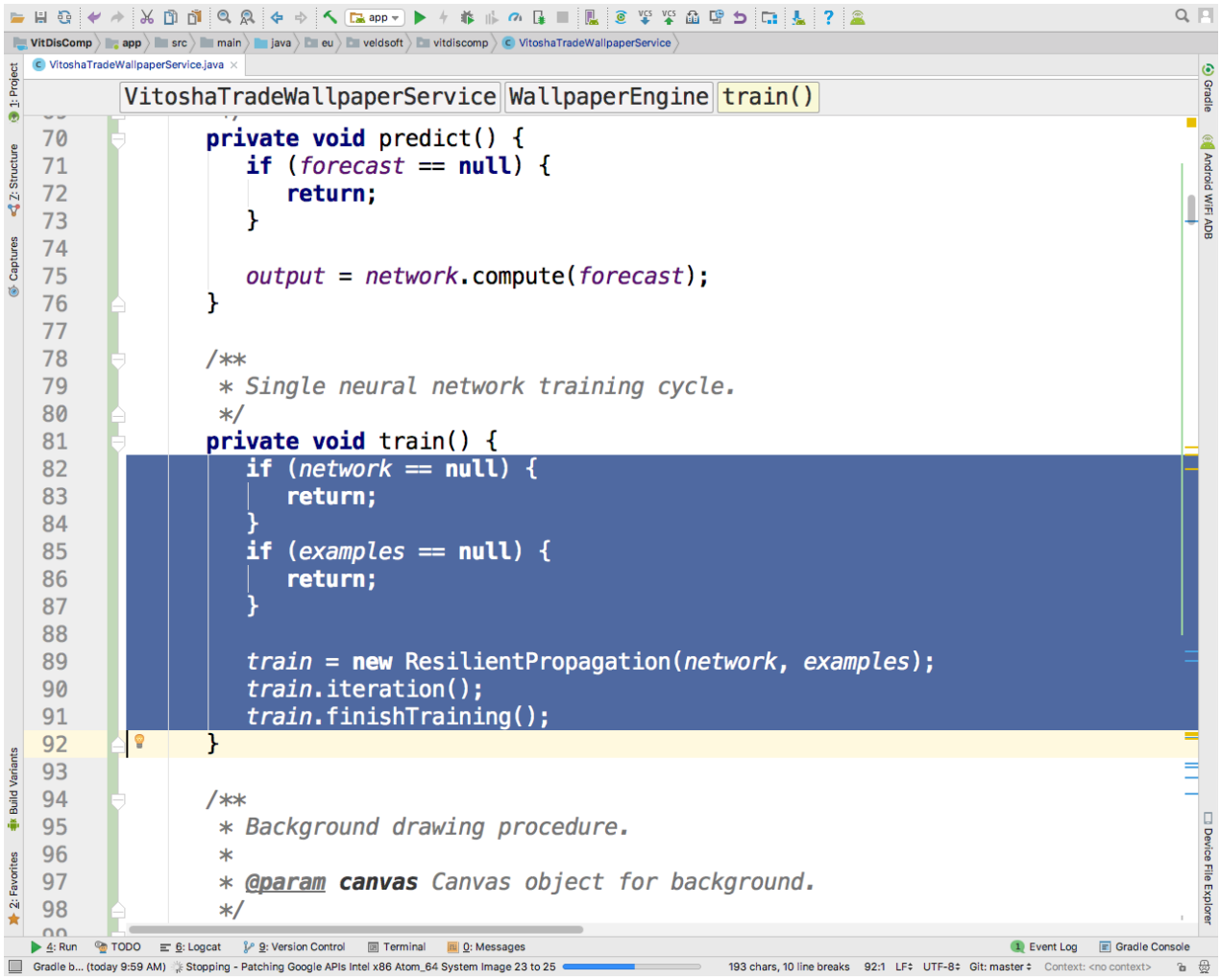
Фигура 4.38: Единичен цикъл във фонов режим

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.



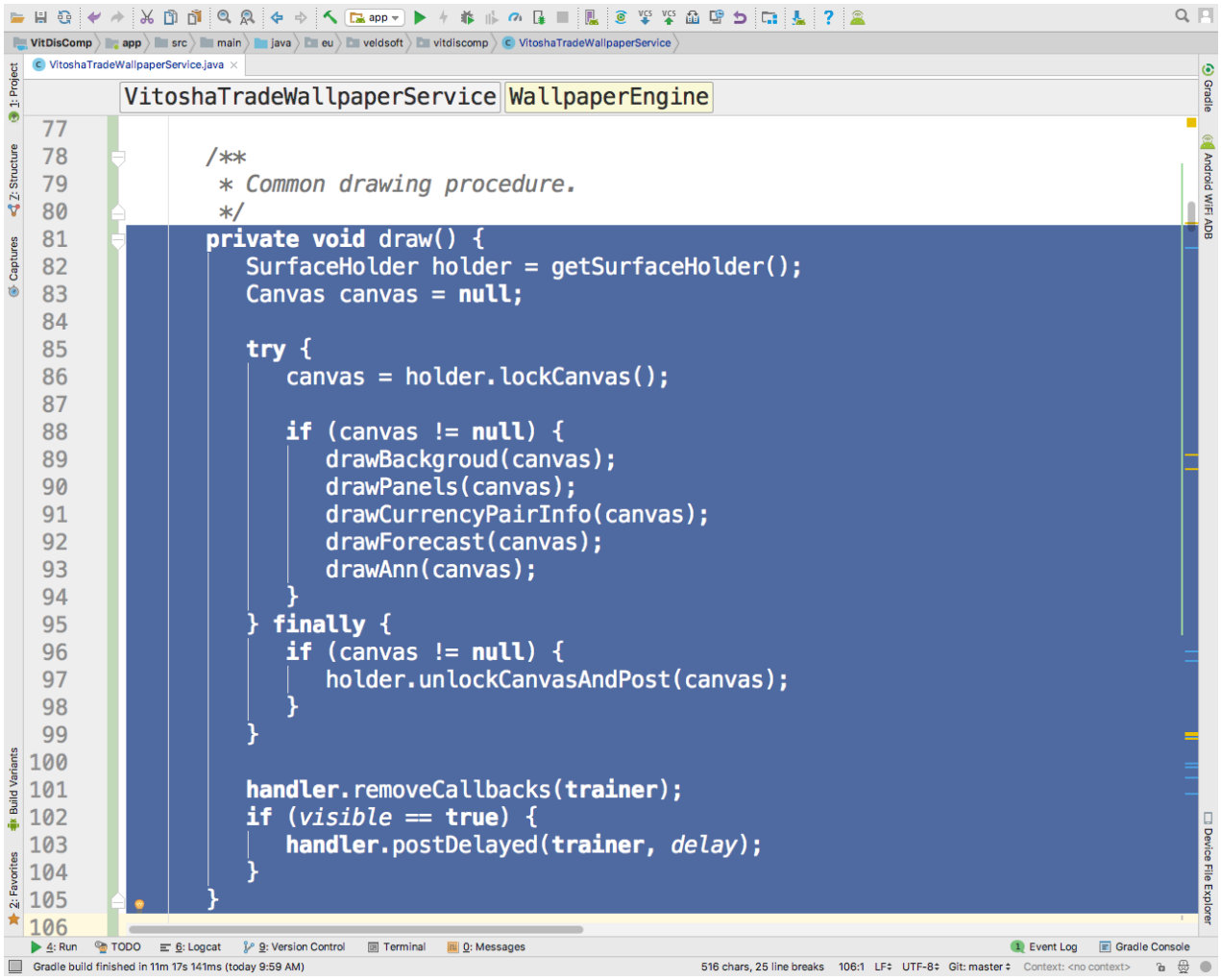
Фигура 4.39: Изчисляване на прогноза

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).



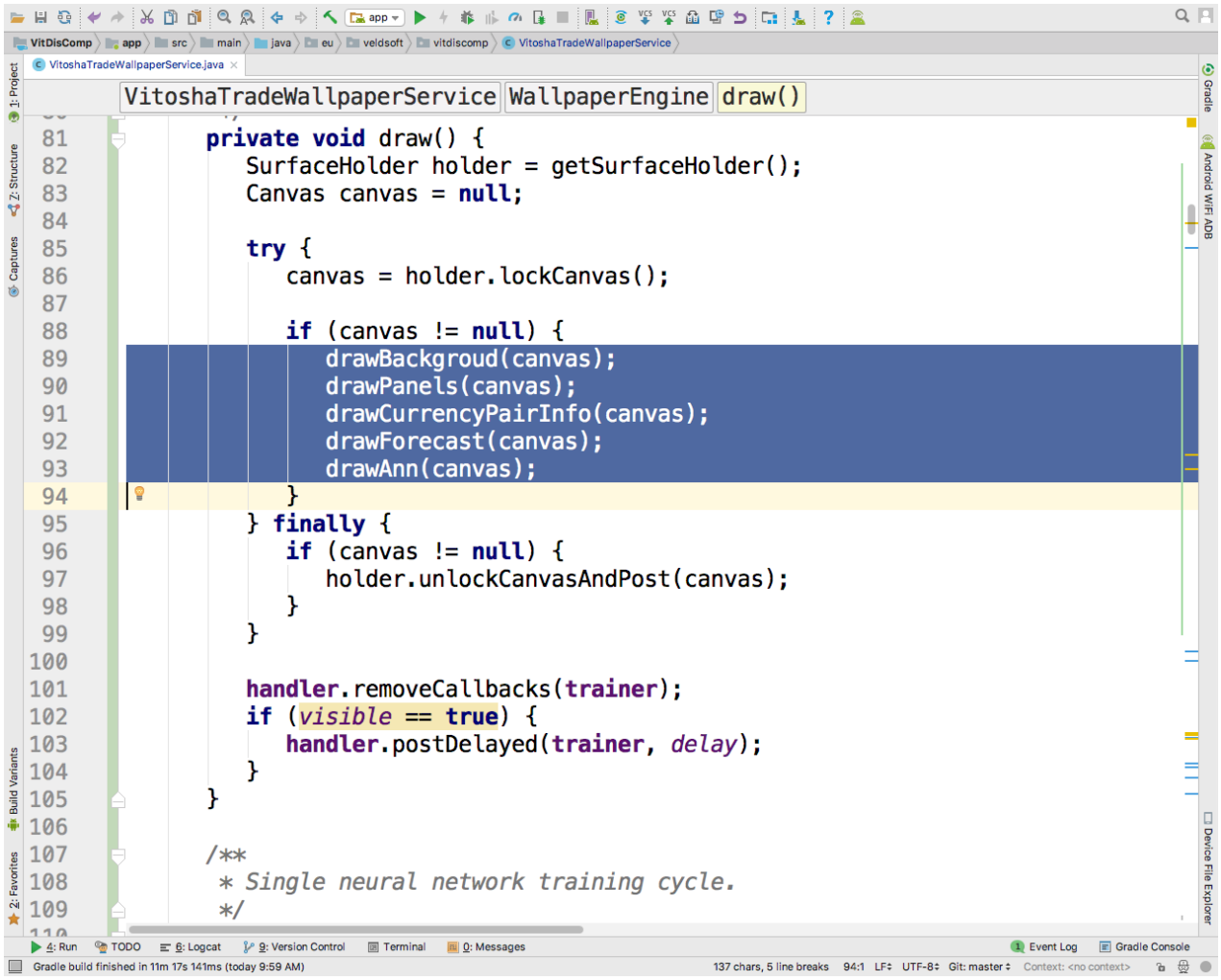
Фигура 4.40: Тренировъчен цикъл на изкуствената невронна мрежа

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).



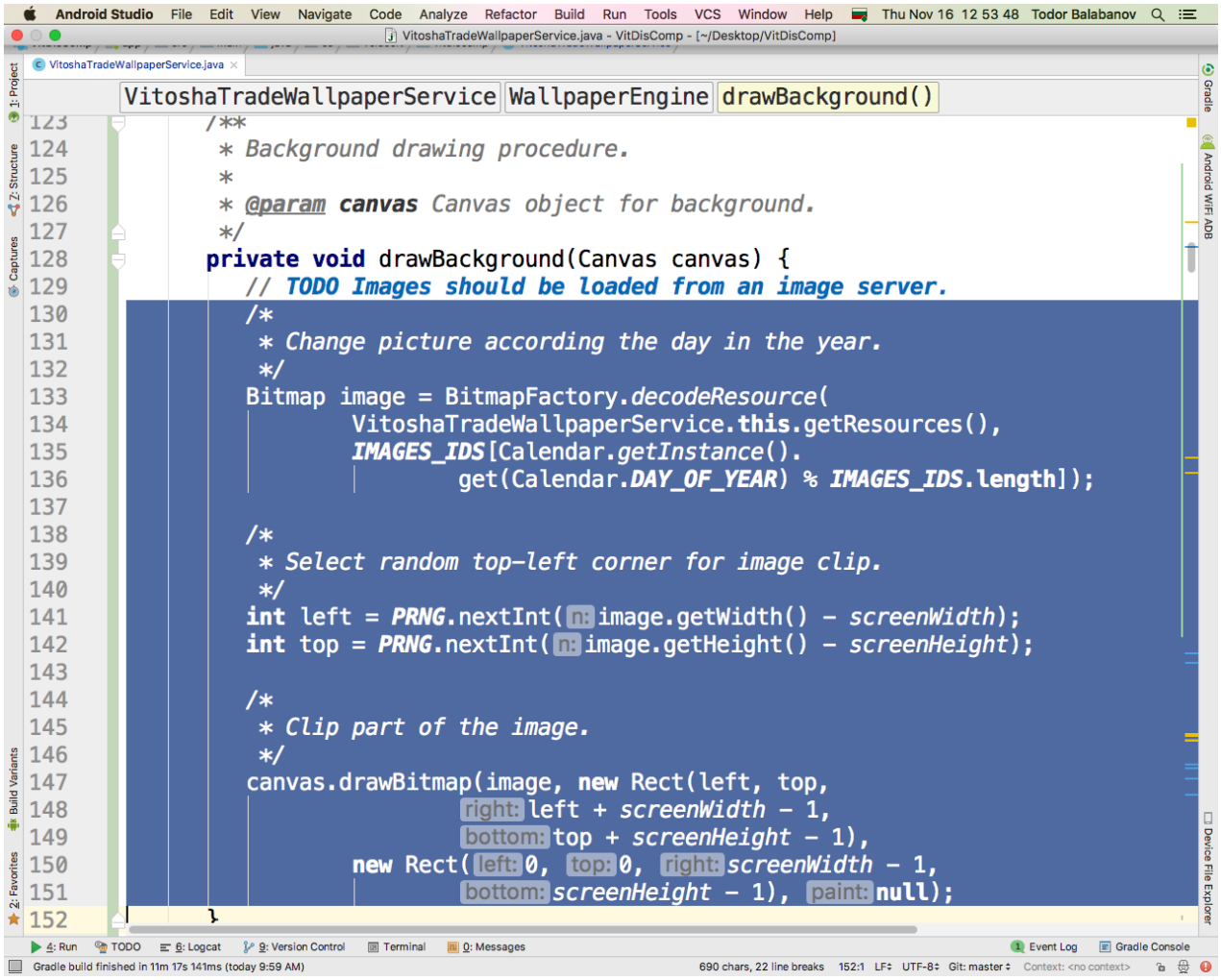
Фигура 4.41: Основна процедура по изрисуване на информацията от тренировъчния процес

Изрисуването 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.



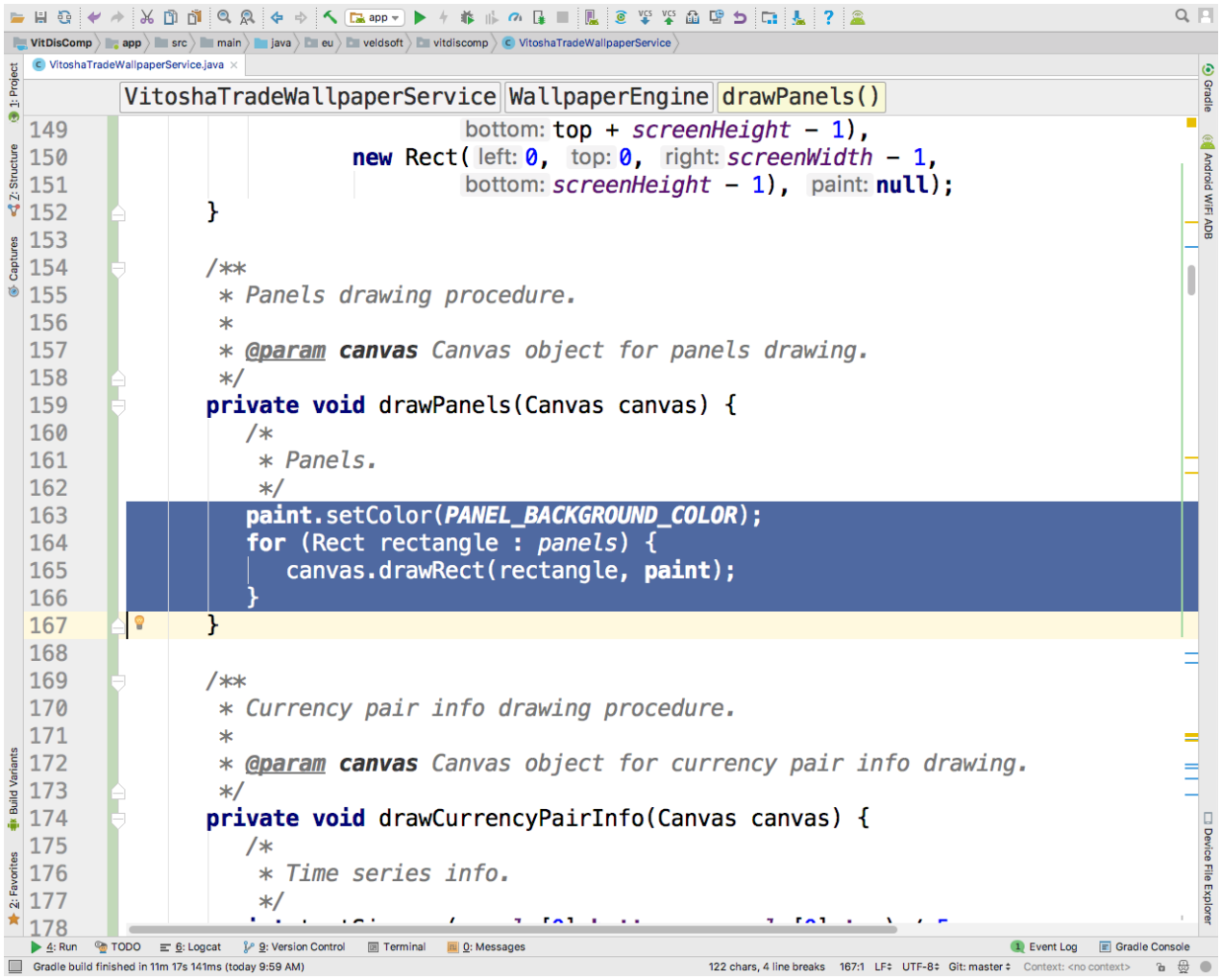
Фигура 4.42: Разбиване на задачата за визуално представяне

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.



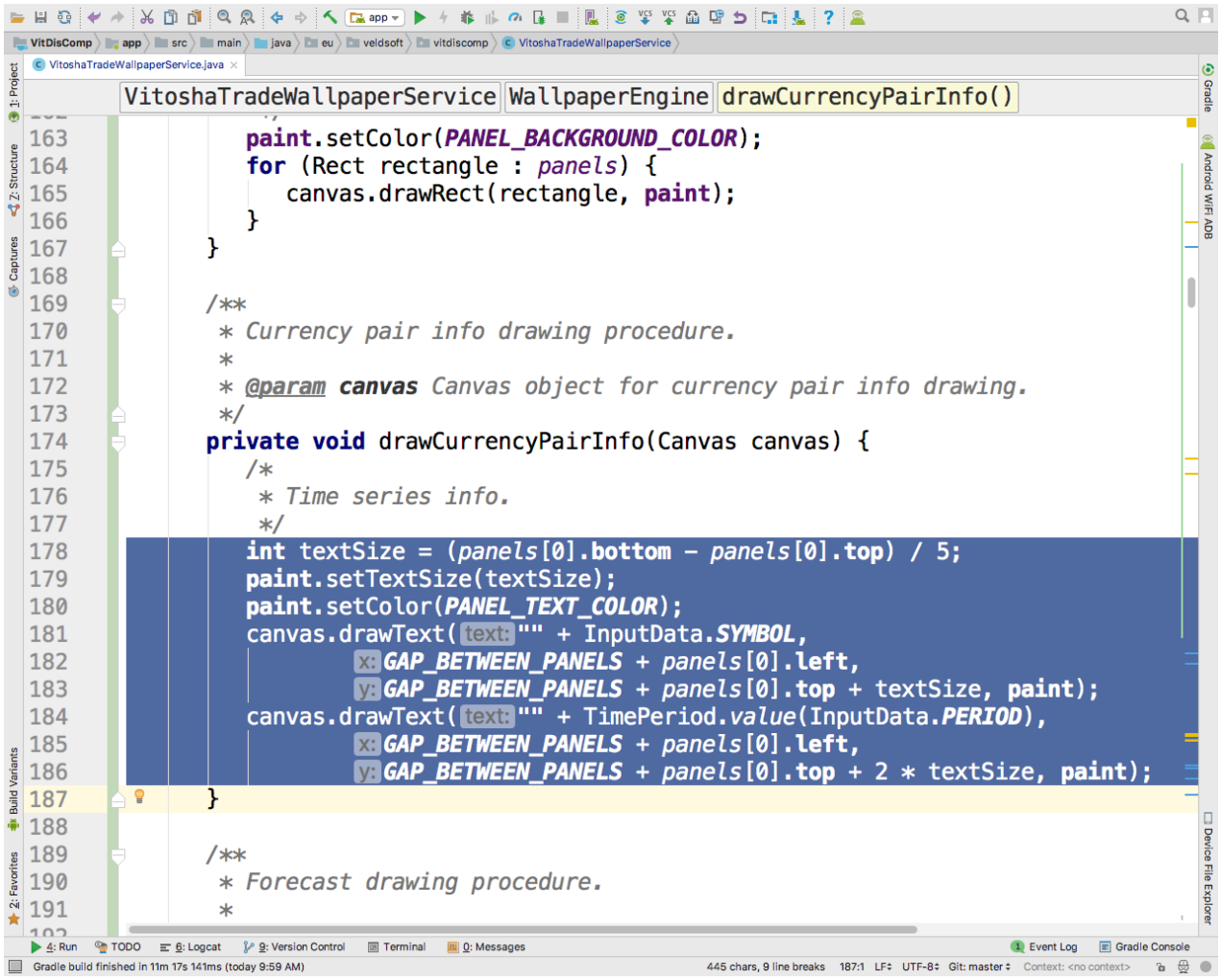
Фигура 4.43: Изрисуване на фона

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.



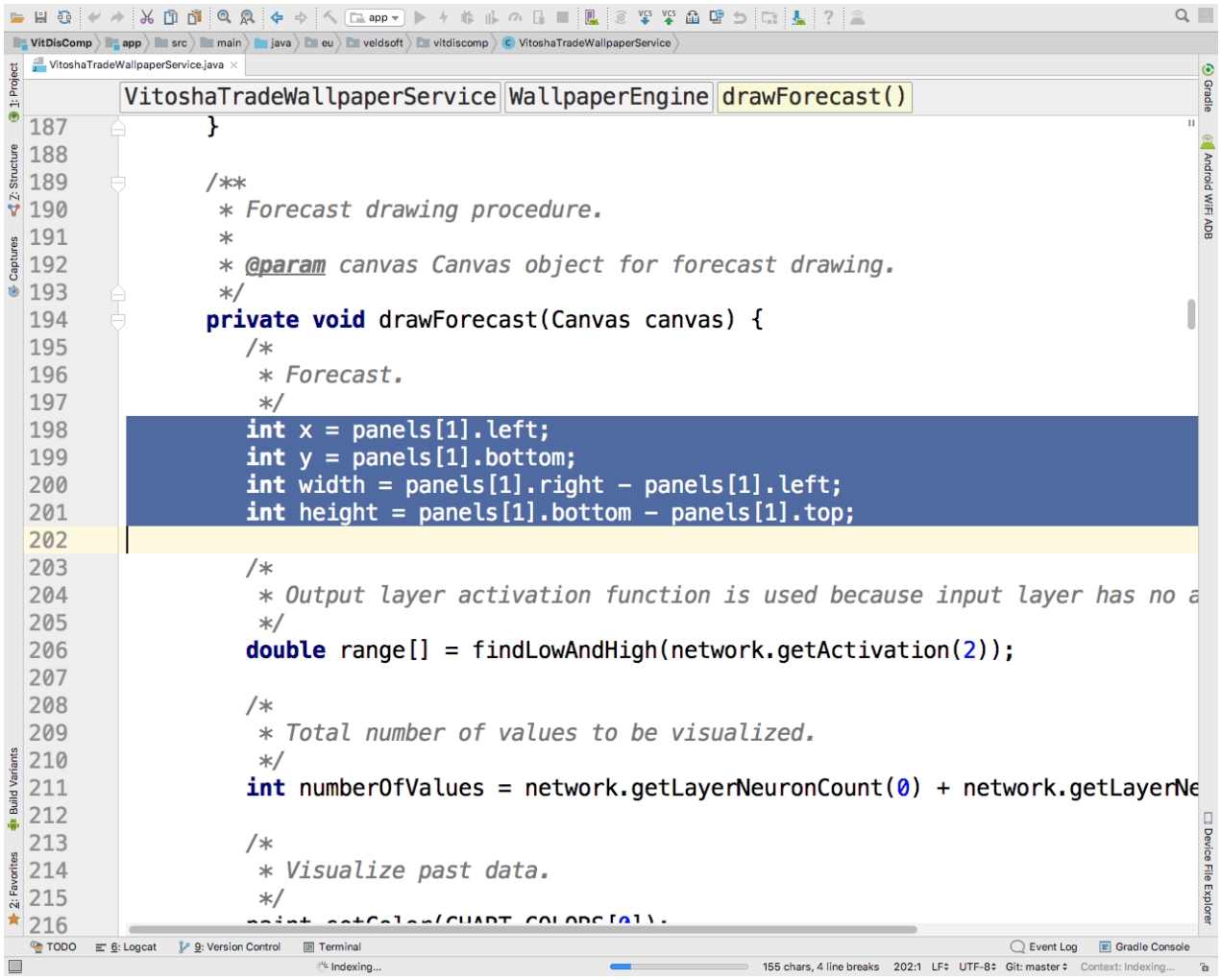
Фигура 4.44: Изрисуване на областите за служебна информация

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.



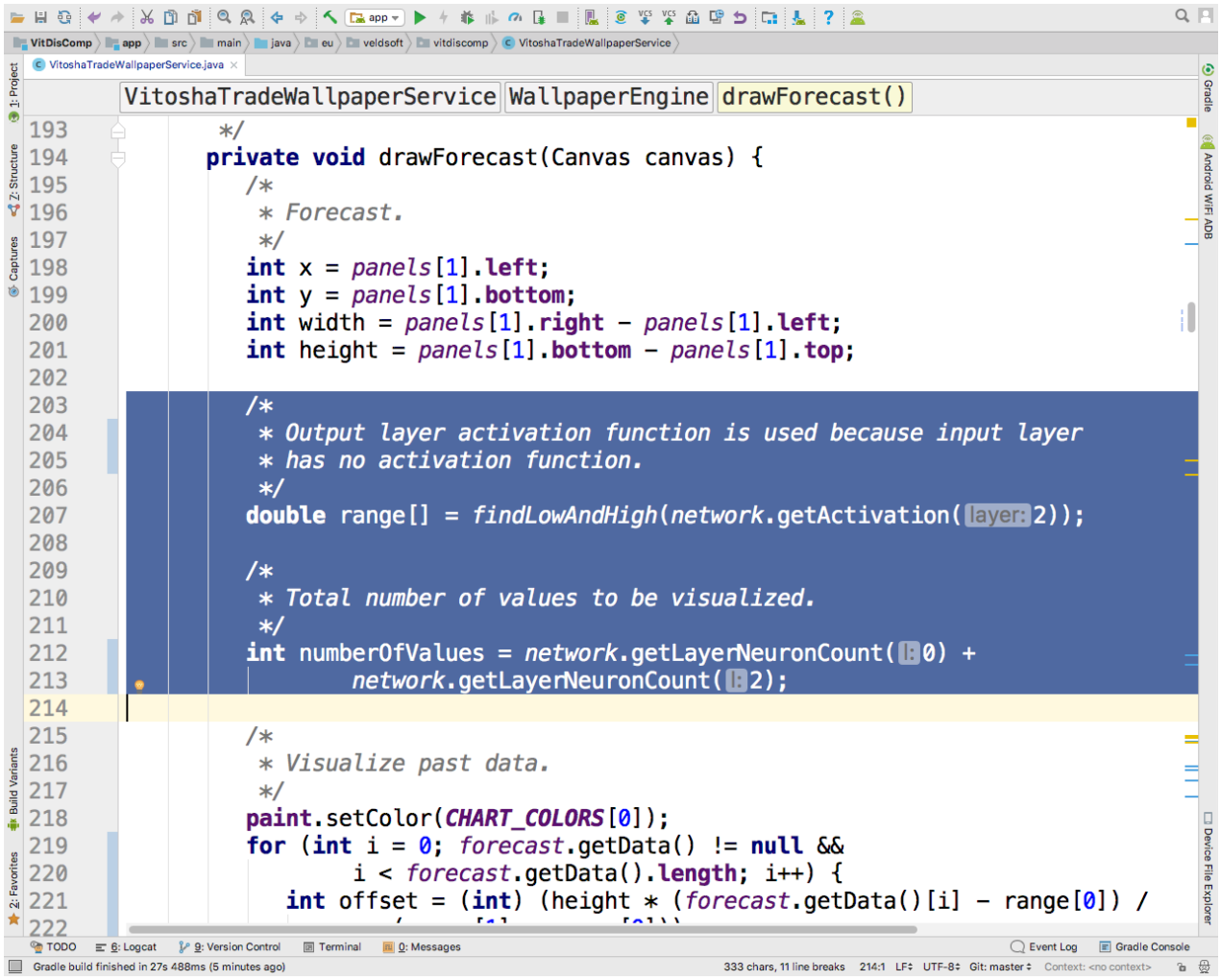
Фигура 4.45: Изрисуване на информацията за валутната двойка

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



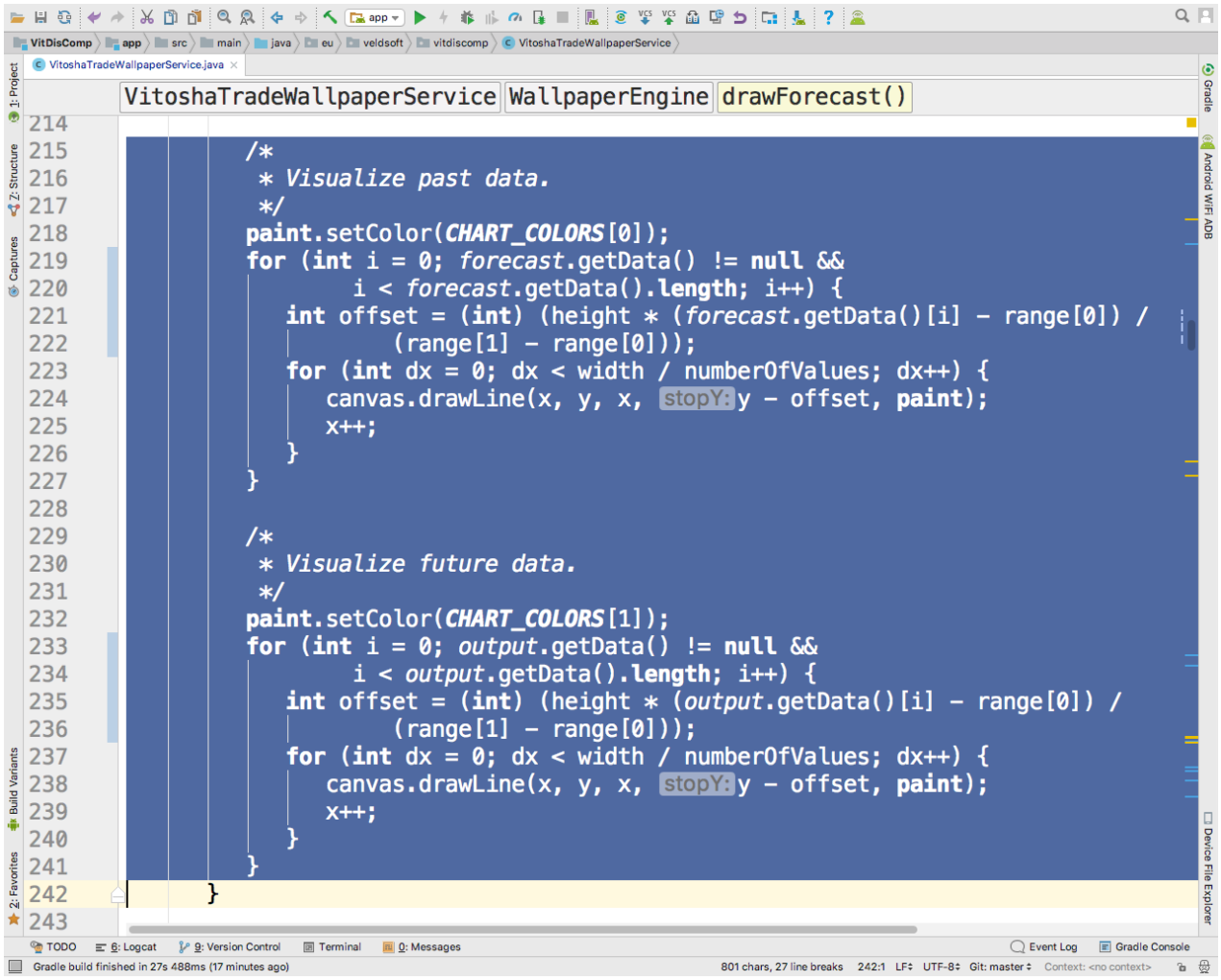
Фигура 4.46: Позиция и размери за визуално представяне на прогнозата

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).



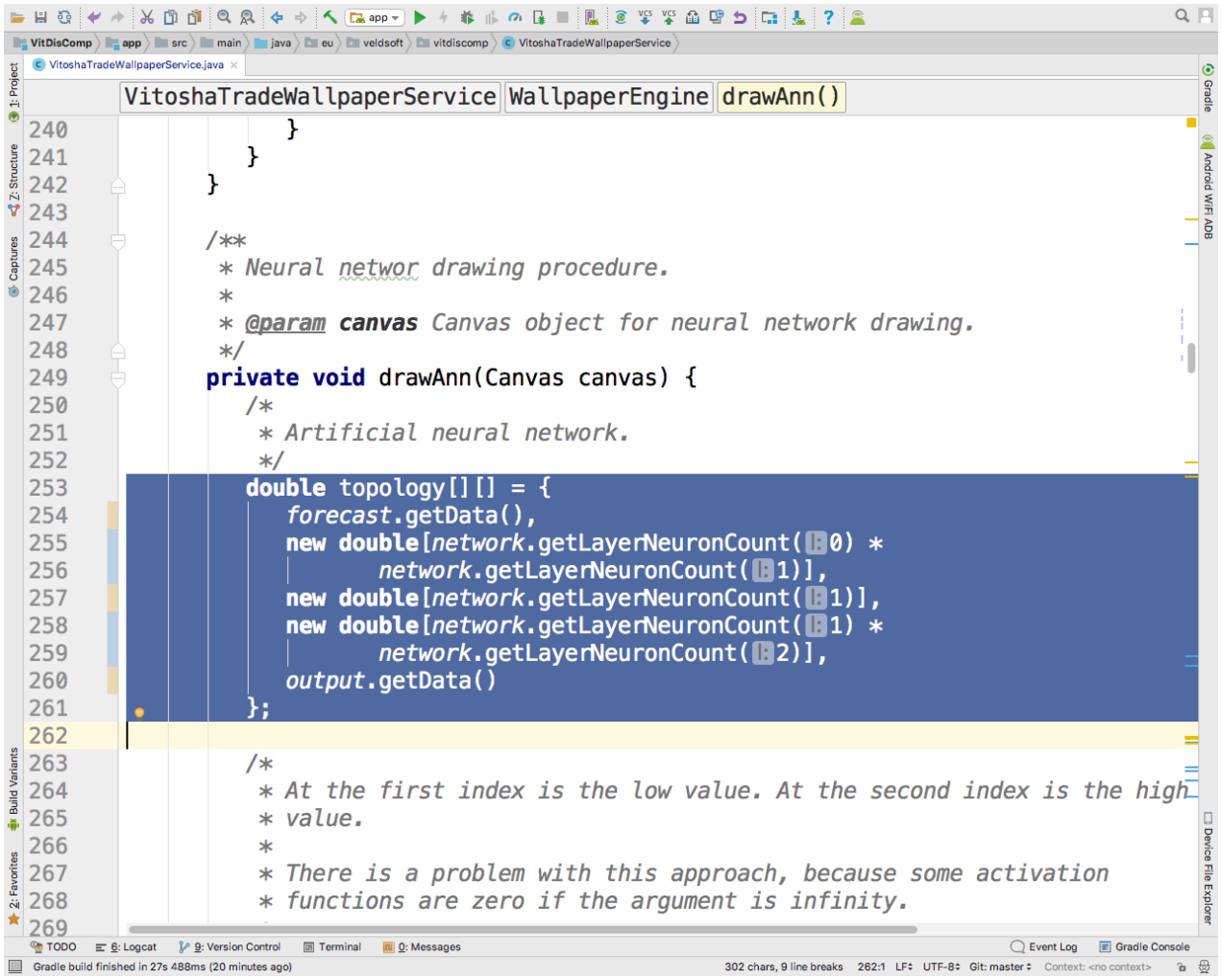
Фигура 4.47: Обхват на данните по ширина и височина

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.



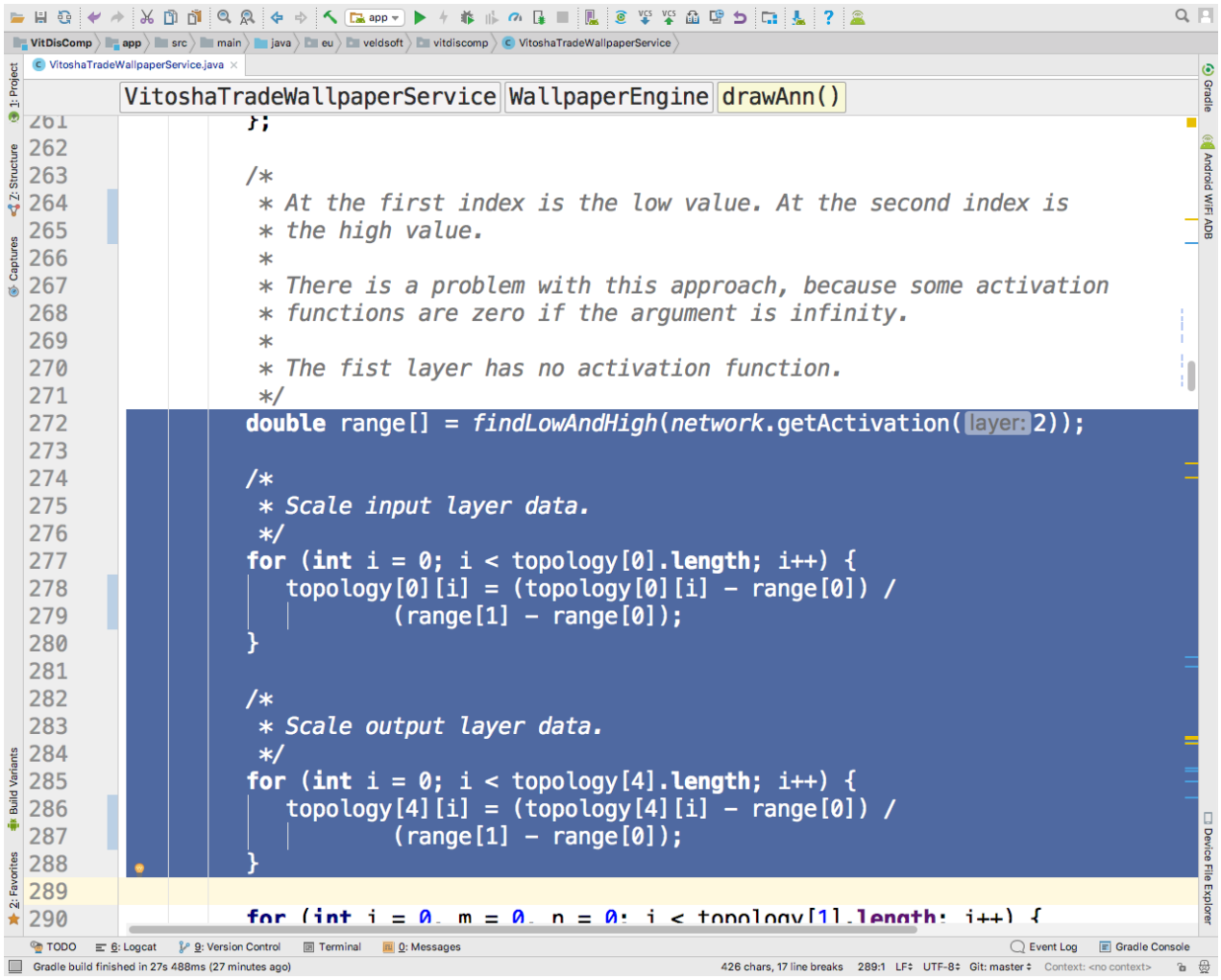
Фигура 4.48: Цикли за визуално представяне на стълбовете от времевия реди и прогнозата

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.



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

В третата област за визуално представяне се изрисува 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 скритя слой; 3. Стойности на невроните в скритя слой; 4. Стойности на теглата между скритя и изходния слой; 5. Стойности на невроните в изходния слой (Фиг. 4.49).

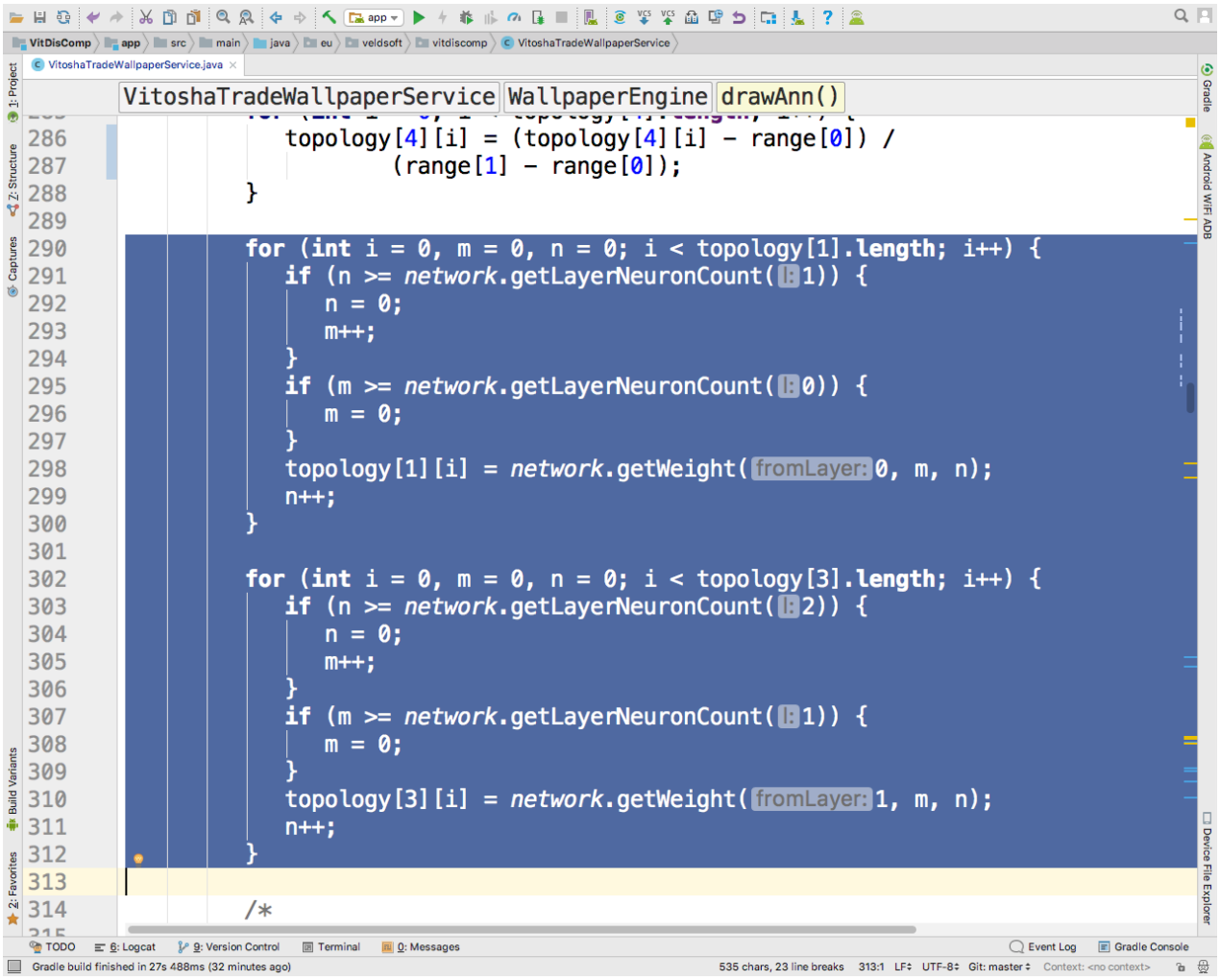


Фигура 4.50: Мащабиране на входната и изходната информация

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).

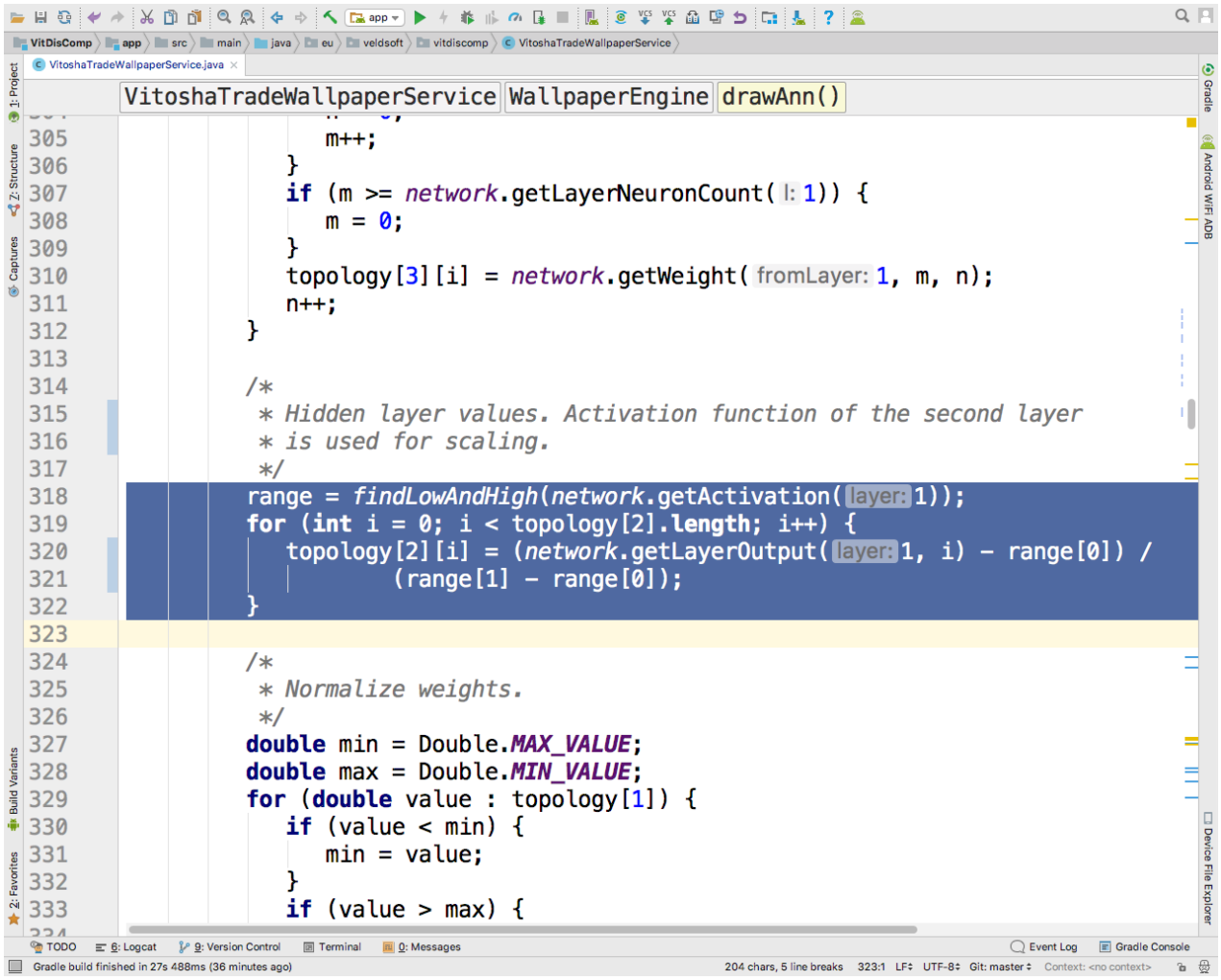
$$X_{sc} = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{4.1}$$

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



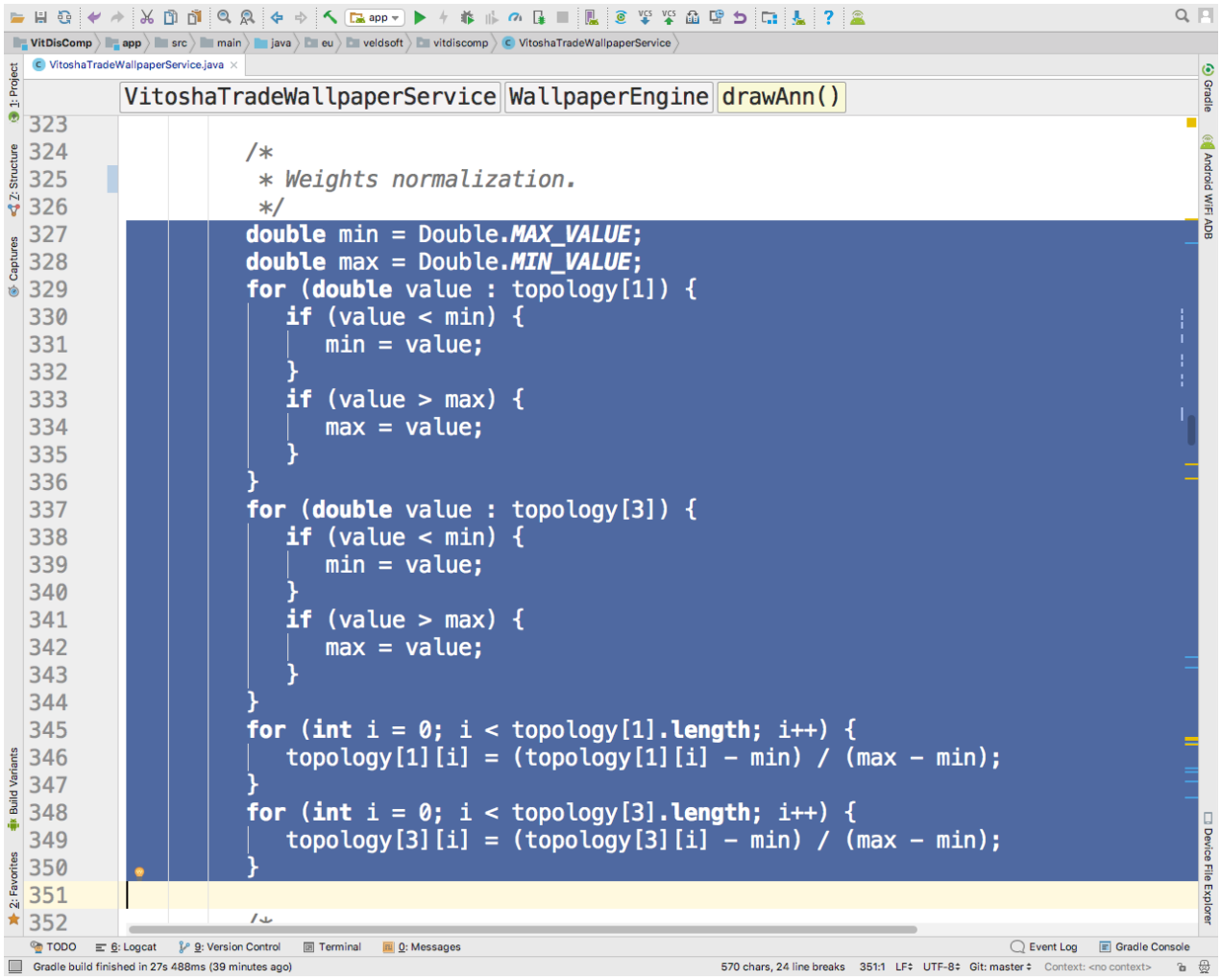
Фигура 4.51: Стойности на теглата между слоевете

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



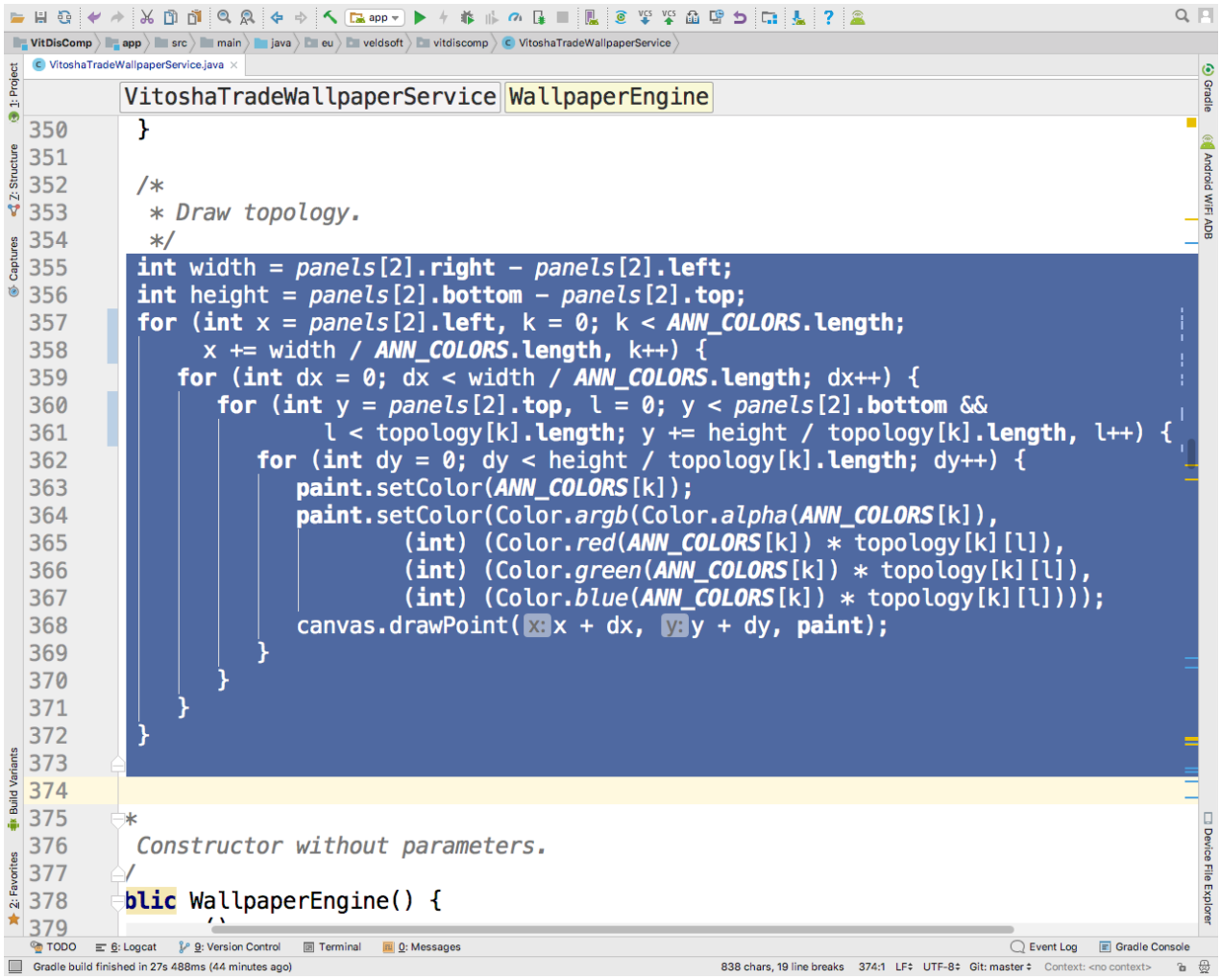
Фигура 4.52: Мащабиране на стойностите в скрития слой

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).



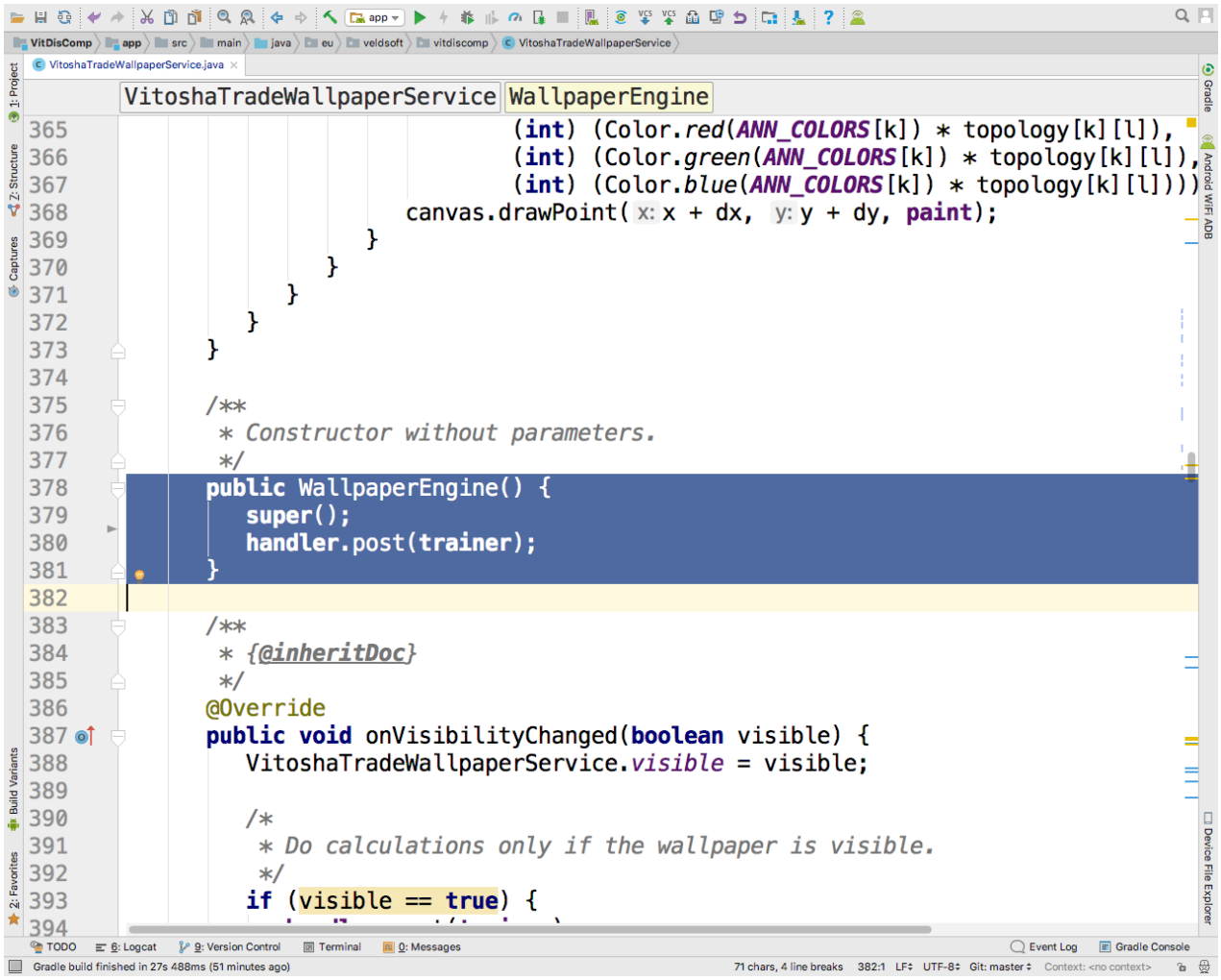
Фигура 4.53: Нормализиране на стойностите за теглата

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).



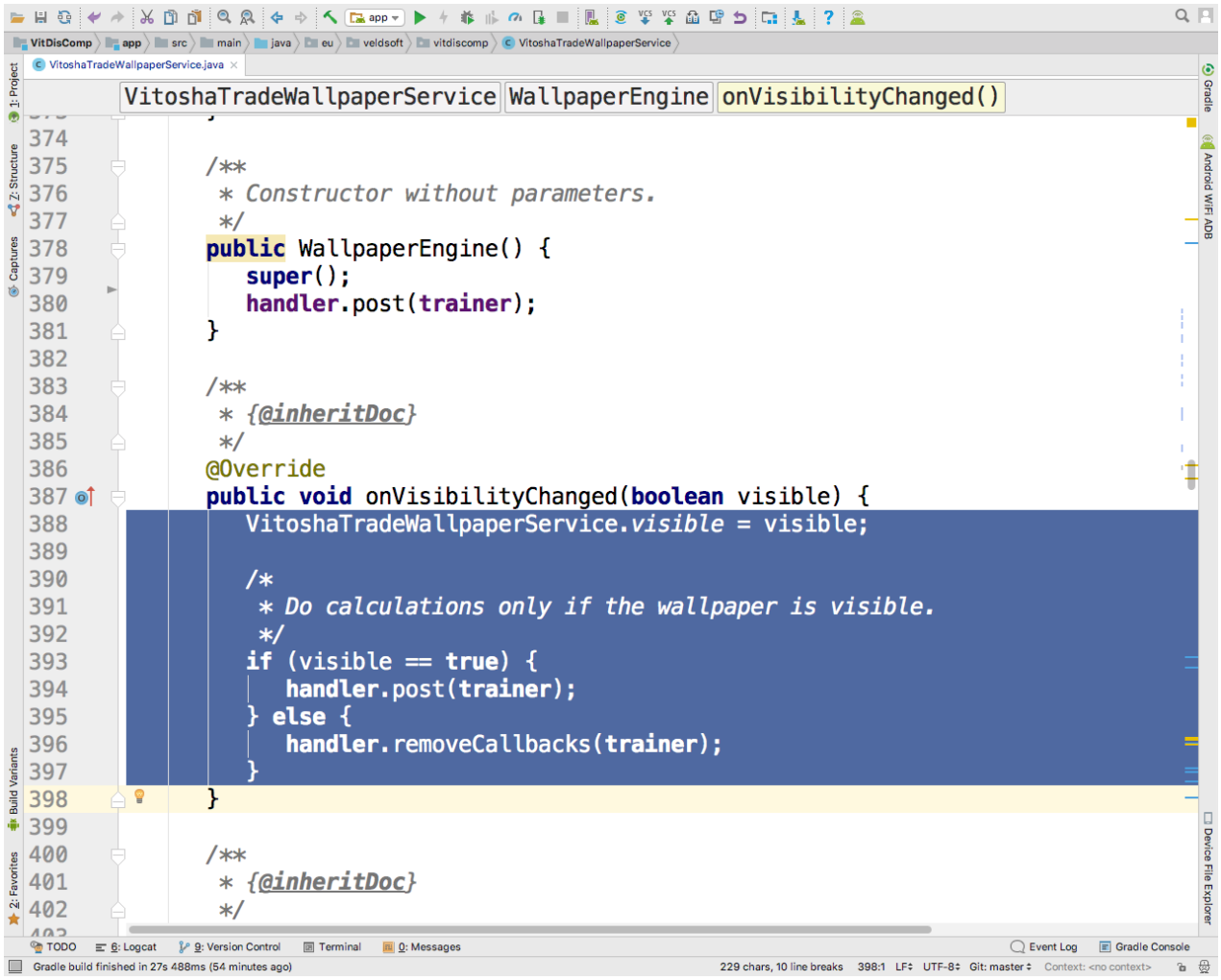
Фигура 4.54: Изрисуване на топологията

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



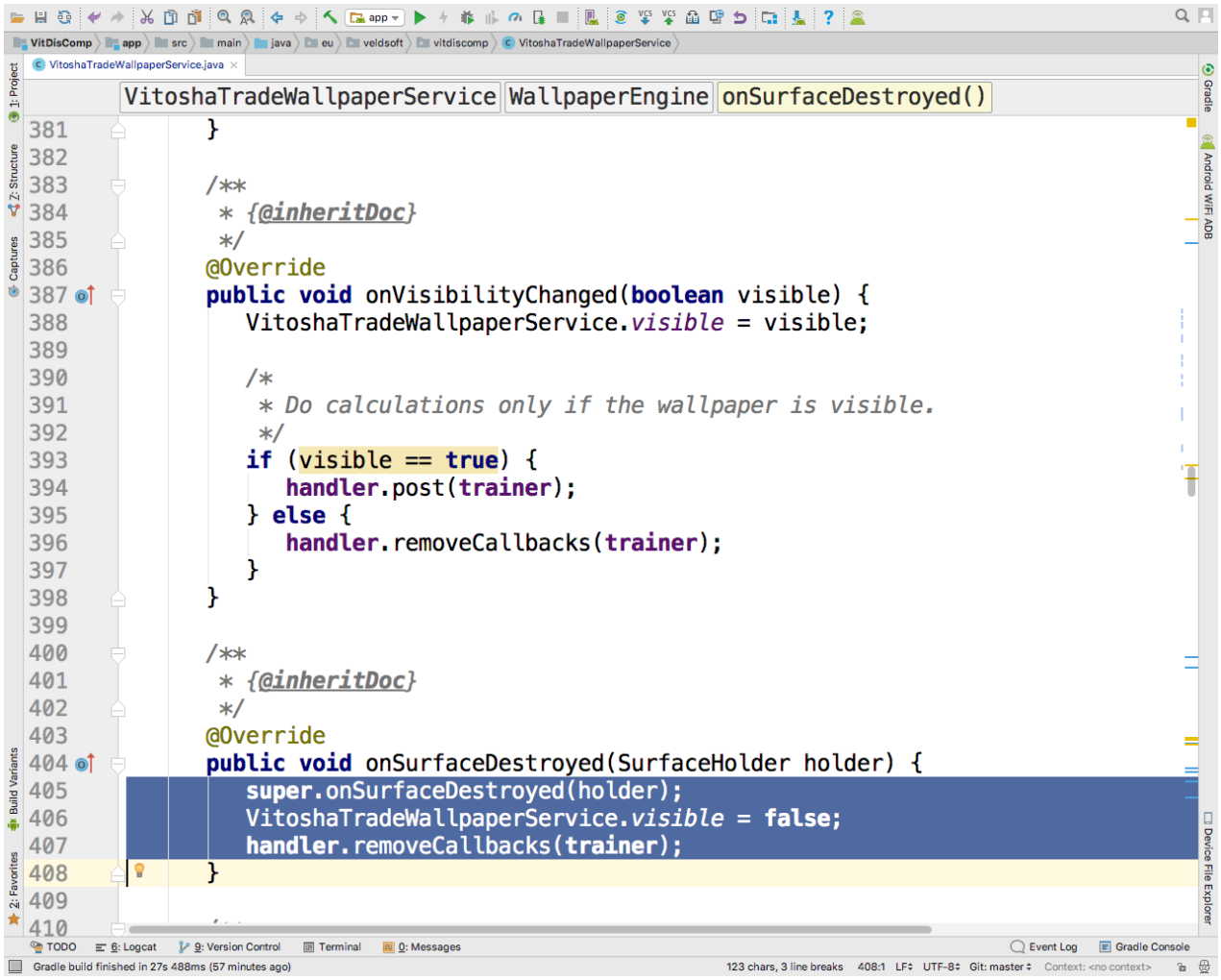
Фигура 4.55: Конструктор на двигателя за услугата

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



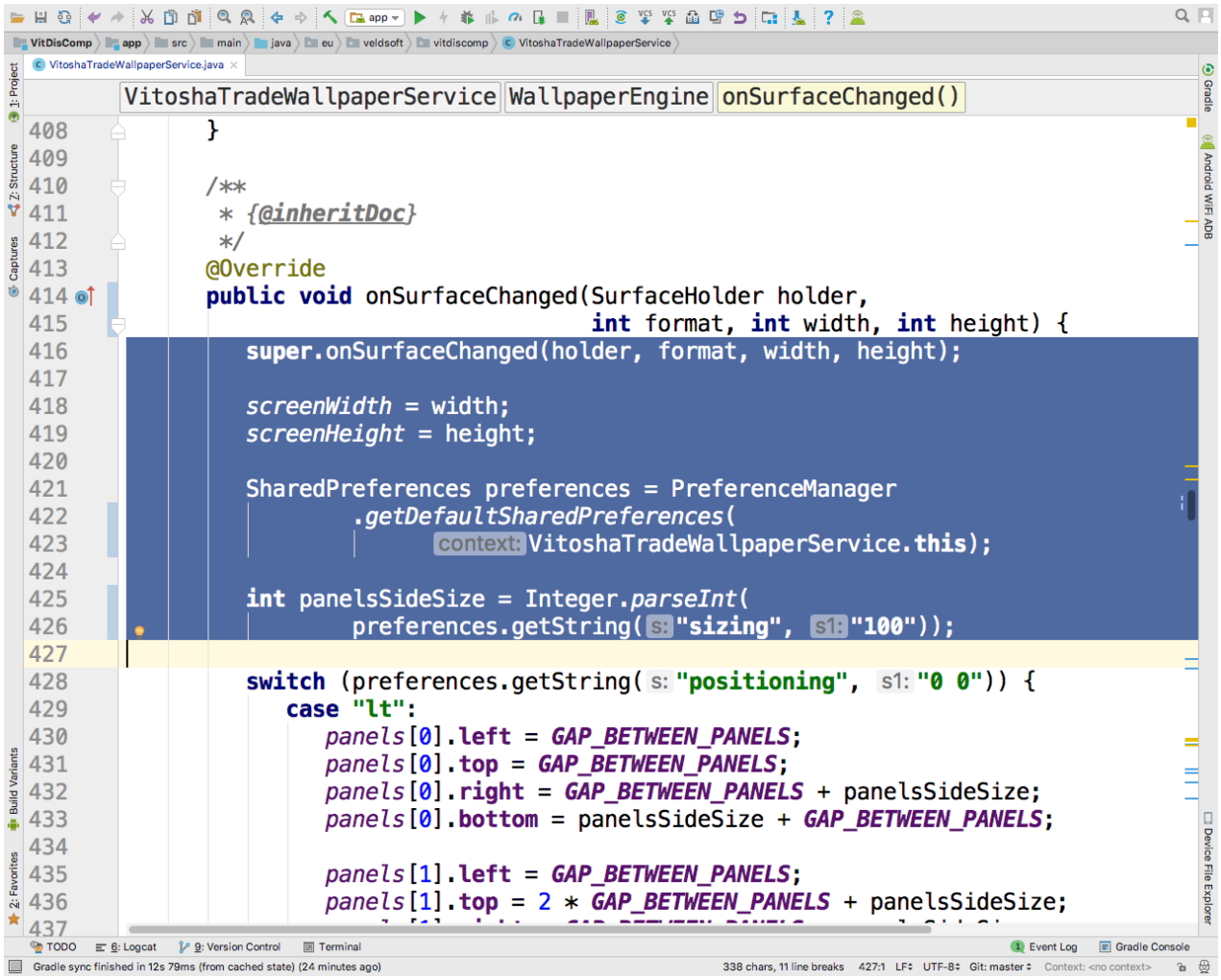
Фигура 4.56: Промяна във видимостта на активния тапет

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).



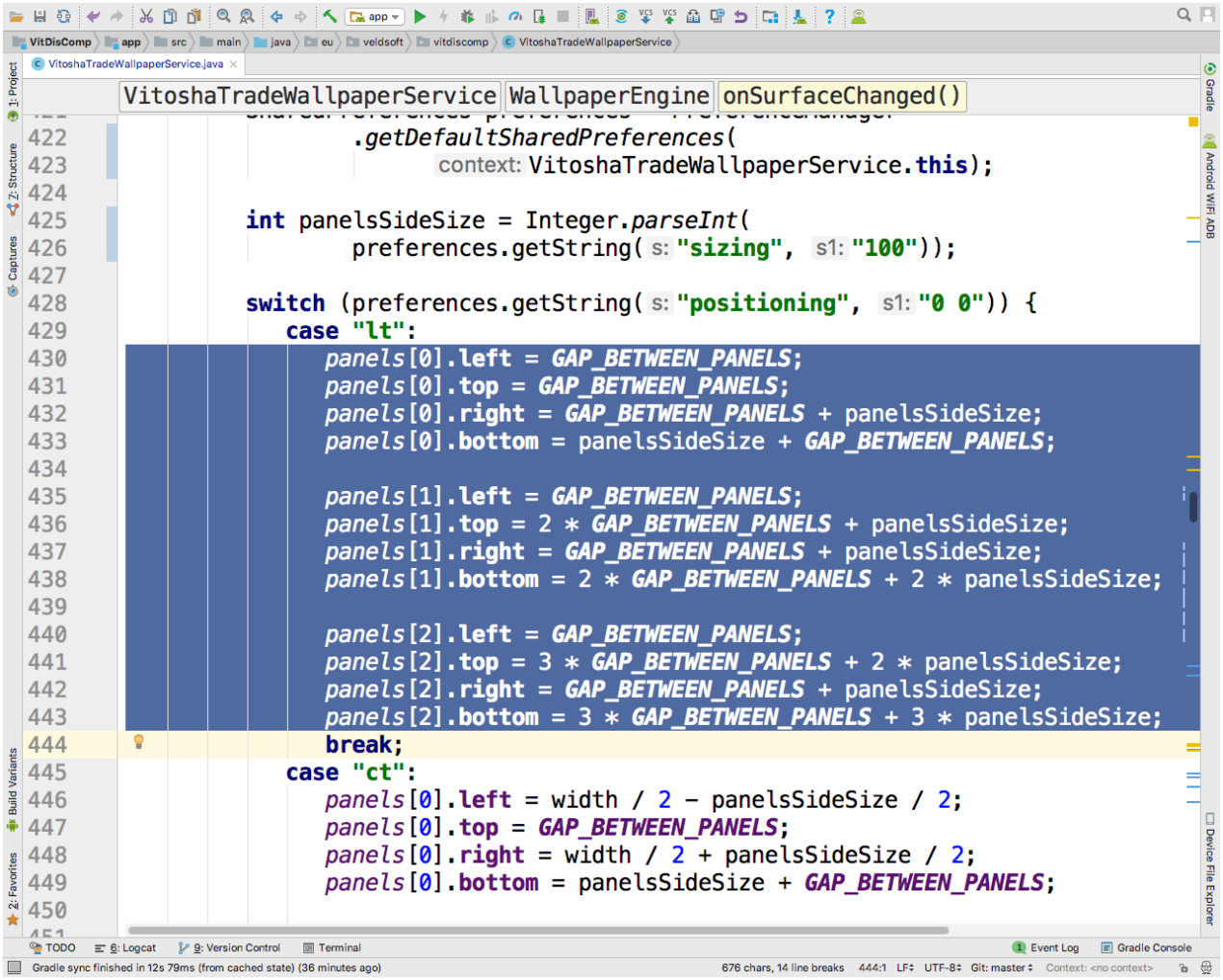
Фигура 4.57: Спиране на пресмятанията при разрушаване на повърхността за изрисуване

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

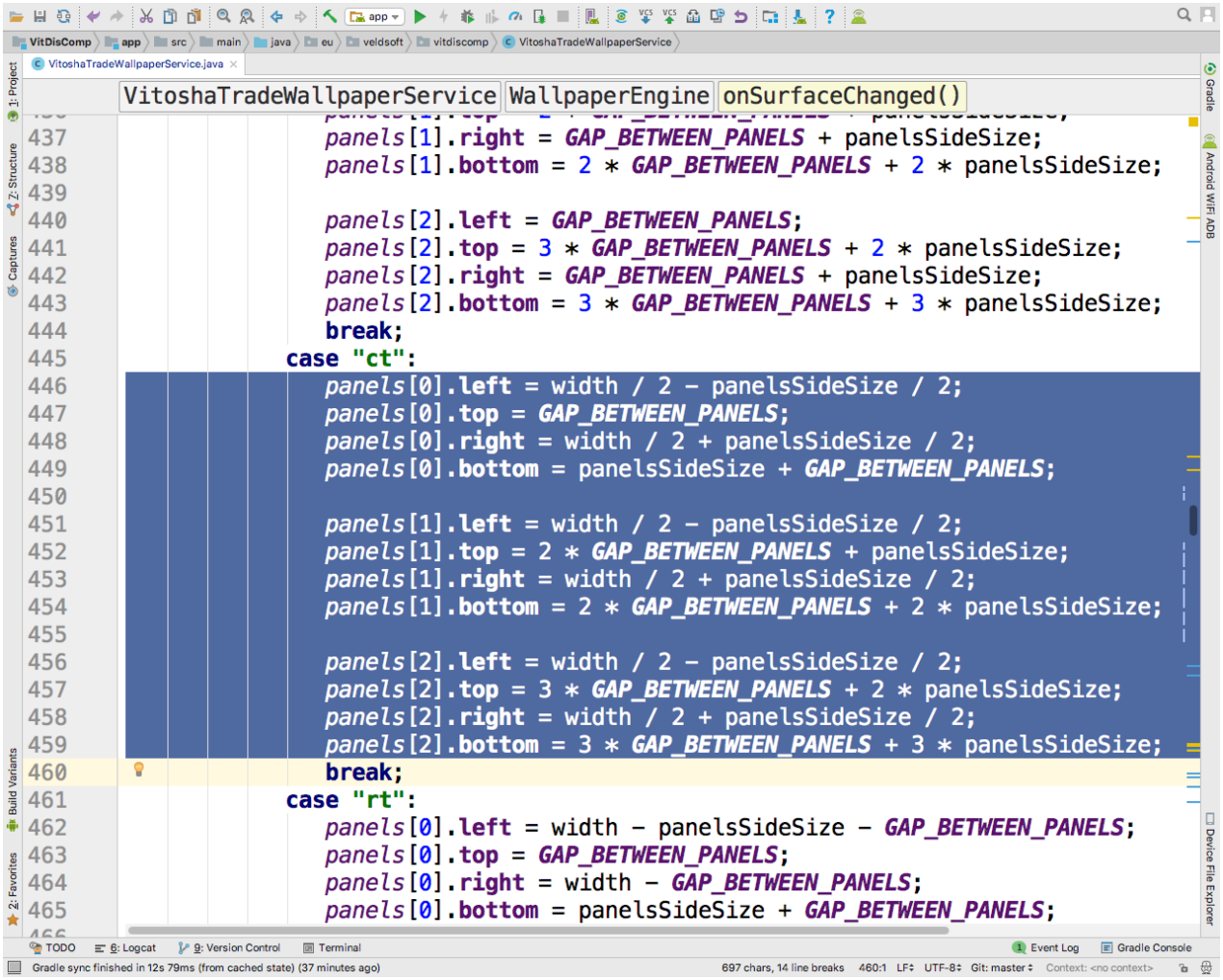


Фигура 4.58: Размер на зоните за изрисуване

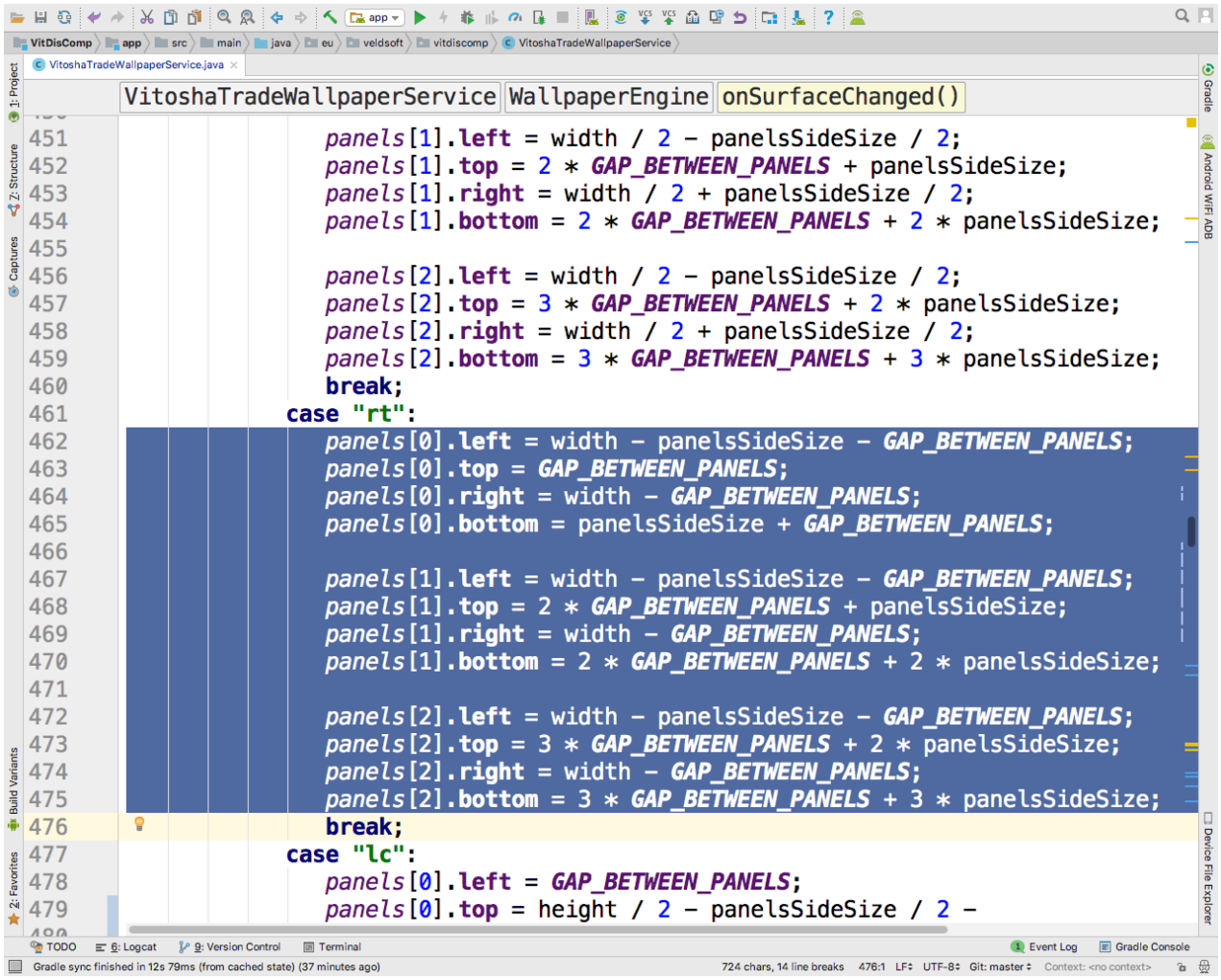
Когато настъпи промяна в площта за изрисуване на активния тапет, е нужно да се вземат серия мерки по преизчисляване на размерите. Такова събитие примерно настъпва, когато устройството сменя визуалното представане от портретна в пейзажна или обратното. Първата характеристика която се следи е размерът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



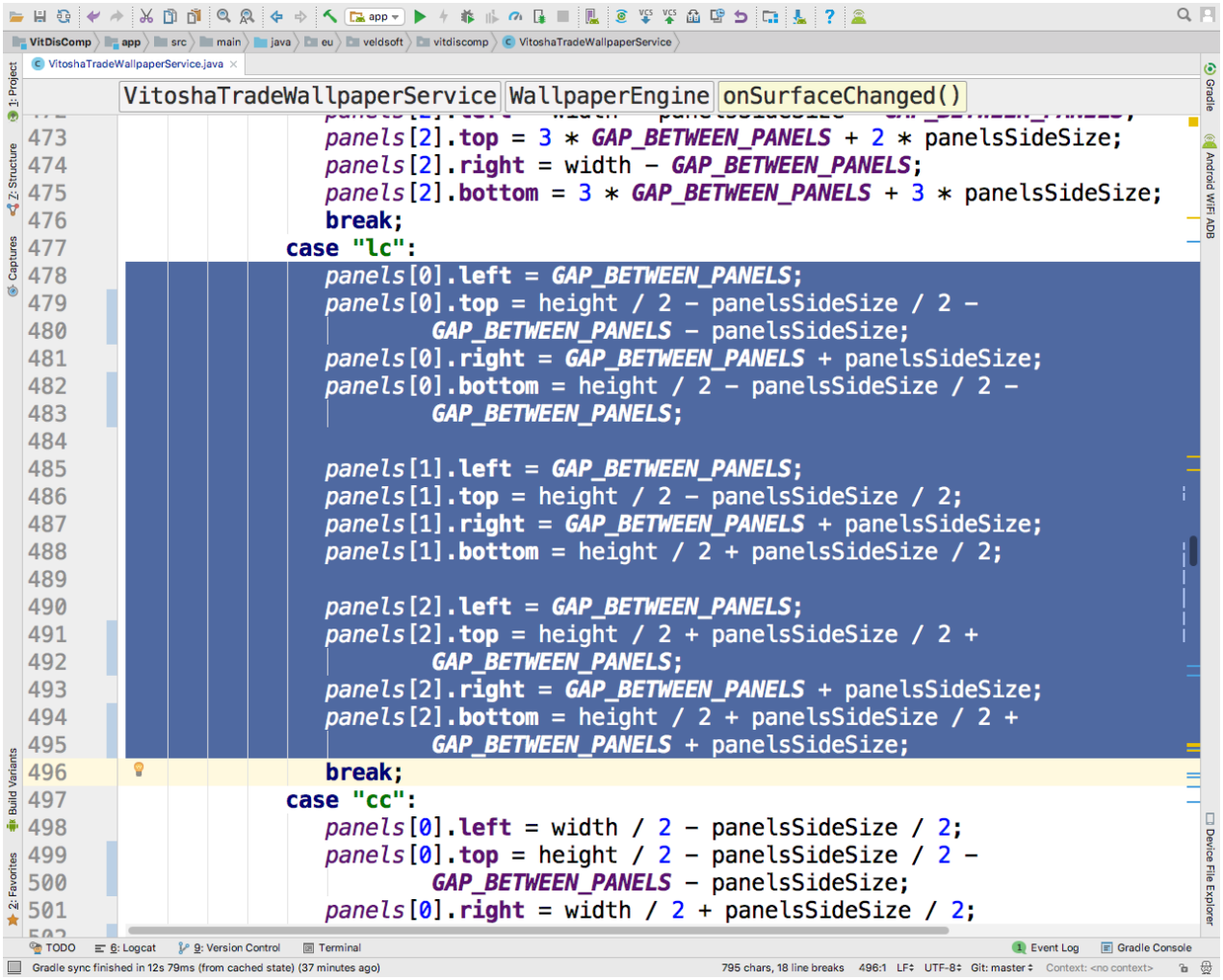
Фигура 4.59: Координати и размери на областите за визуално представяне - горе-ляво



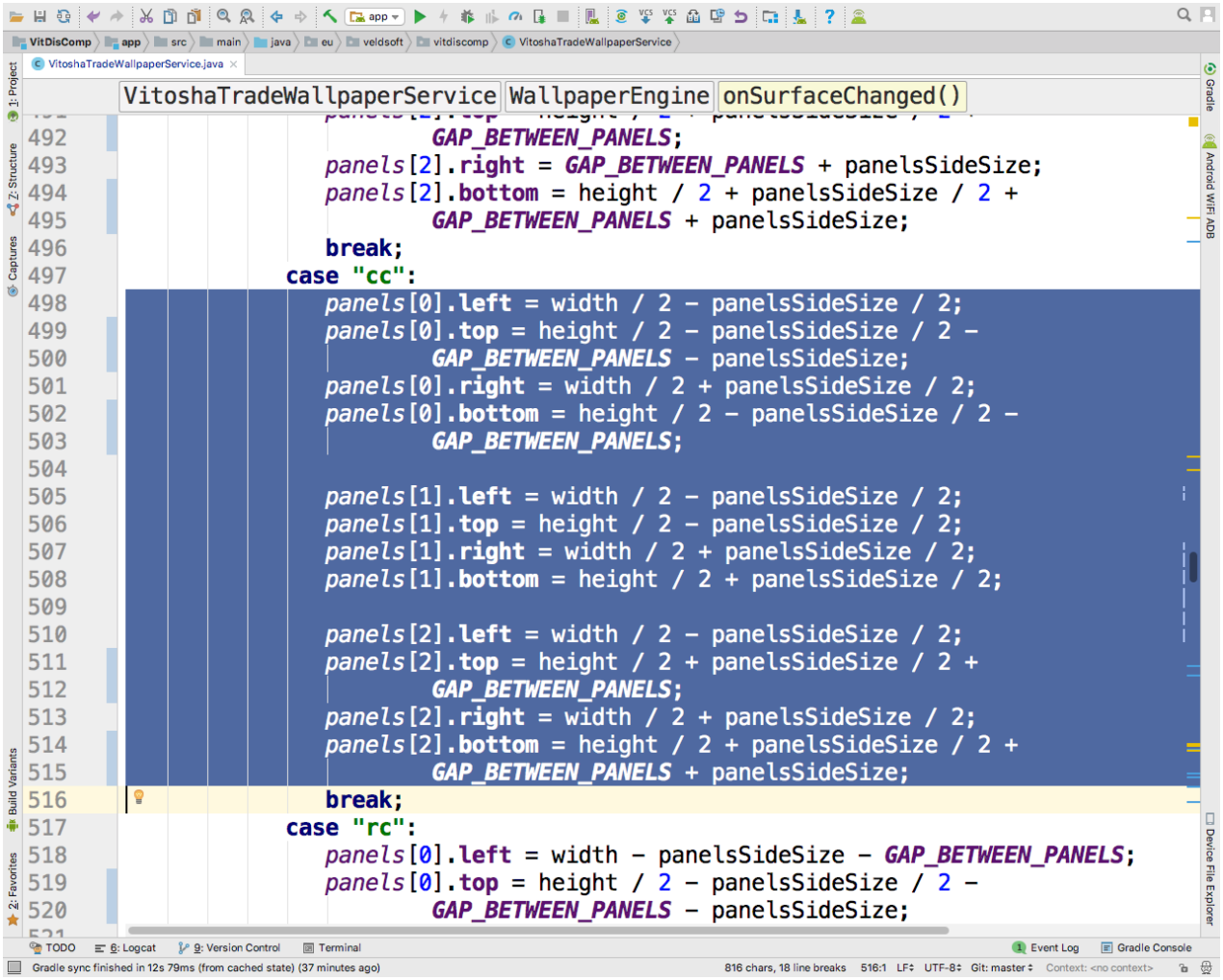
Фигура 4.60: Координати и размери на областите за визуално представяне - горе-център



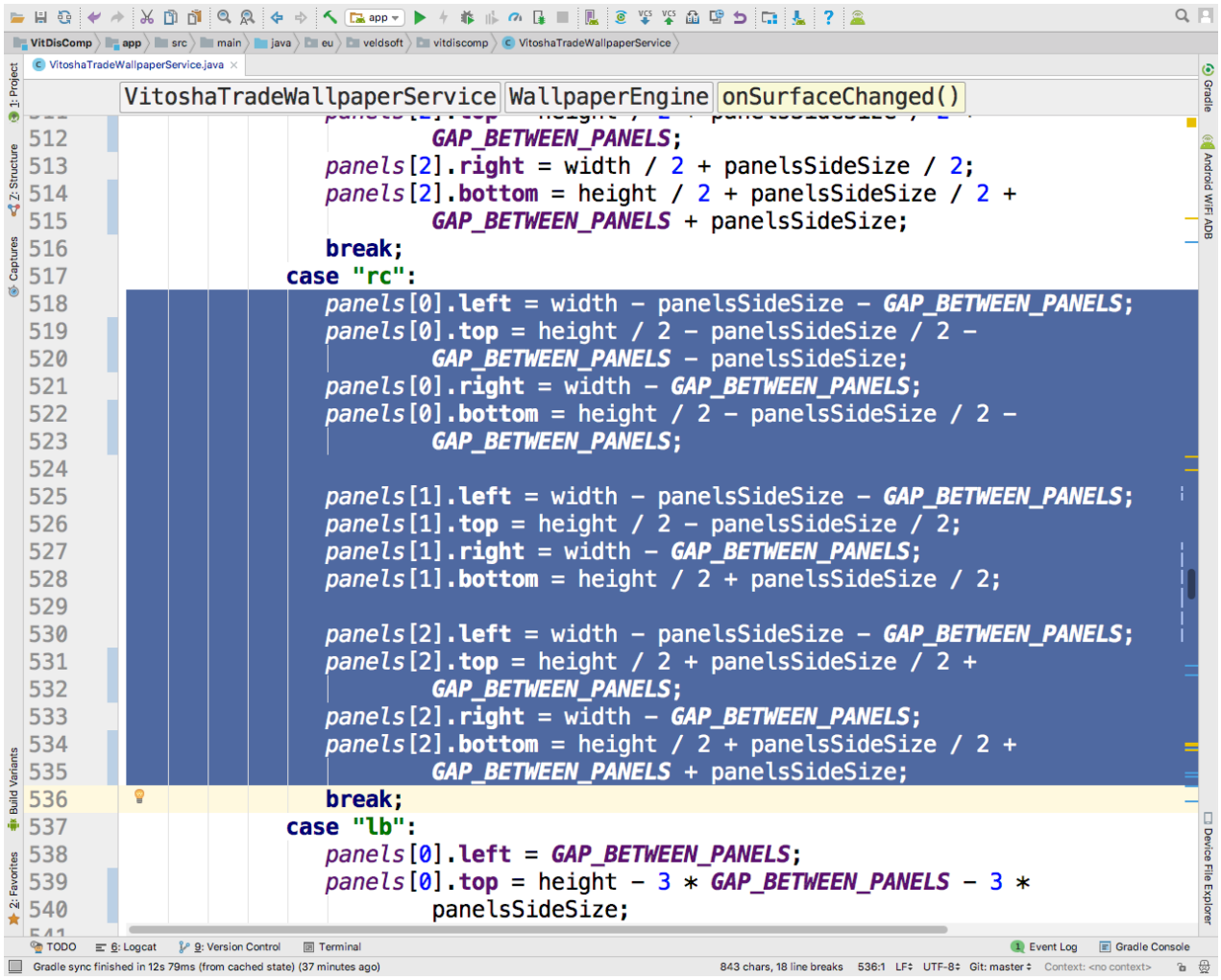
Фигура 4.61: Координати и размери на областите за визуално представяне - горе-дясно



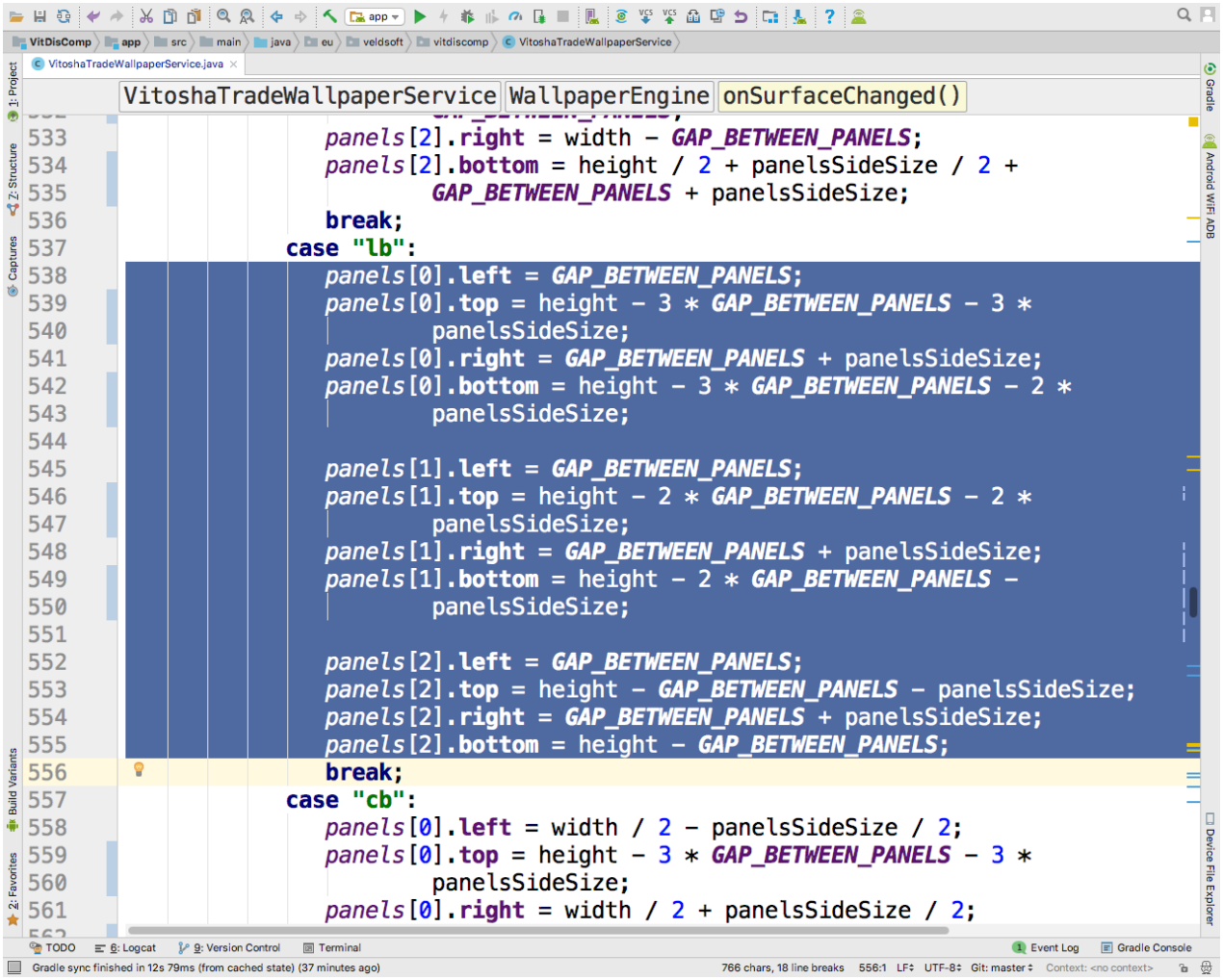
Фигура 4.62: Координати и размери на областите за визуално представяне - център-ляво



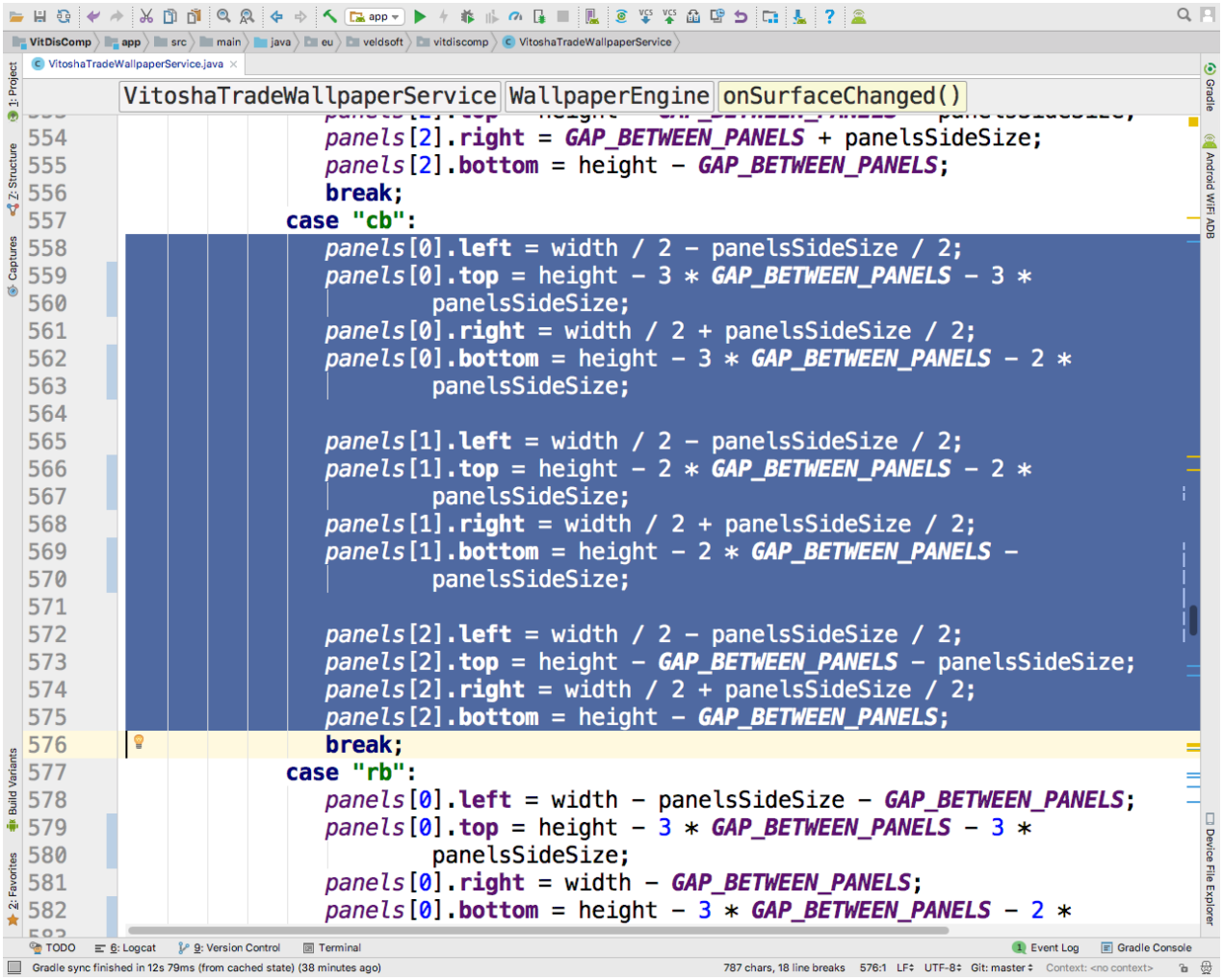
Фигура 4.63: Координати и размери на областите за визуално представяне - център-център



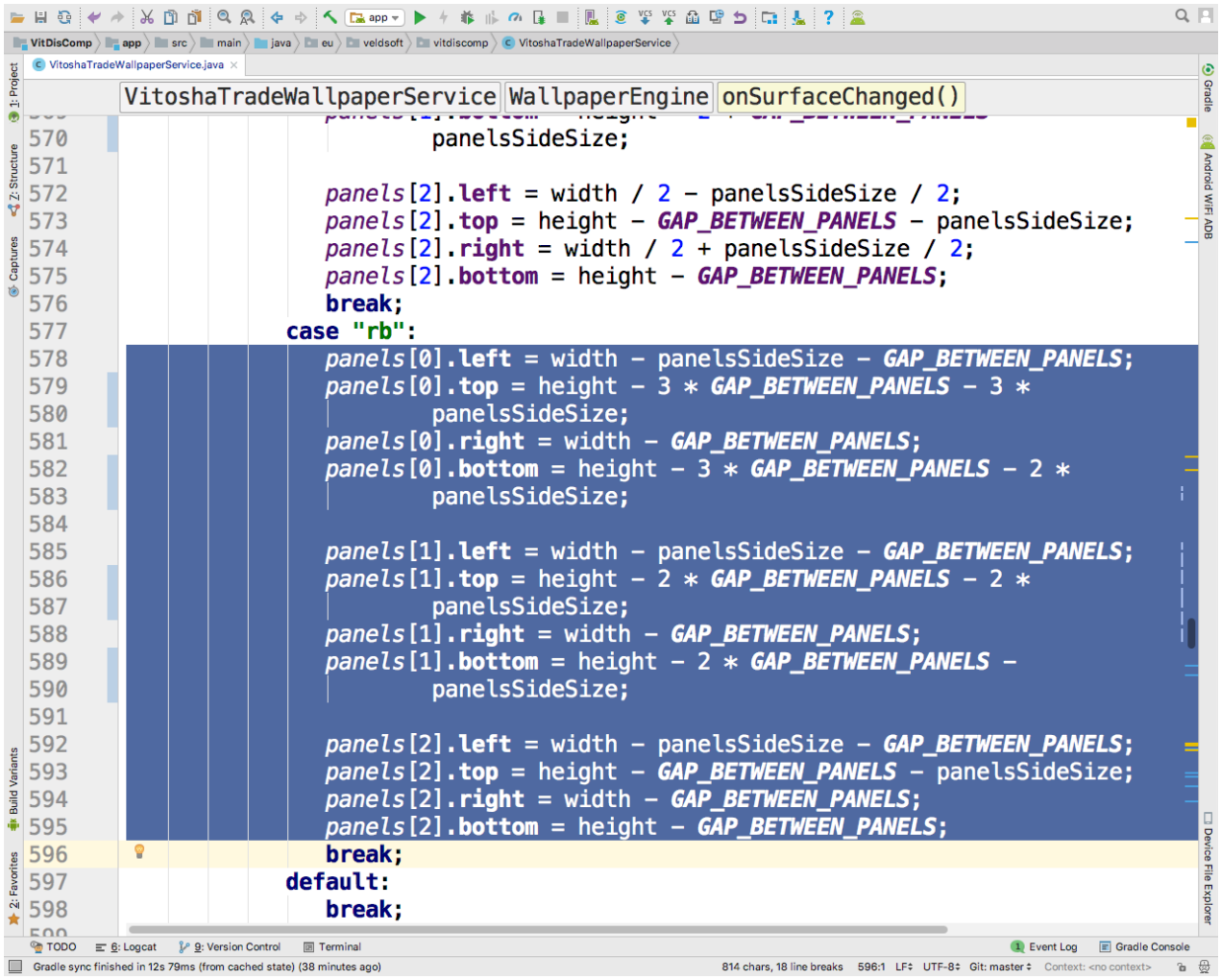
Фигура 4.64: Координати и размери на областите за визуално представяне - център-дясно



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

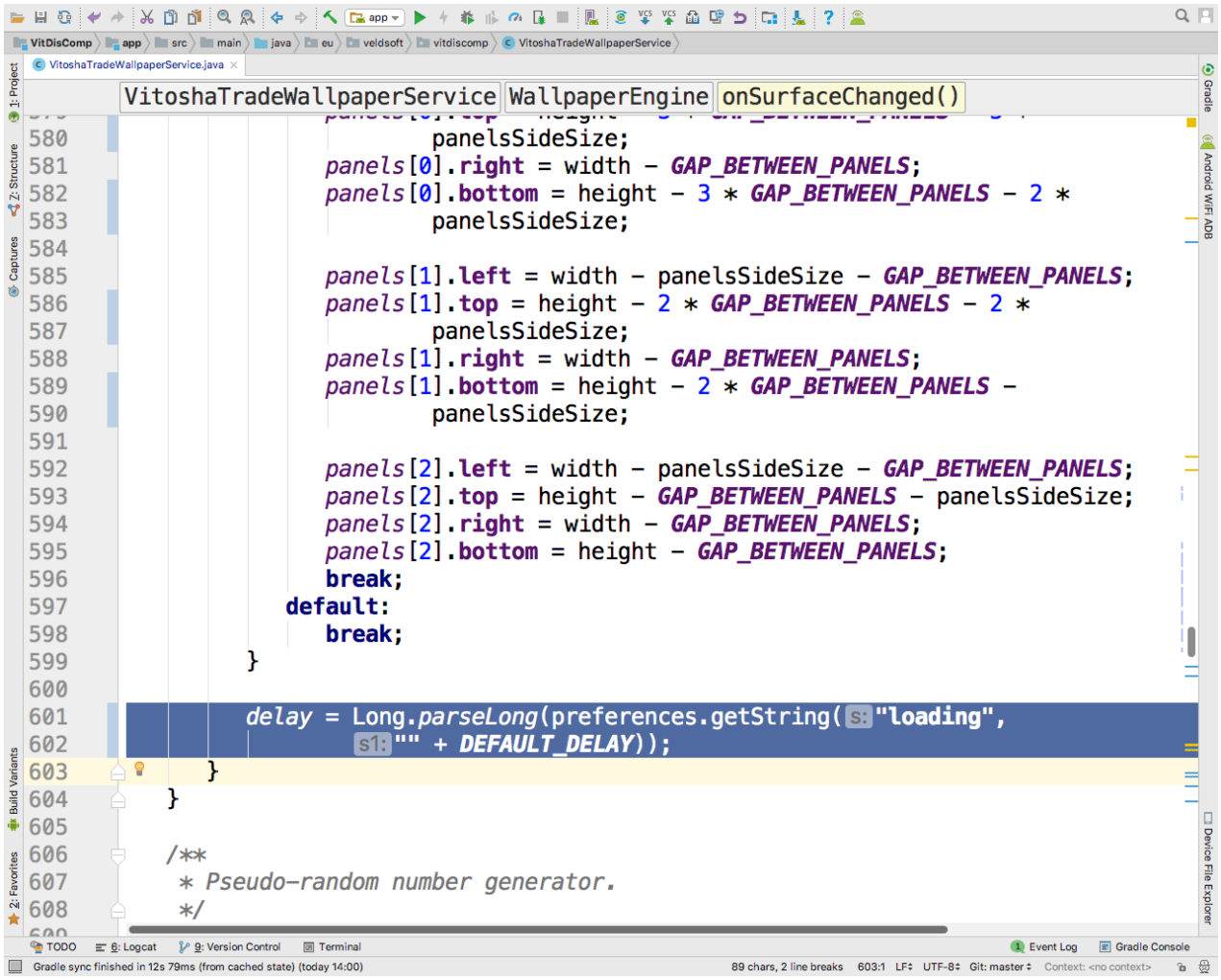


Фигура 4.66: Координати и размери на областите за визуално представяне - долу-център



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.

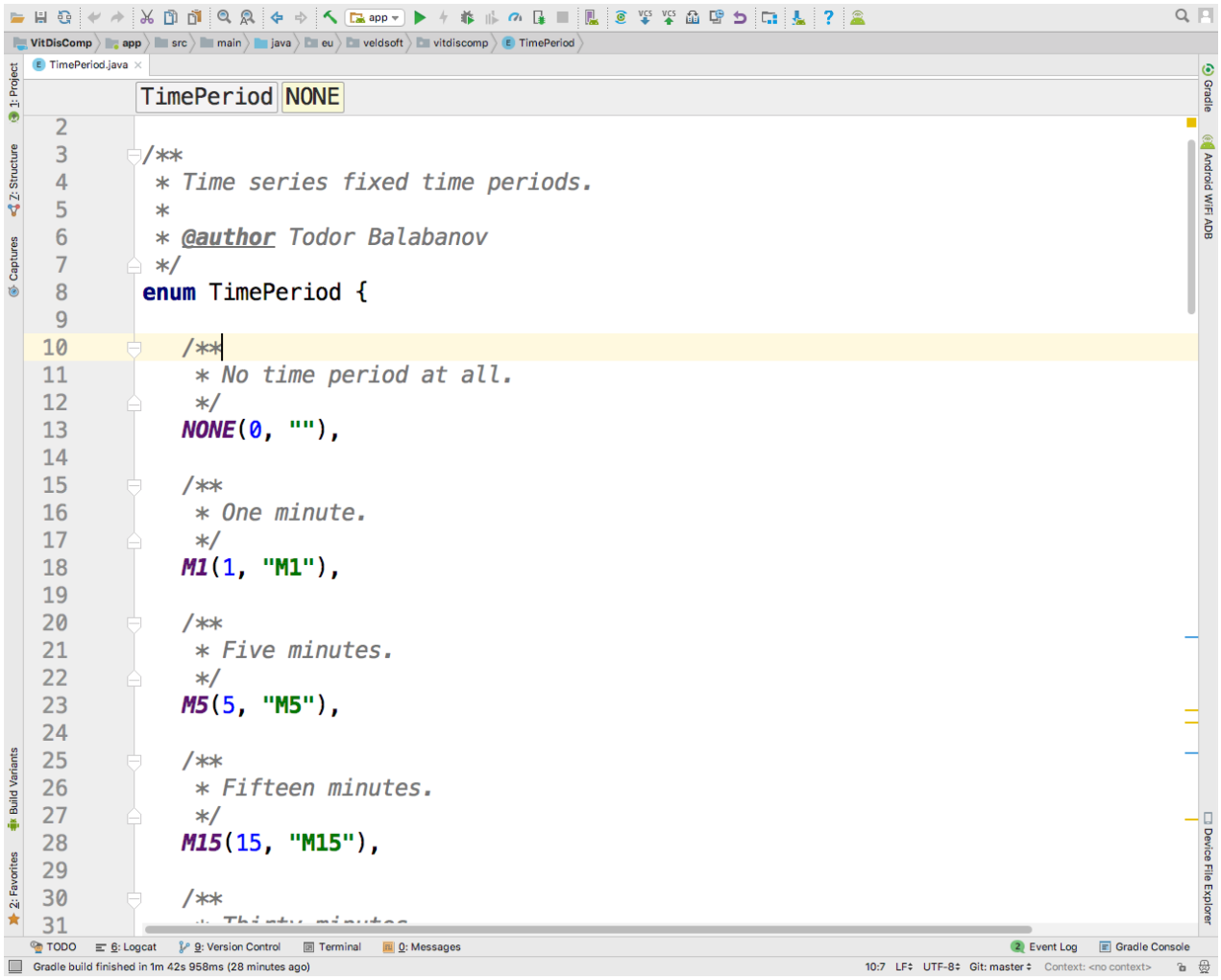


Фигура 4.68: Натоварване на системата за фонови пресмятания

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

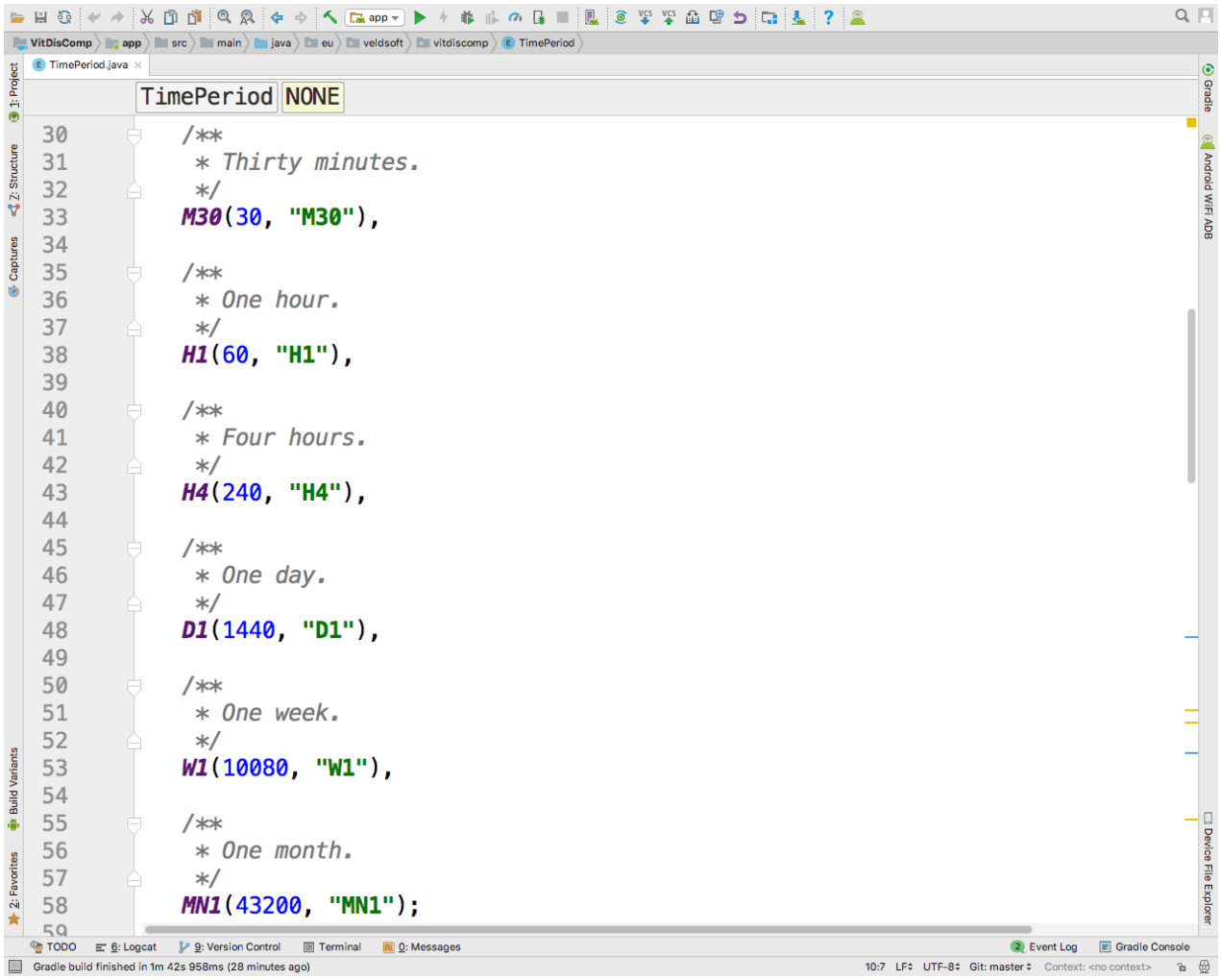
4.5 Представяне на информацията върху локалното устройство

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



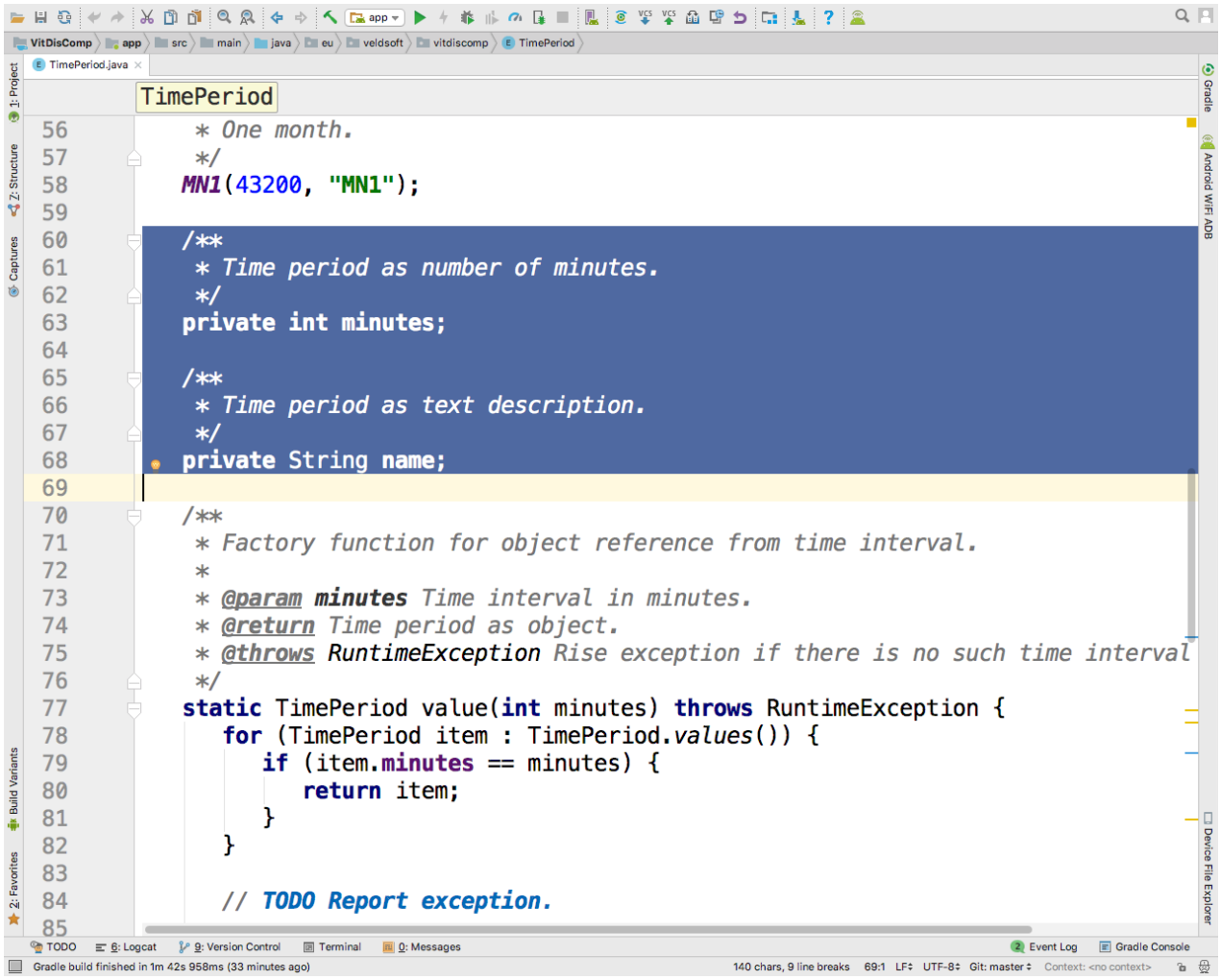
Фигура 4.69: Константи за интервалите на времевия ред (0 до 15)

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



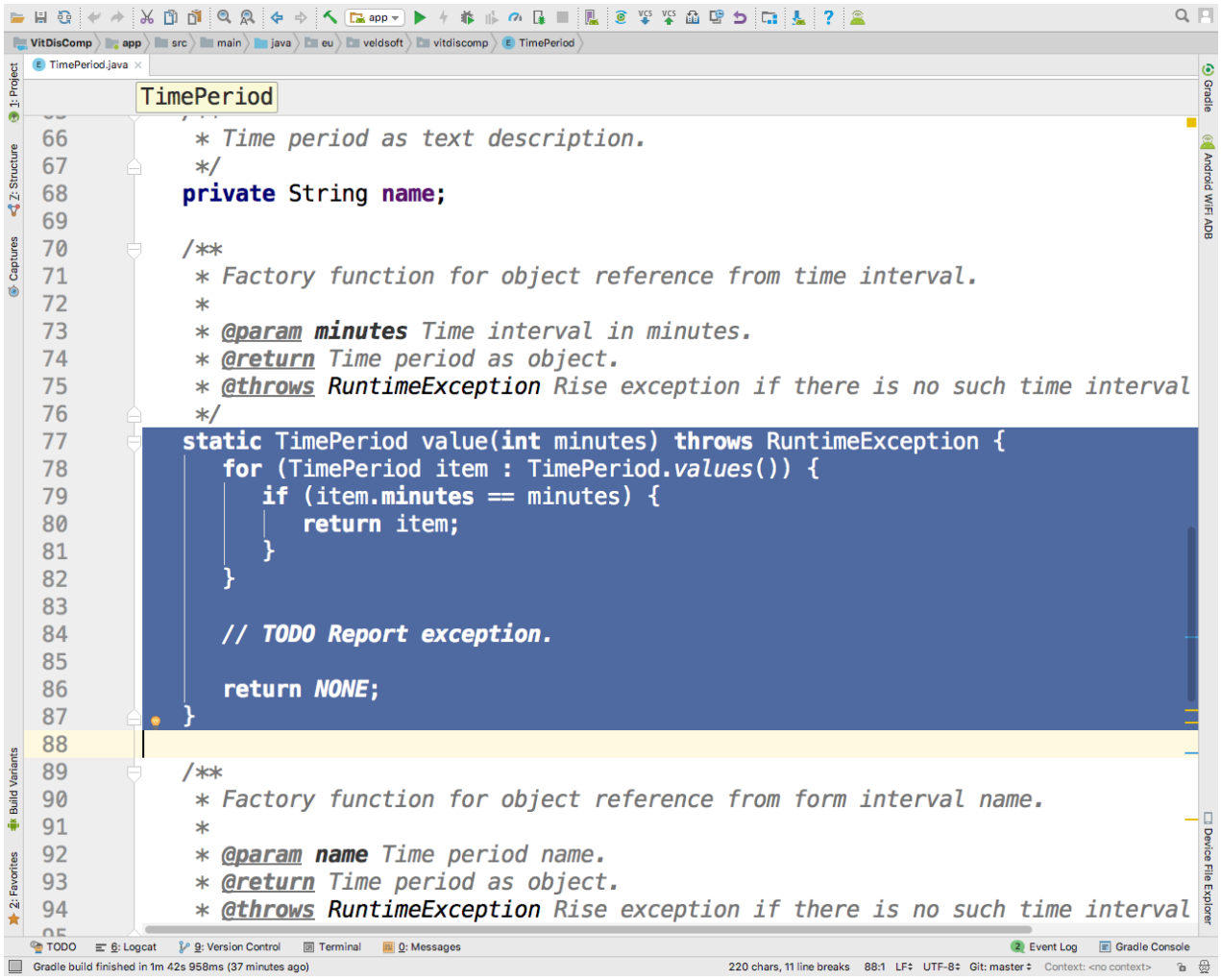
Фигура 4.70: Константи за интервалите на времевия ред (30 до 43200)

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).



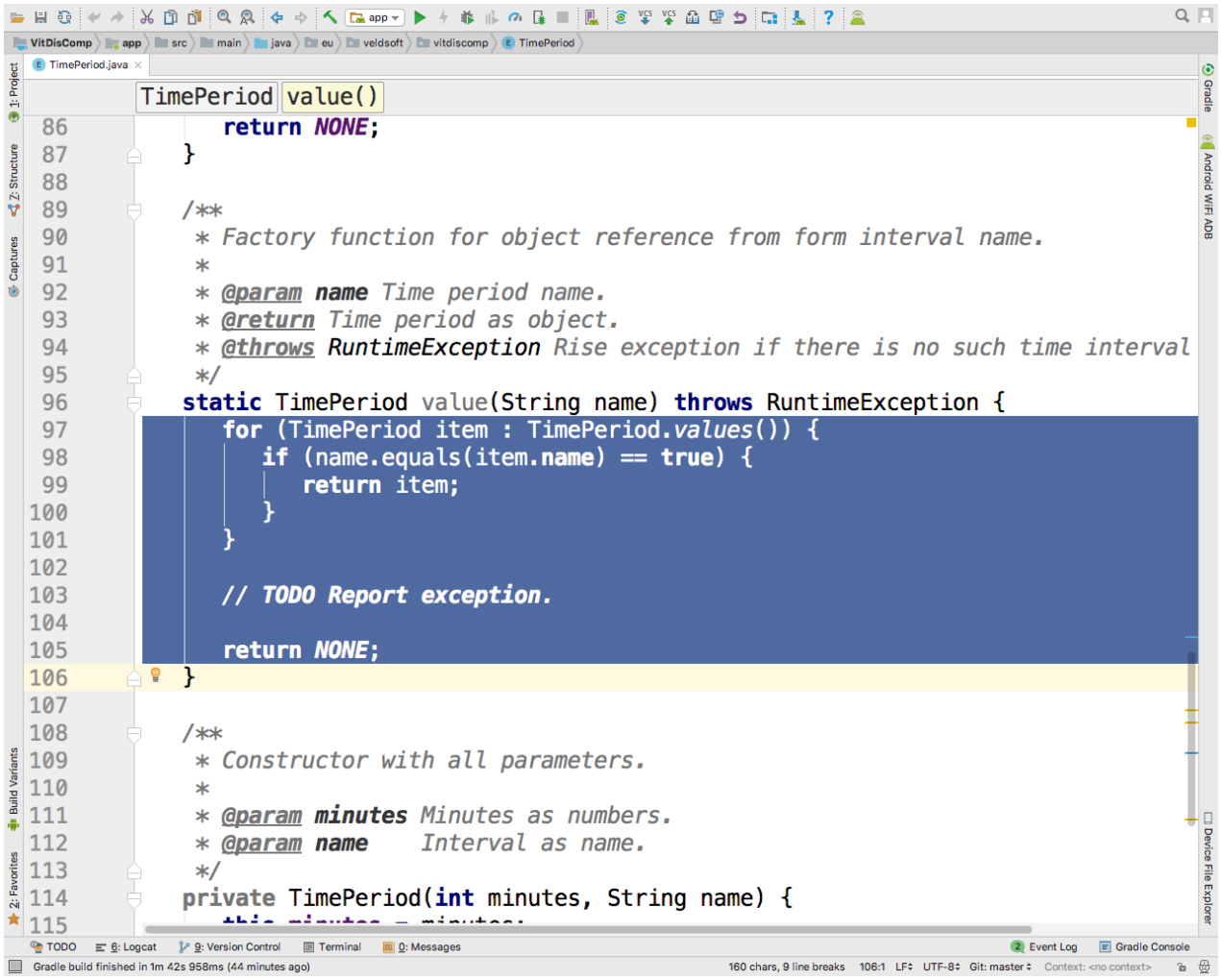
Фигура 4.71: Описание на времеви интервал

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).



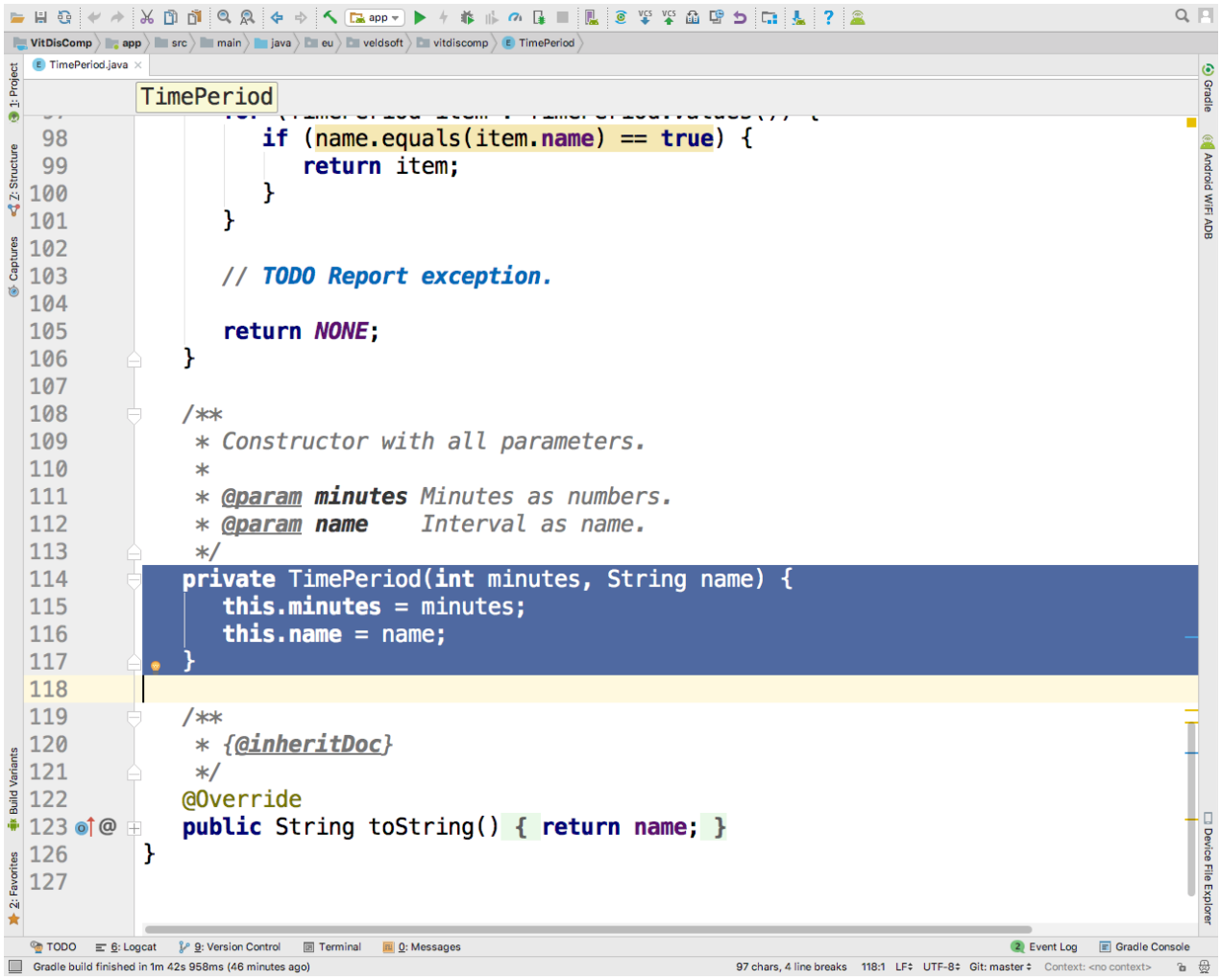
Фигура 4.72: Определяне на константа за интервал по брой минути

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



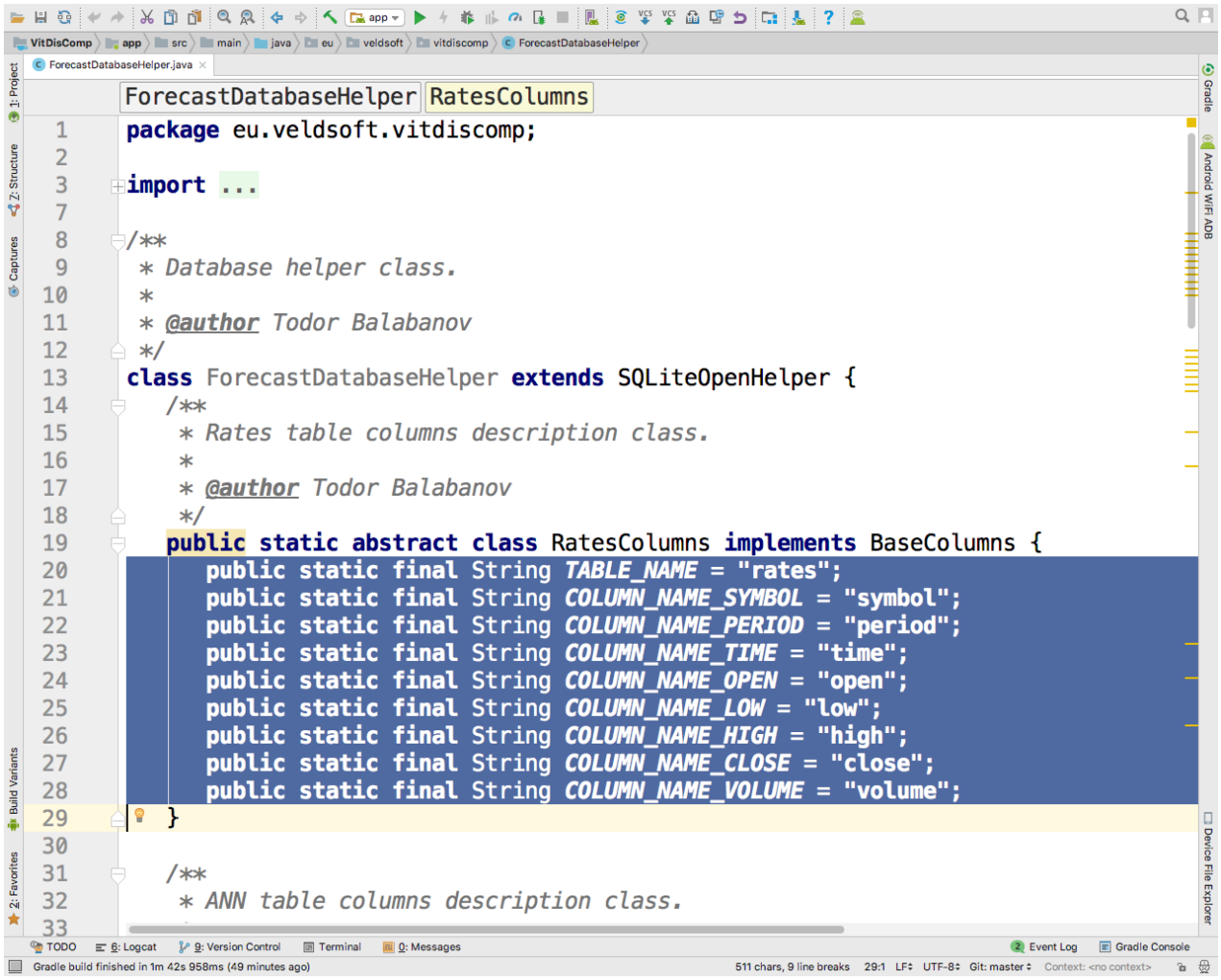
Фигура 4.73: Определяне на константа за интервал по текстово описание

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



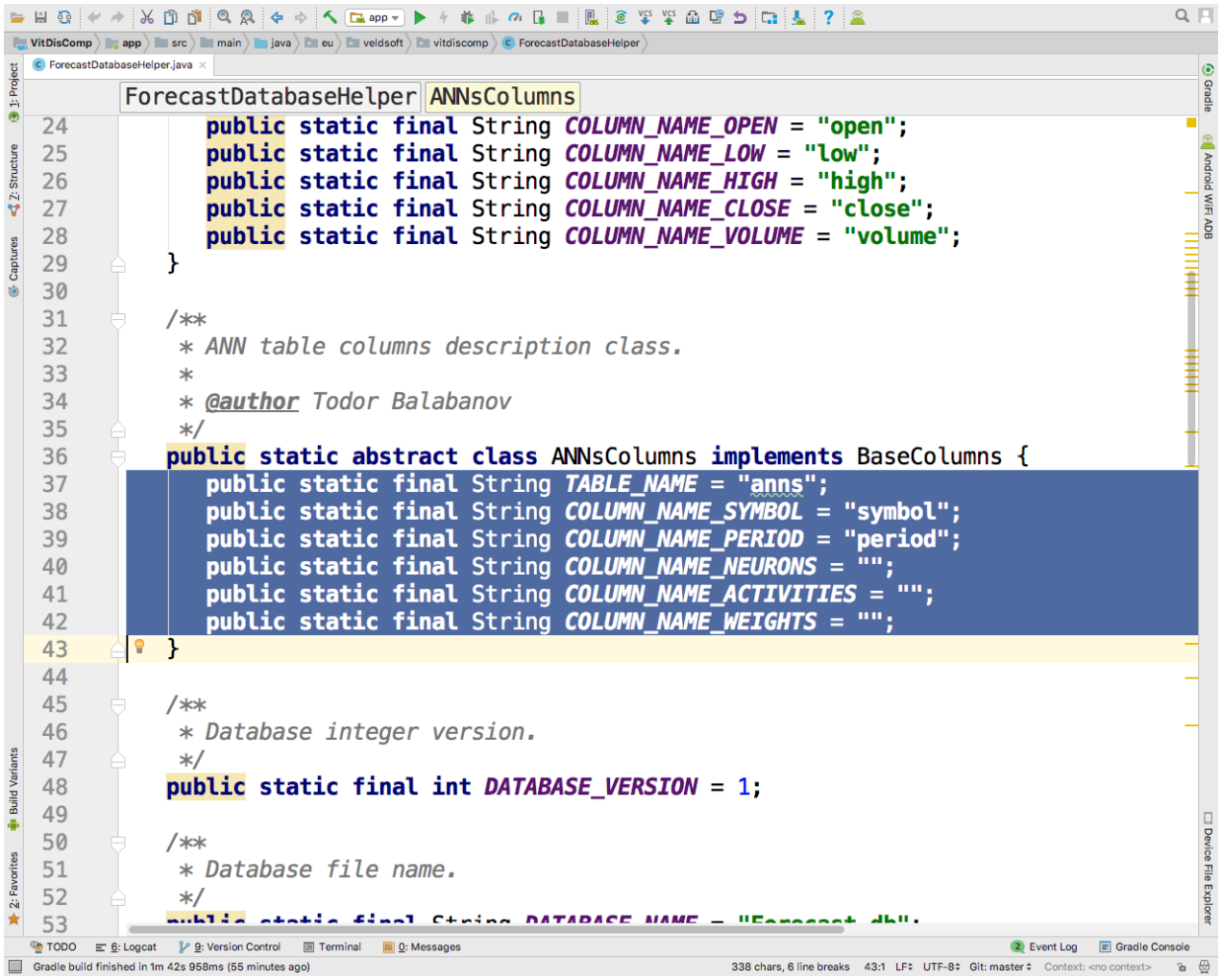
Фигура 4.74: Конструктор на изброения тип за интервал

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



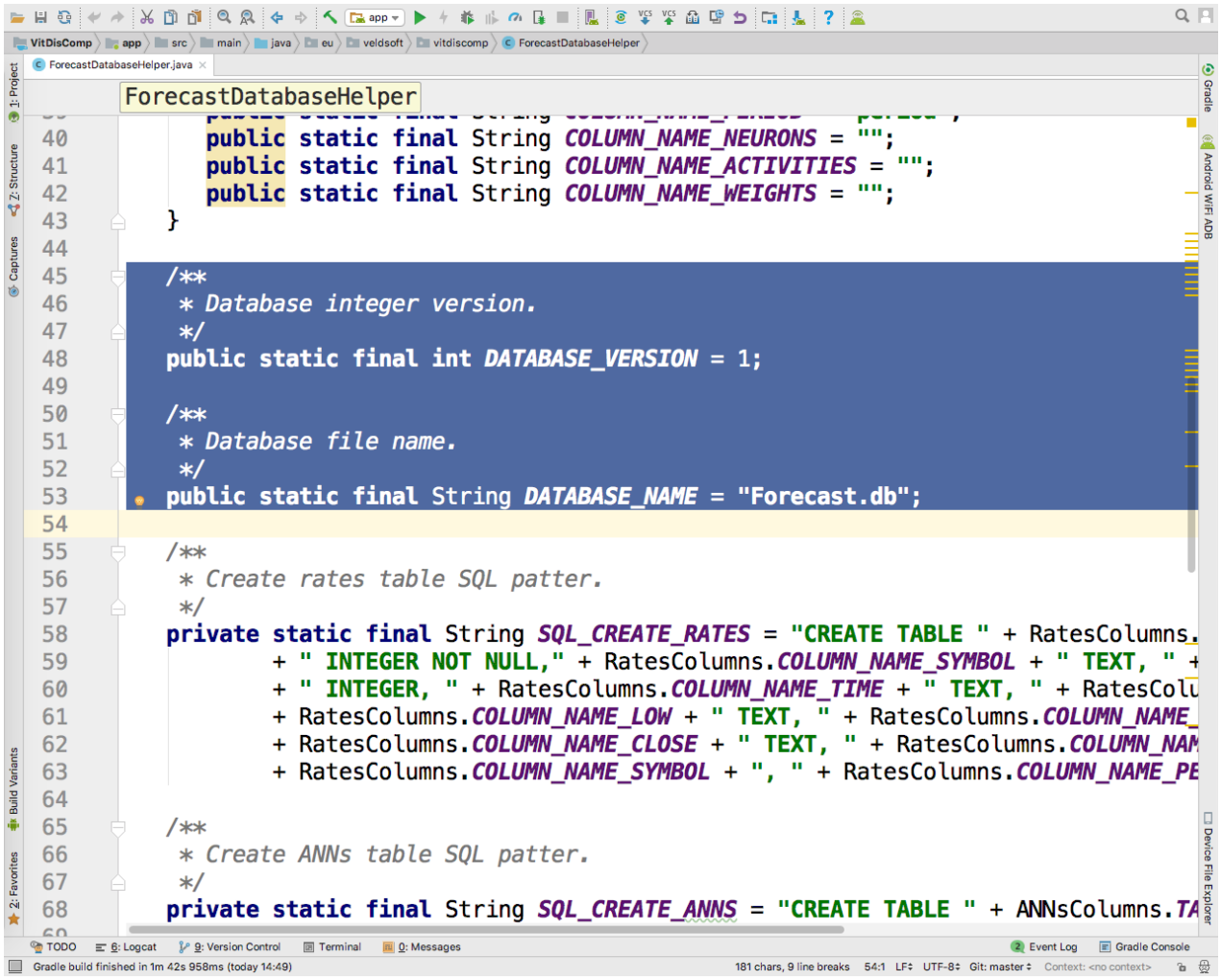
Фигура 4.75: Структура на локалната таблица за котировките

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).



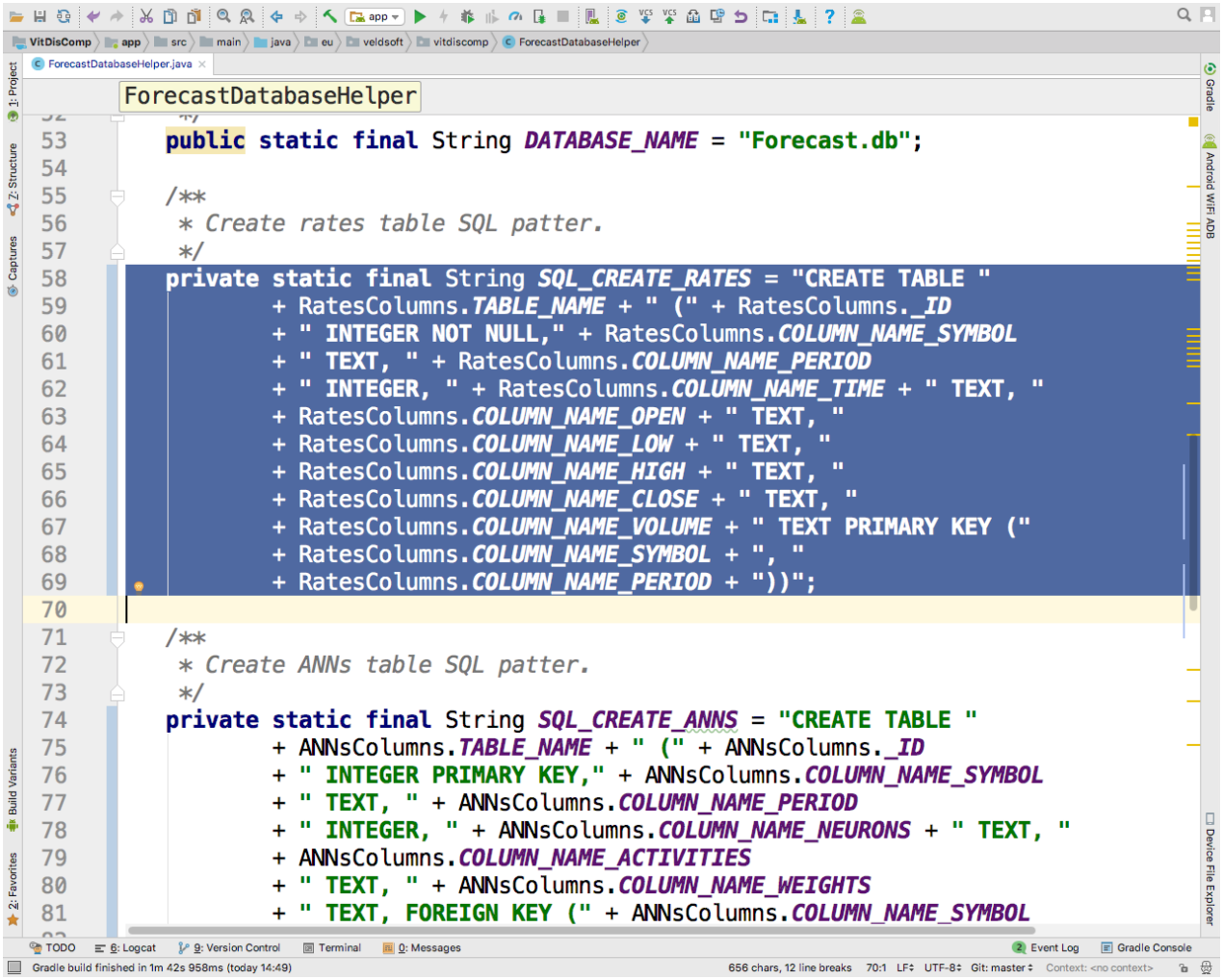
Фигура 4.76: Структура на локалната таблица за изкуствените невронни мрежи

Допълнителна таблица поема съхранението на информацията за изкуствените неврони 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).

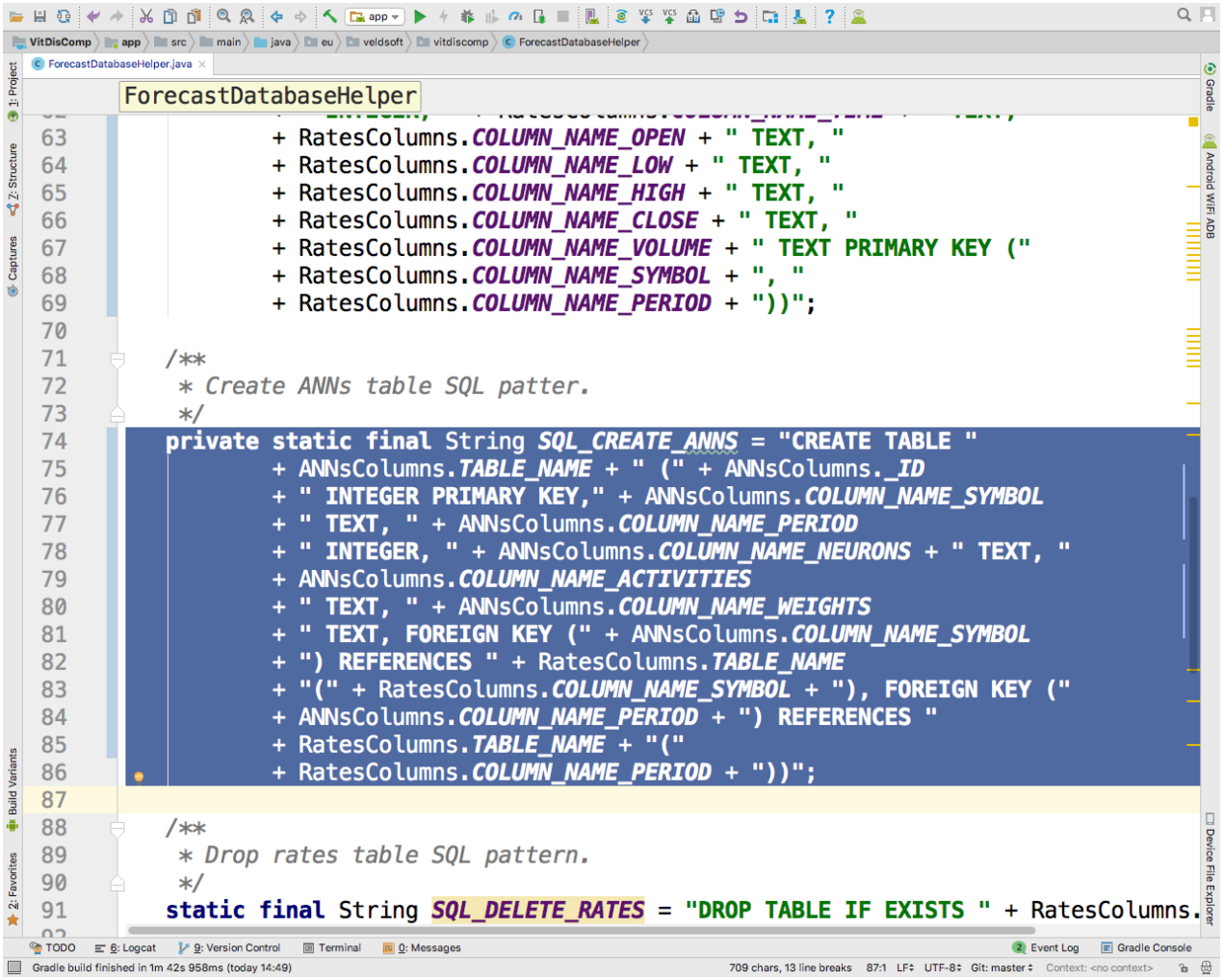


Фигура 4.77: Име и версията на базата данни

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

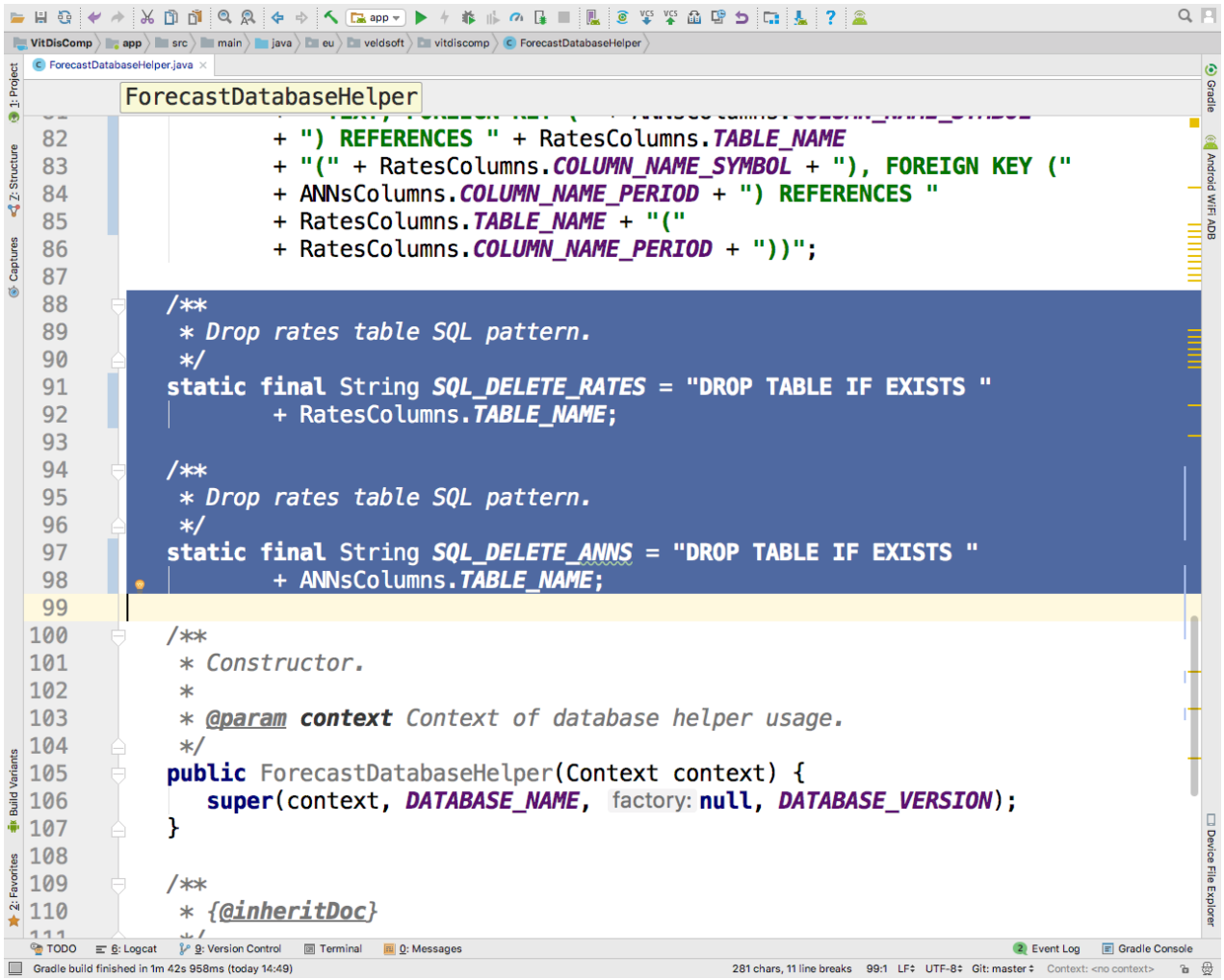


Фигура 4.78: Код за създаване на таблицата за котировки



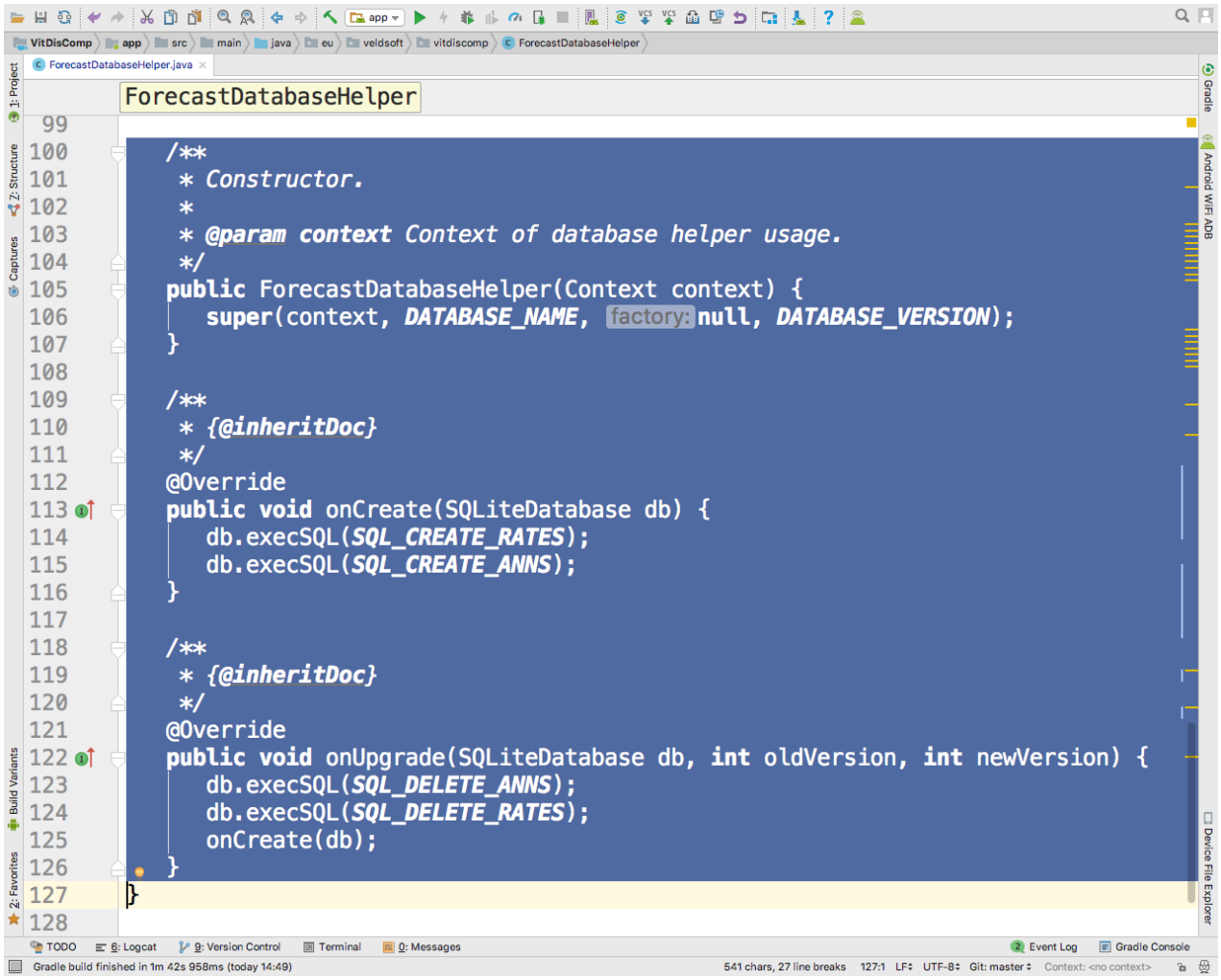
Фигура 4.79: Код за създаване на таблицата за изкуствените невронни мрежи

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



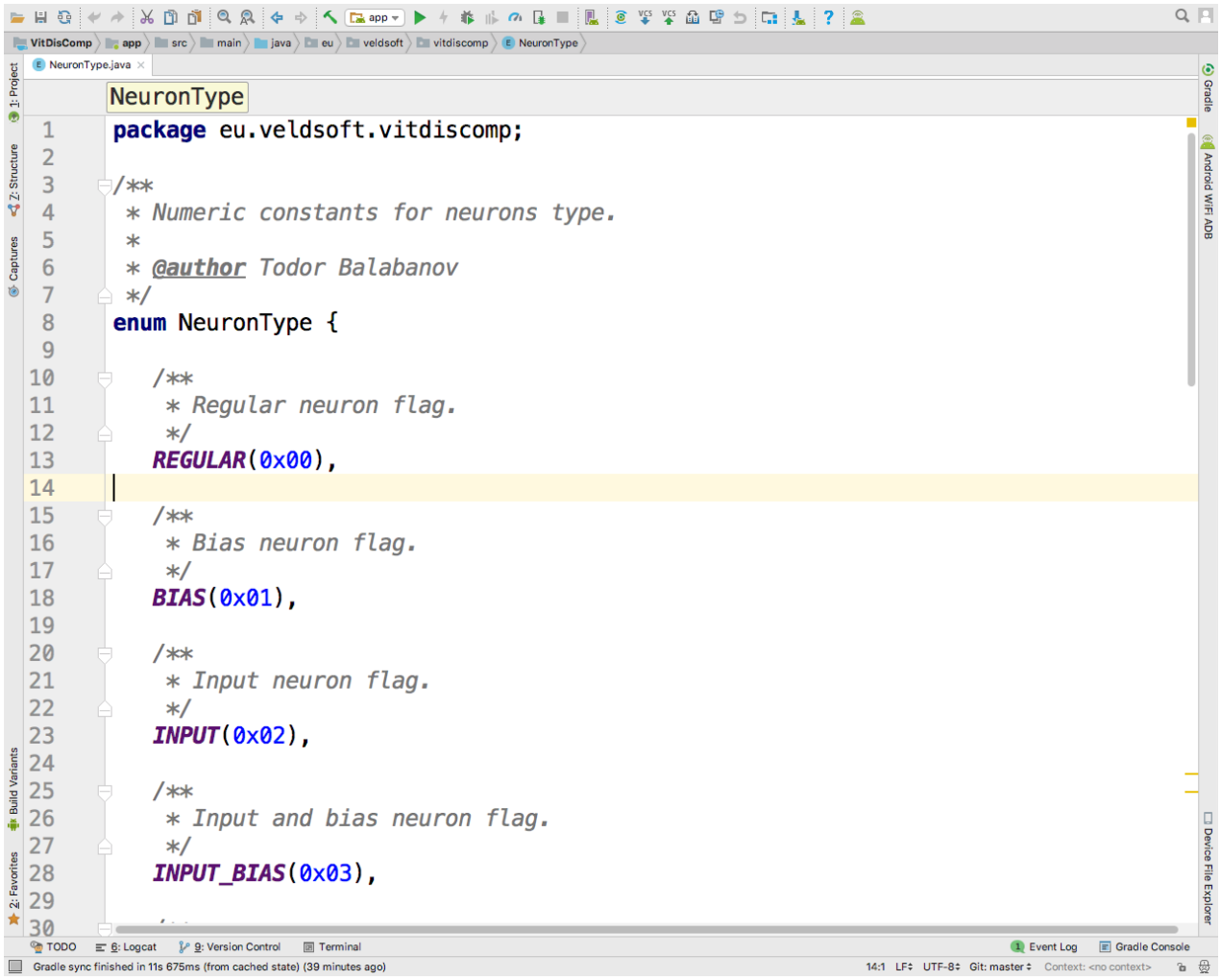
Фигура 4.80: Код за изтриване на таблиците

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



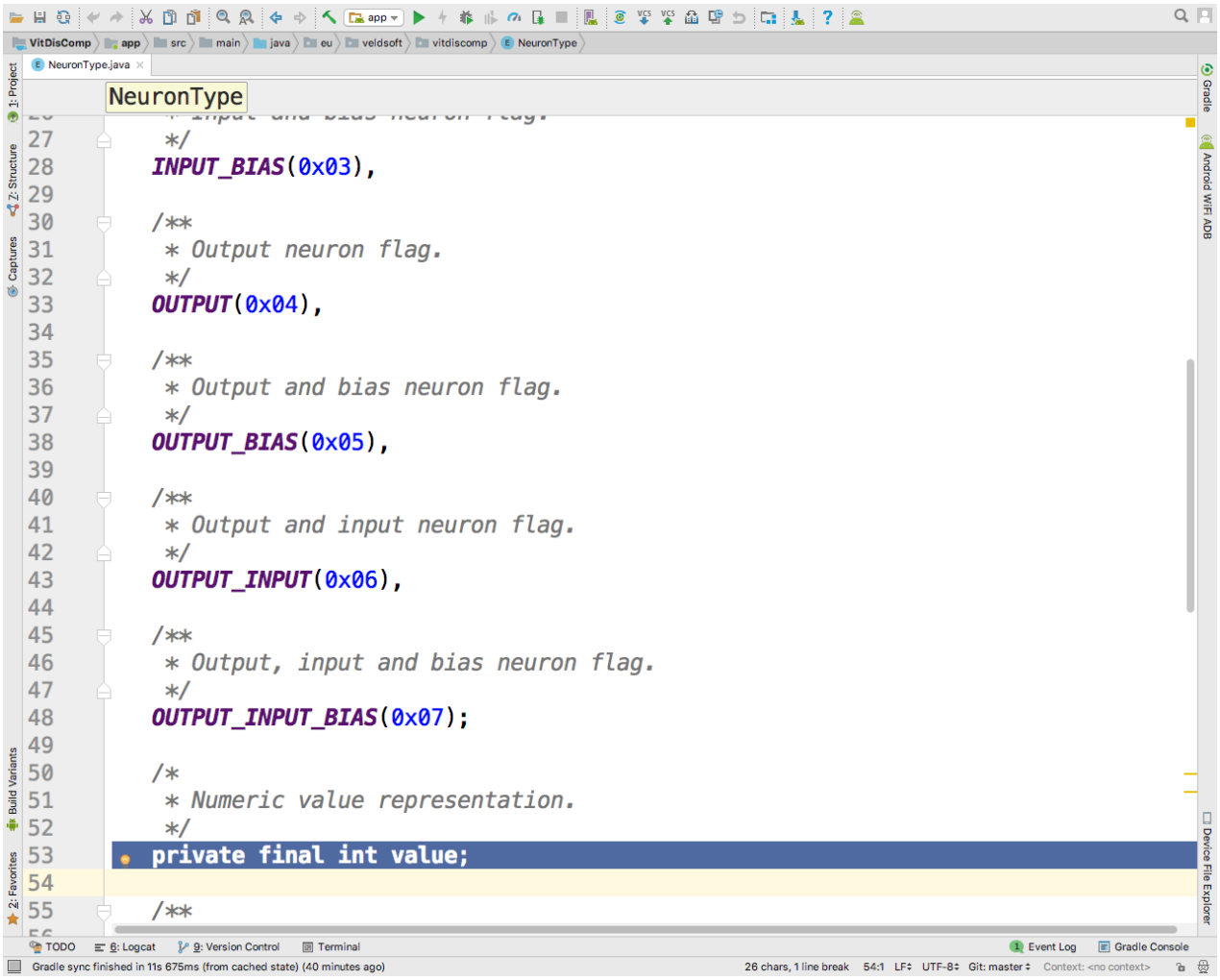
Фигура 4.81: Конструктор и виртуални методи на спомагателния клас за базата данни

Конструкторът на спомагателния клас за базата данни има единствена задача да извика конструктора 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).



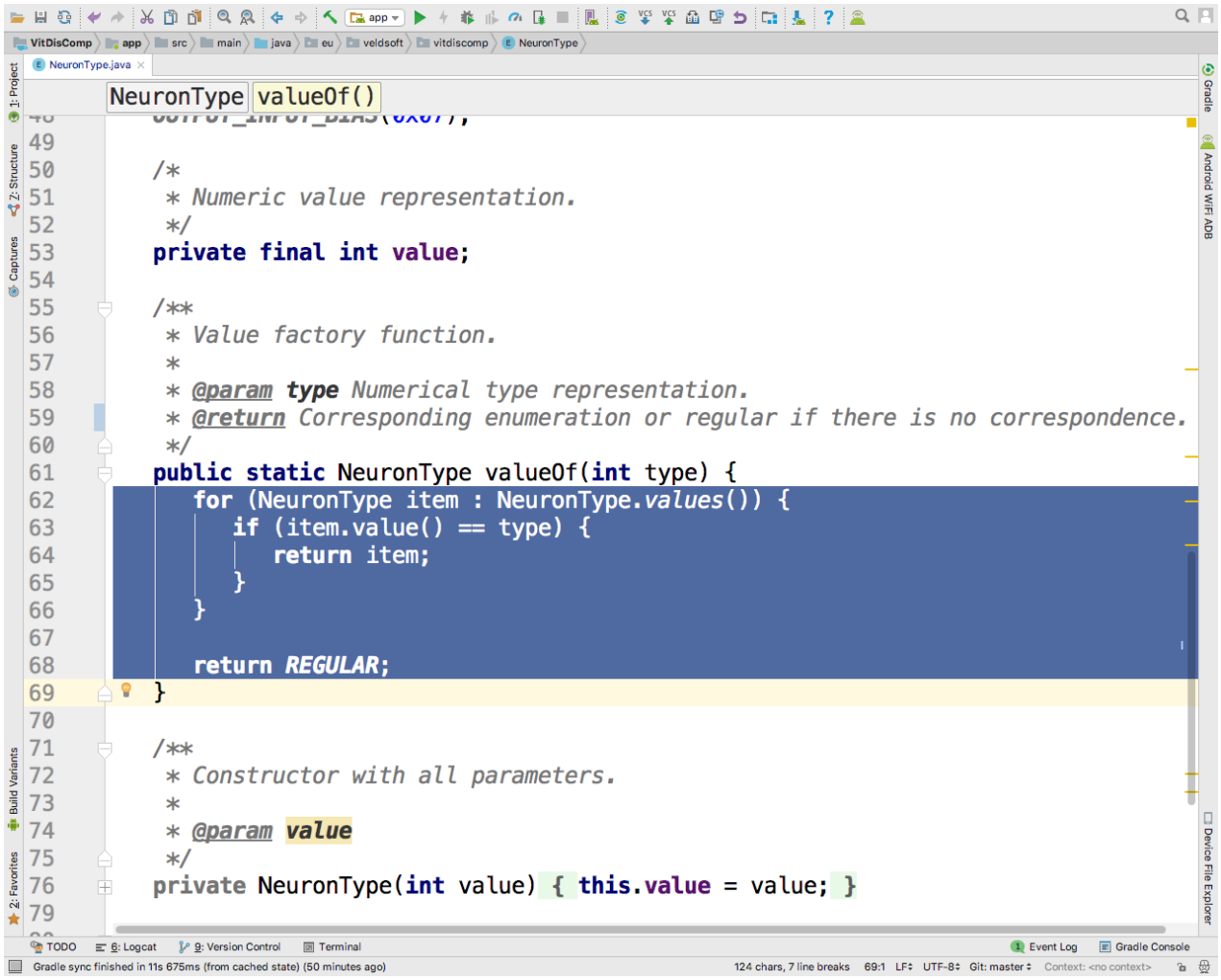
Фигура 4.82: Изброен тип за вида на невроните

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).



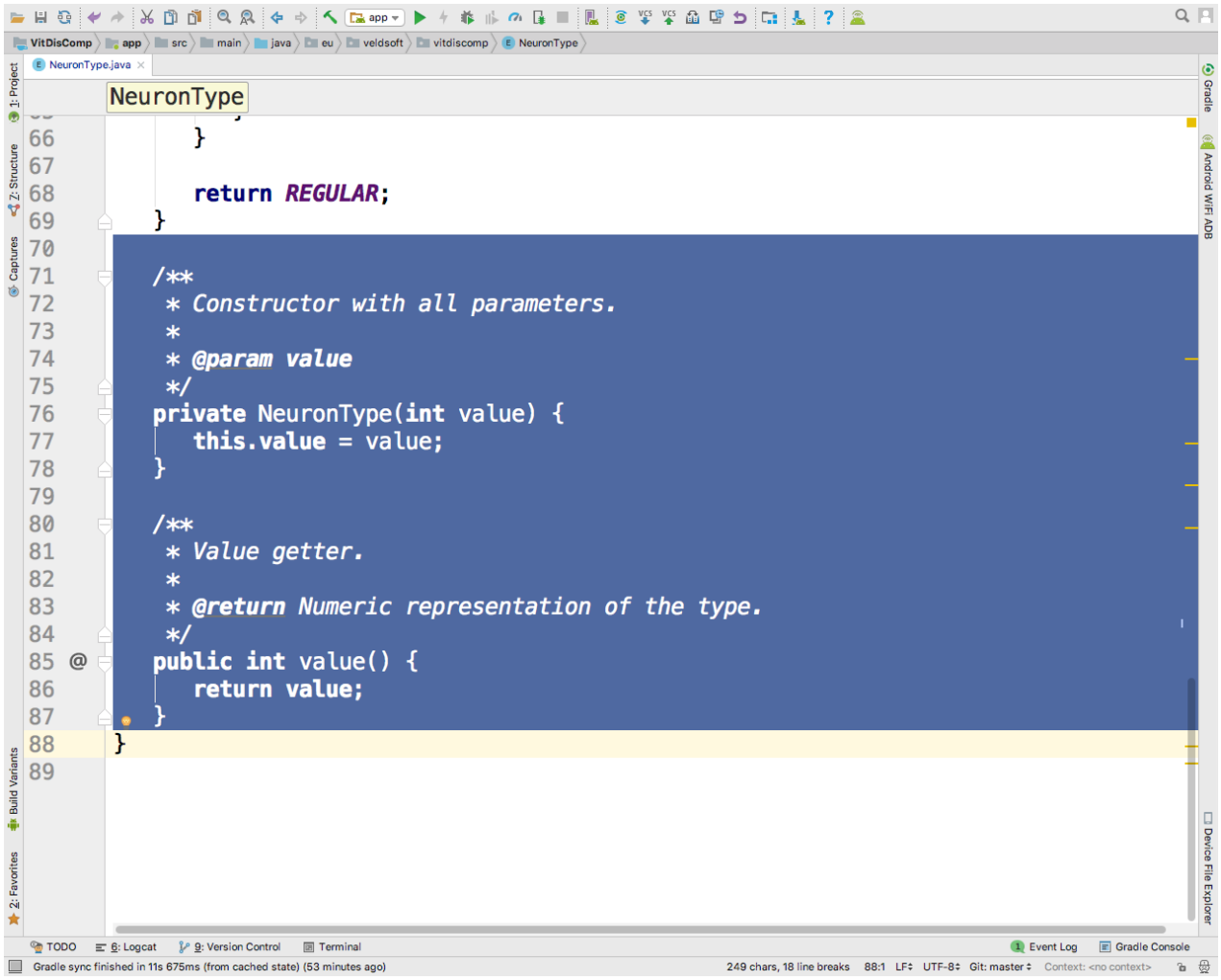
Фигура 4.83: Целочислена константа за типа на неврона, която може да се ползва и като битово поле

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).



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 статичена функция за конструиране на обекта (Factory Method Design Pattern), която по числена стойност да определя константата в изброения тип (Фиг. 4.84).



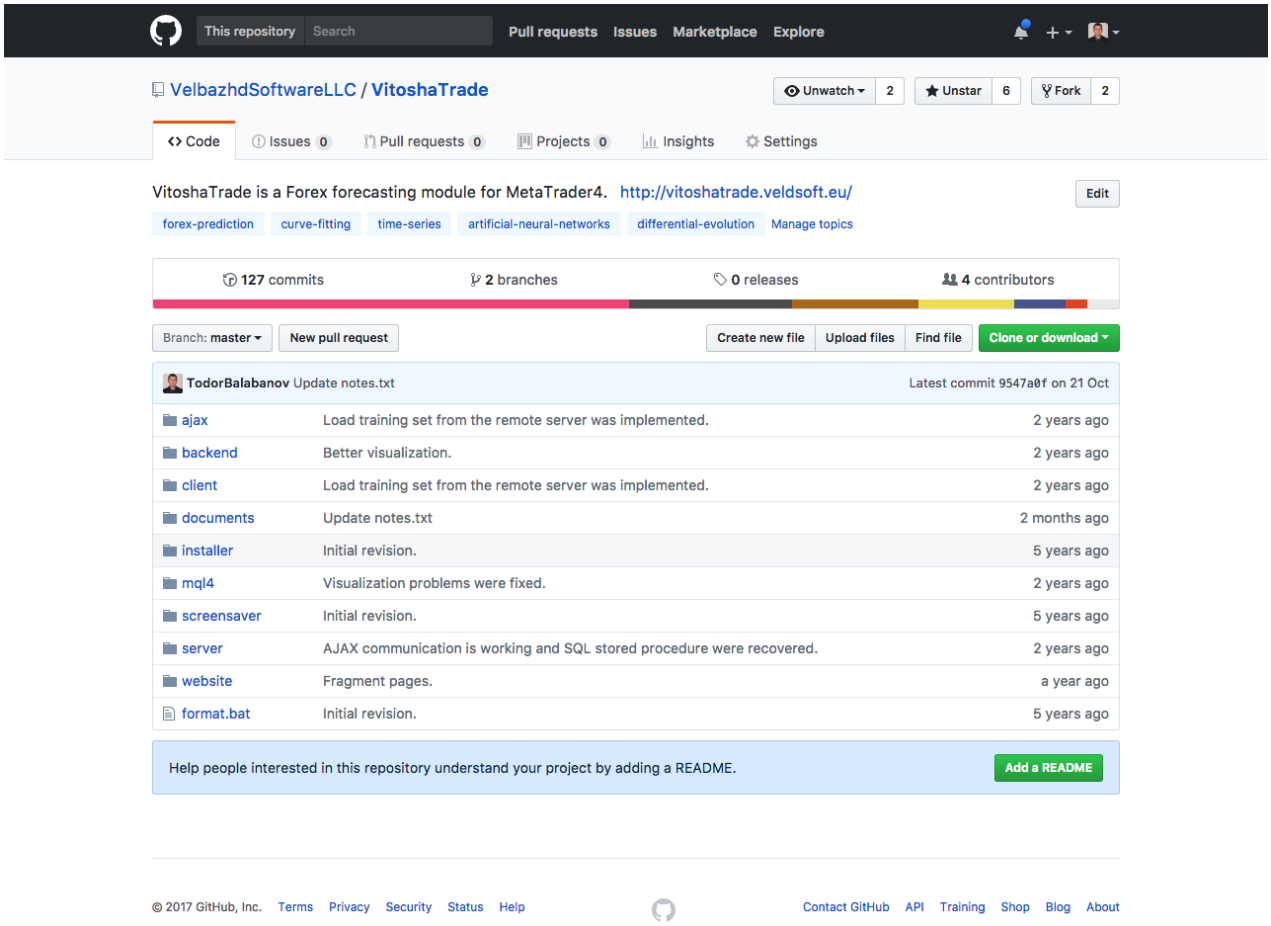
Фигура 4.85: Методи за достъп до вътрешните стойности на константите

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

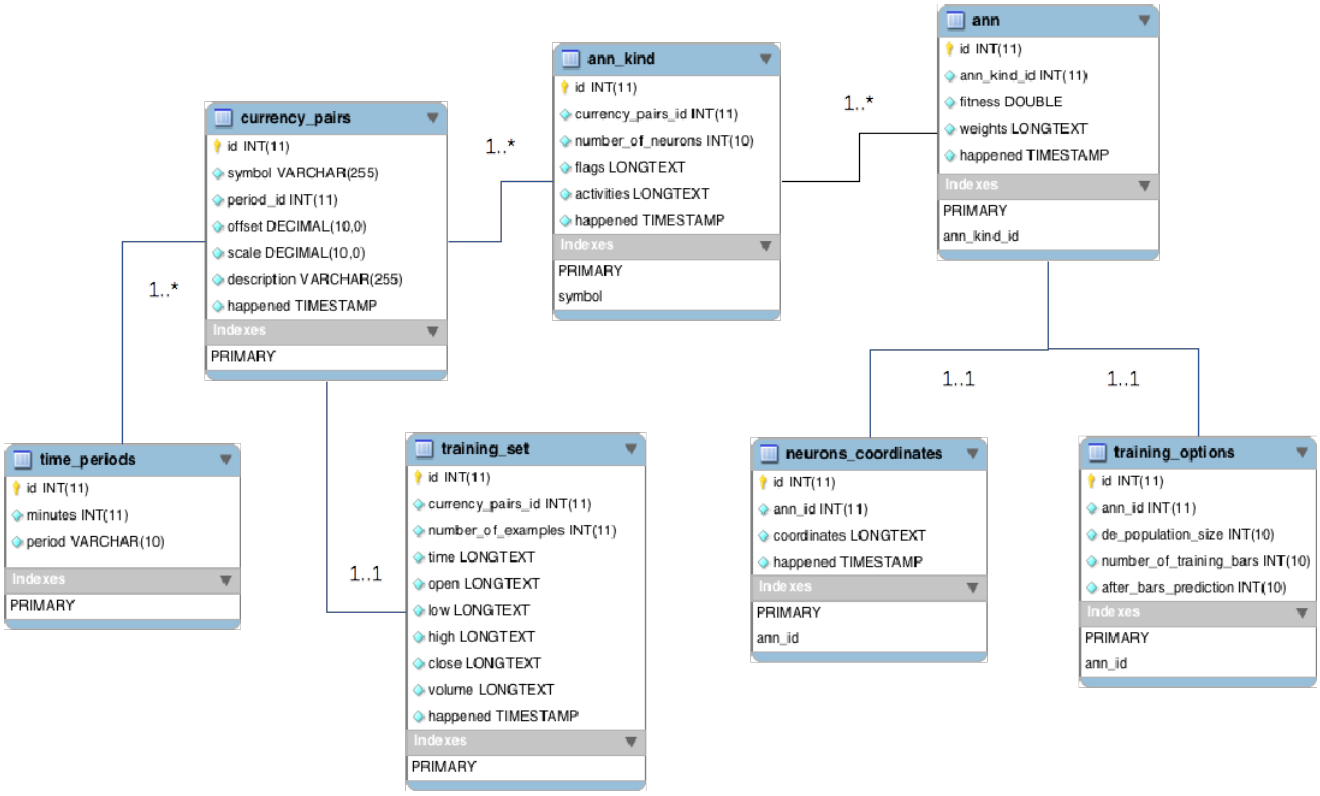
За да бъдат успешно извършени изчисленията в разпределена среда, освен наличието на множество устройства е от съществено значение да има централизирано място, от което да бъдат раздавани задачите и където да бъдат събирани резултатите. Архитектура от тип клиент-сървър би била изключително удачна при 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

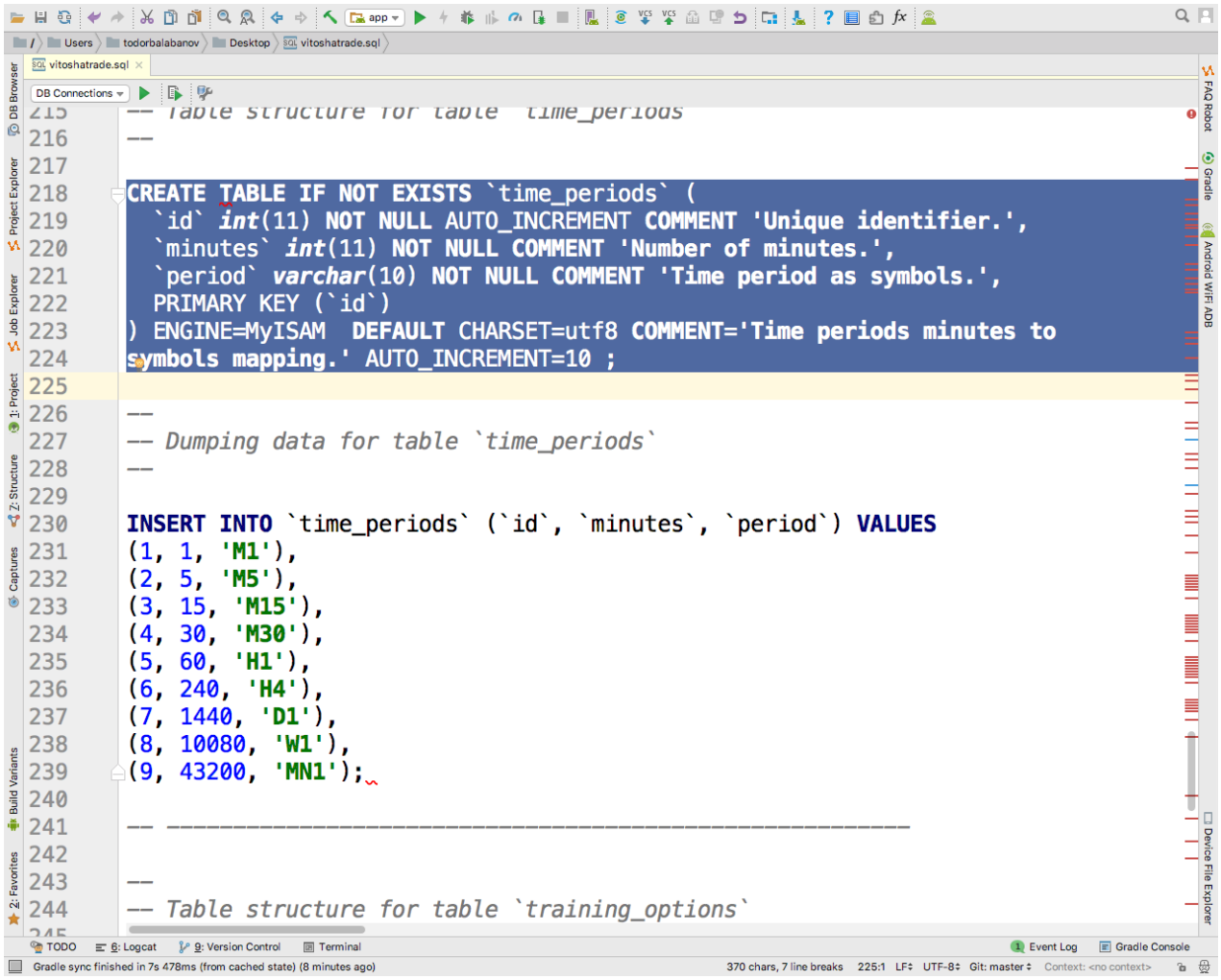
5.1 Релационна база данни

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).



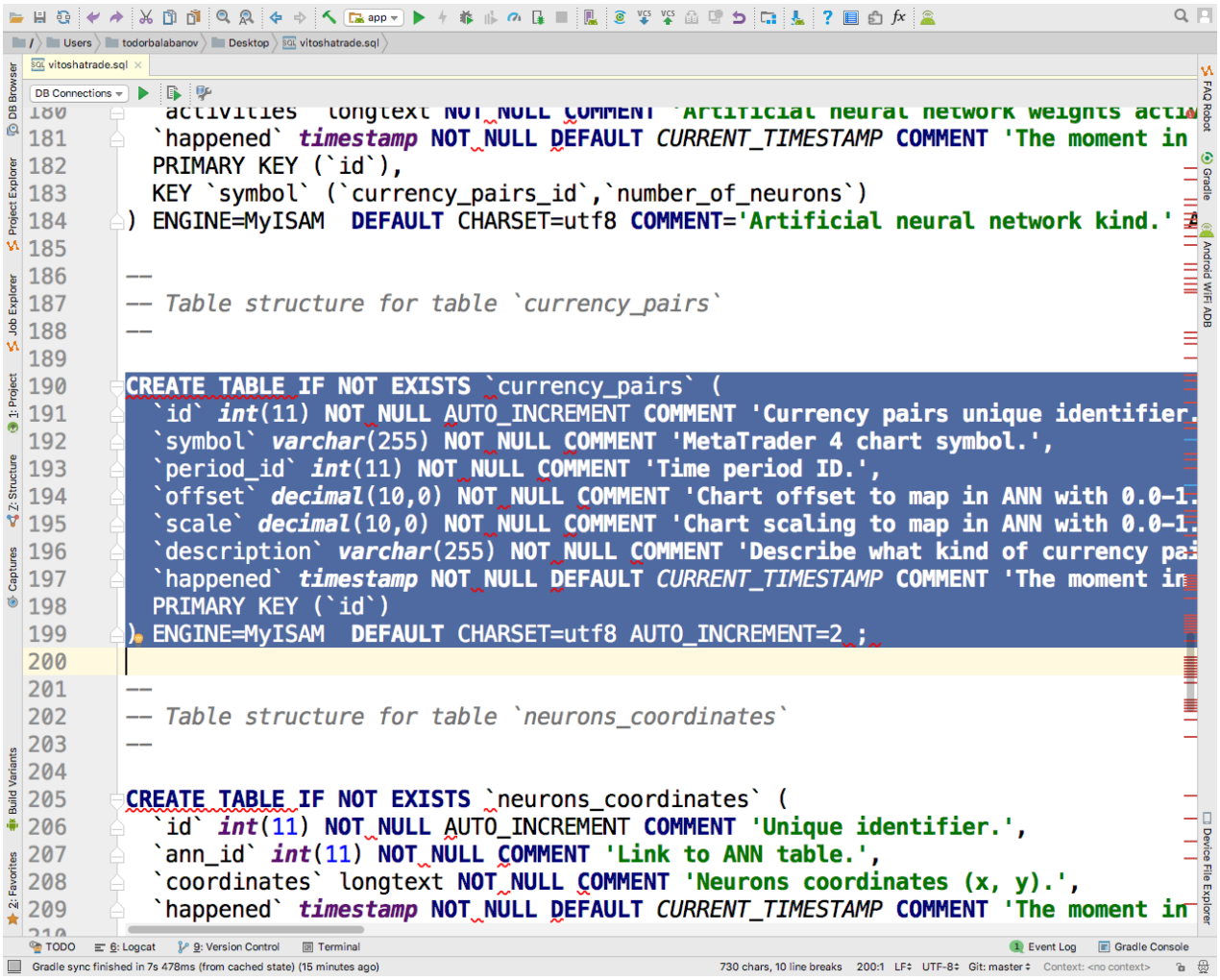
Фигура 5.2: Физическа структура на базата данни

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).



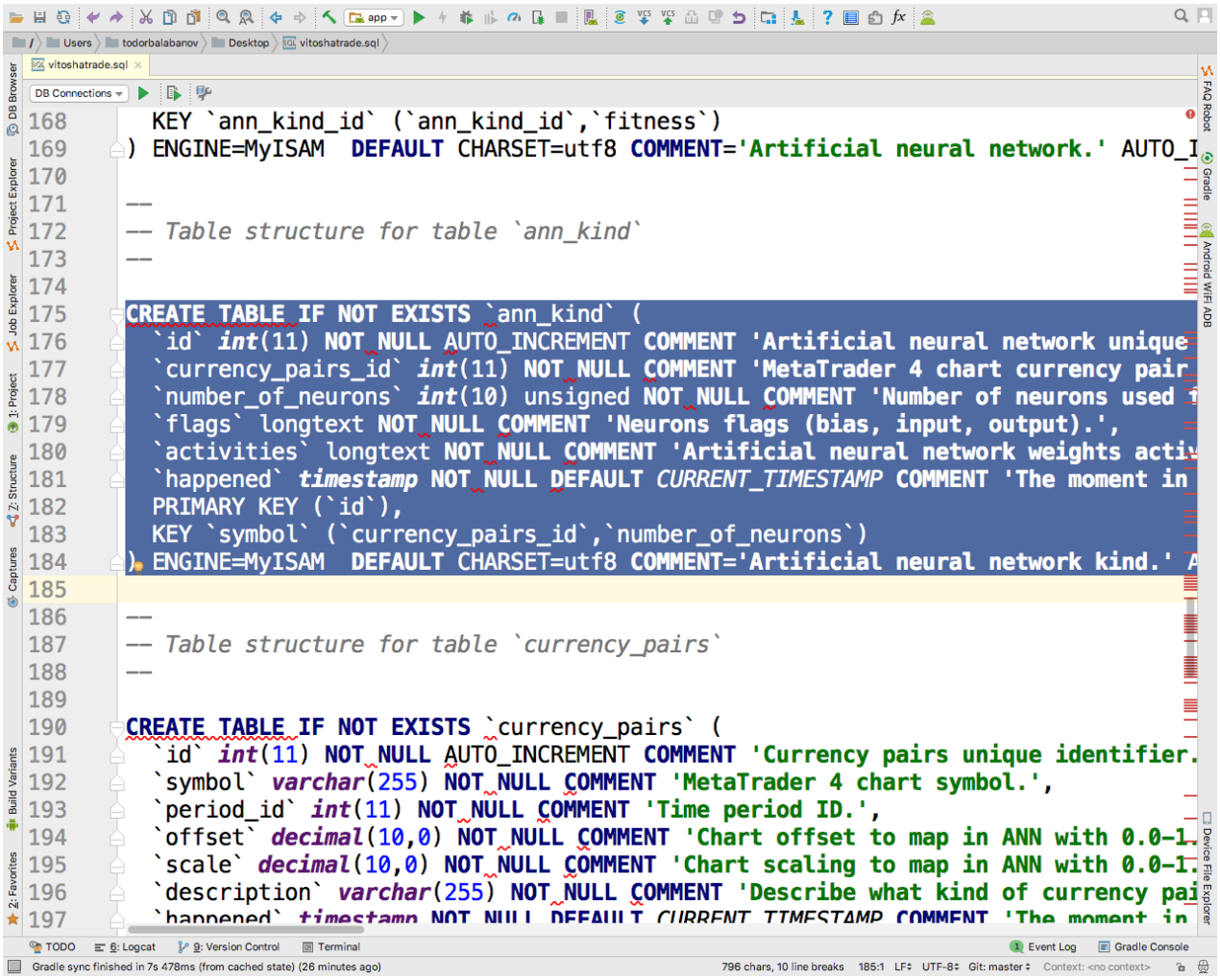
Фигура 5.3: Таблица с времевите интервали за редовете

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.



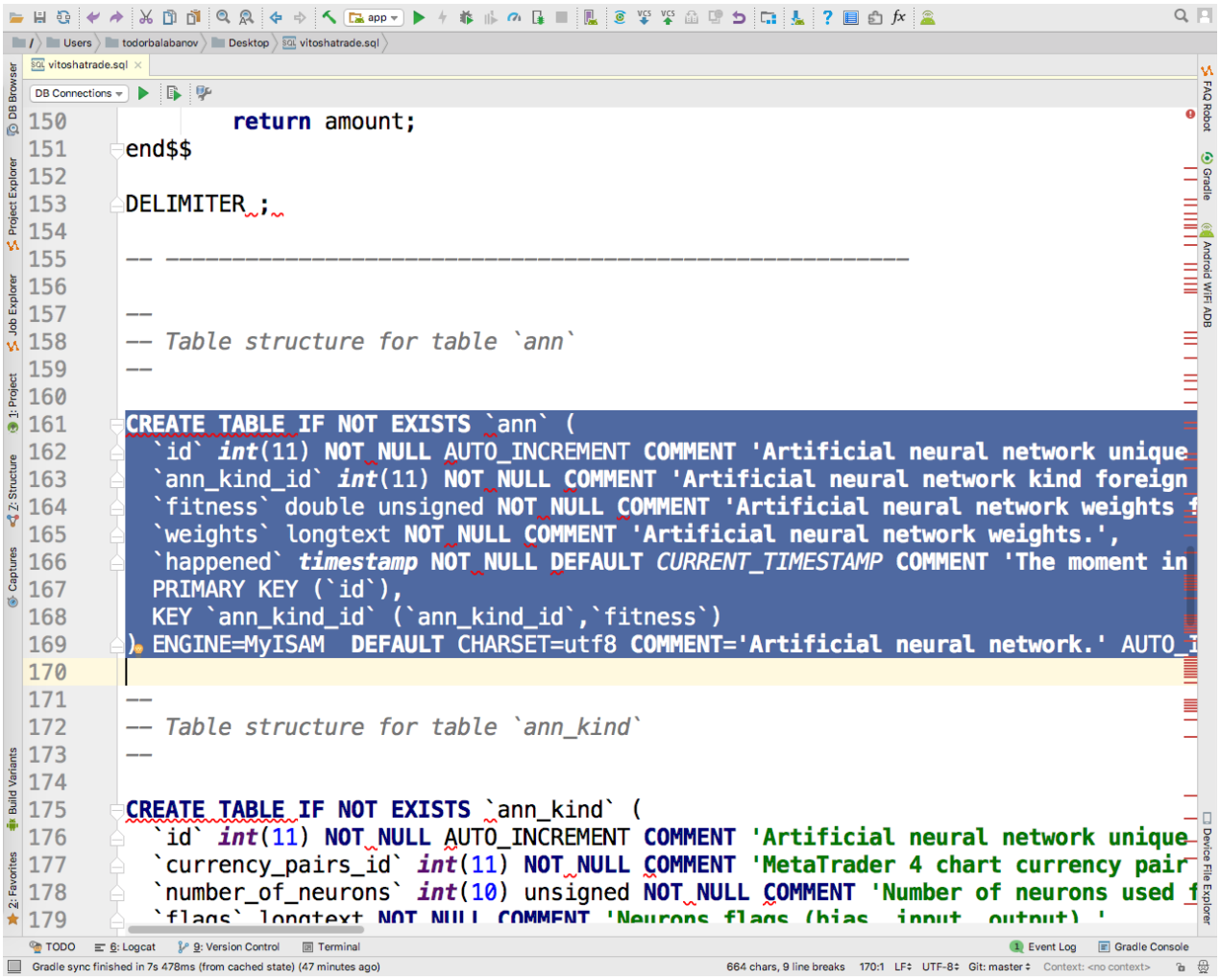
Фигура 5.4: Таблица за представяне на валутните двойки

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.



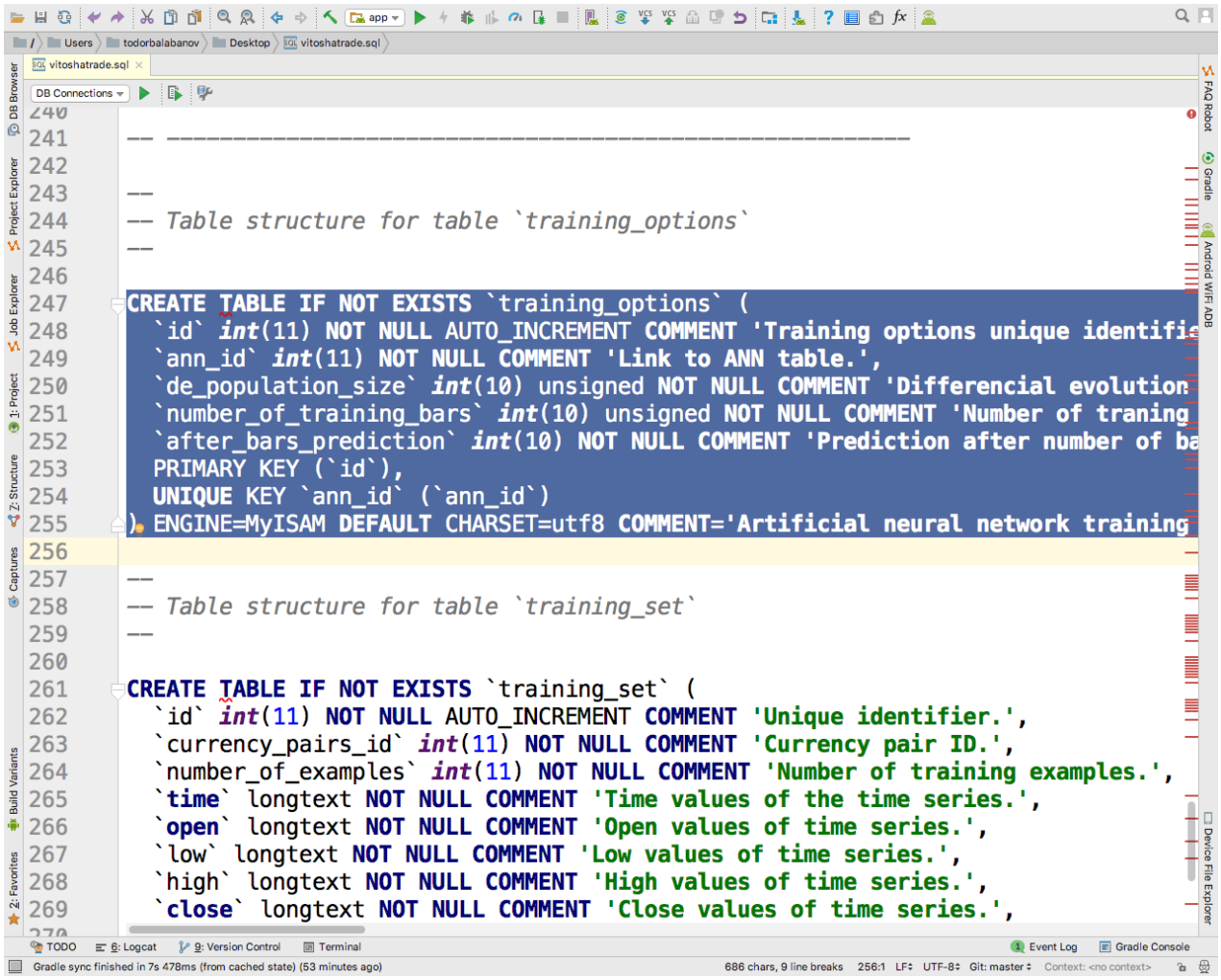
Фигура 5.5: Таблица за представяне на типовете мрежи

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.



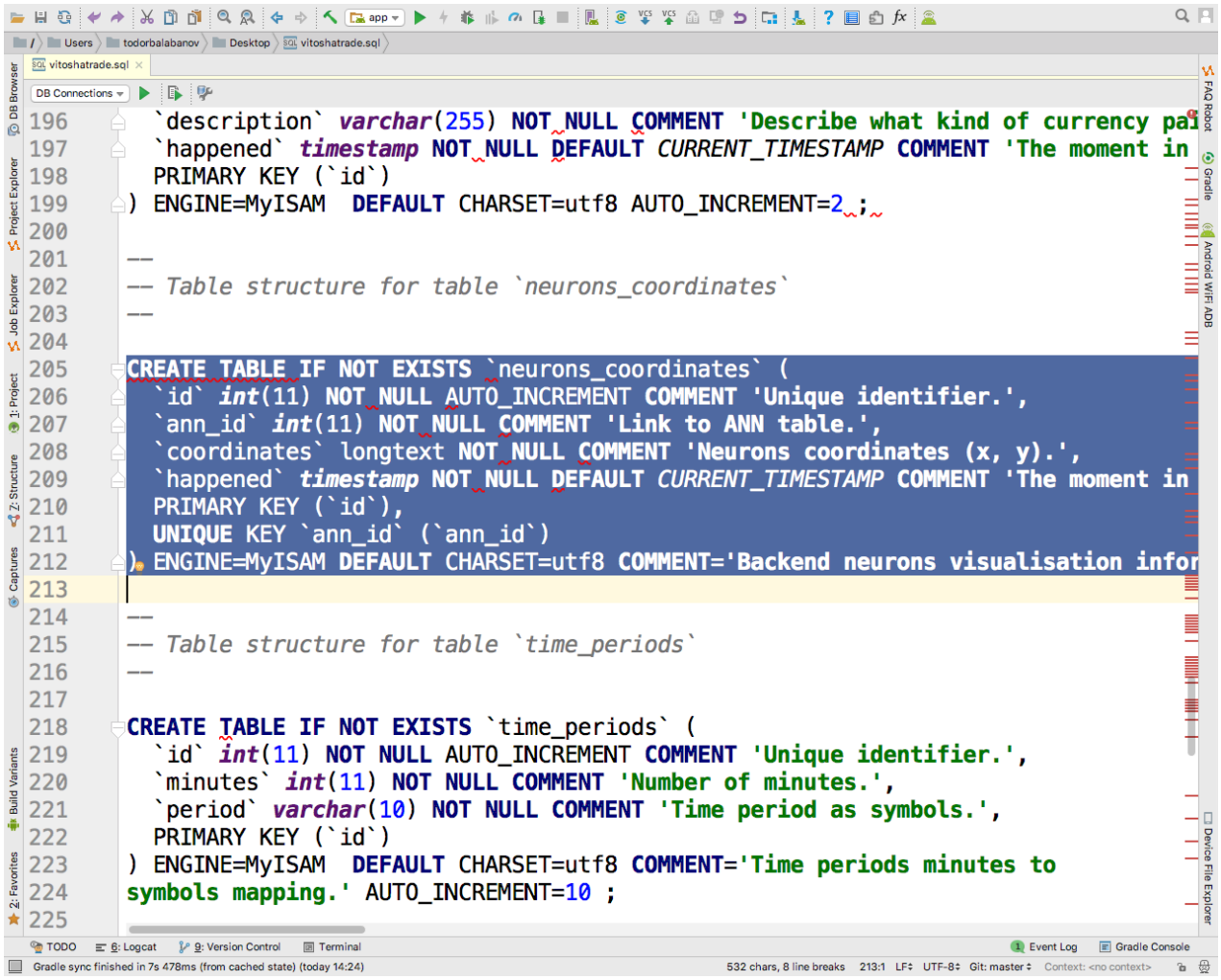
Фигура 5.6: Таблица за представяне на типовете мрежи

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.



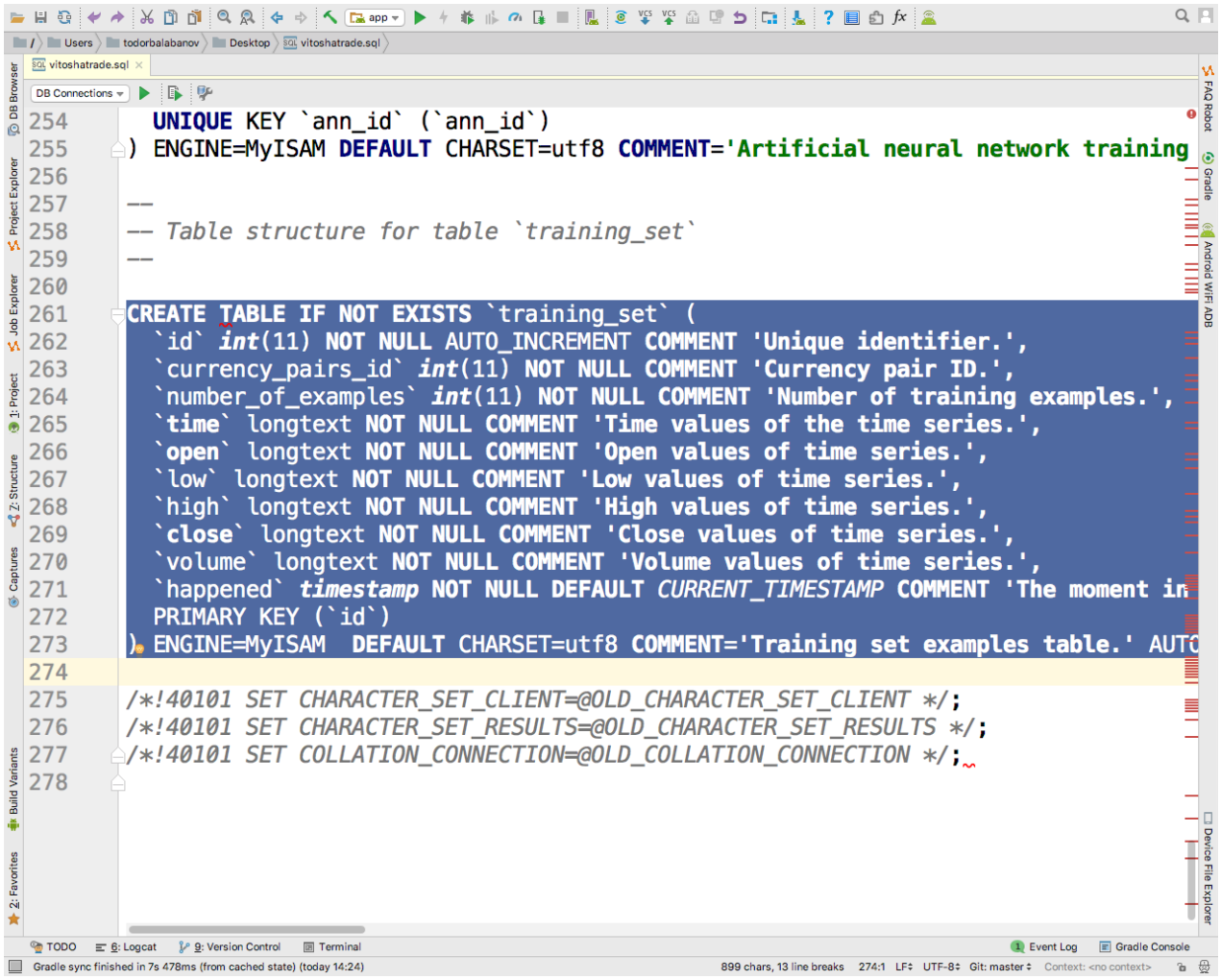
Фигура 5.7: Таблица с опции за протичане на обучението

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.



Фигура 5.8: Таблица с опции за протичане на обучението

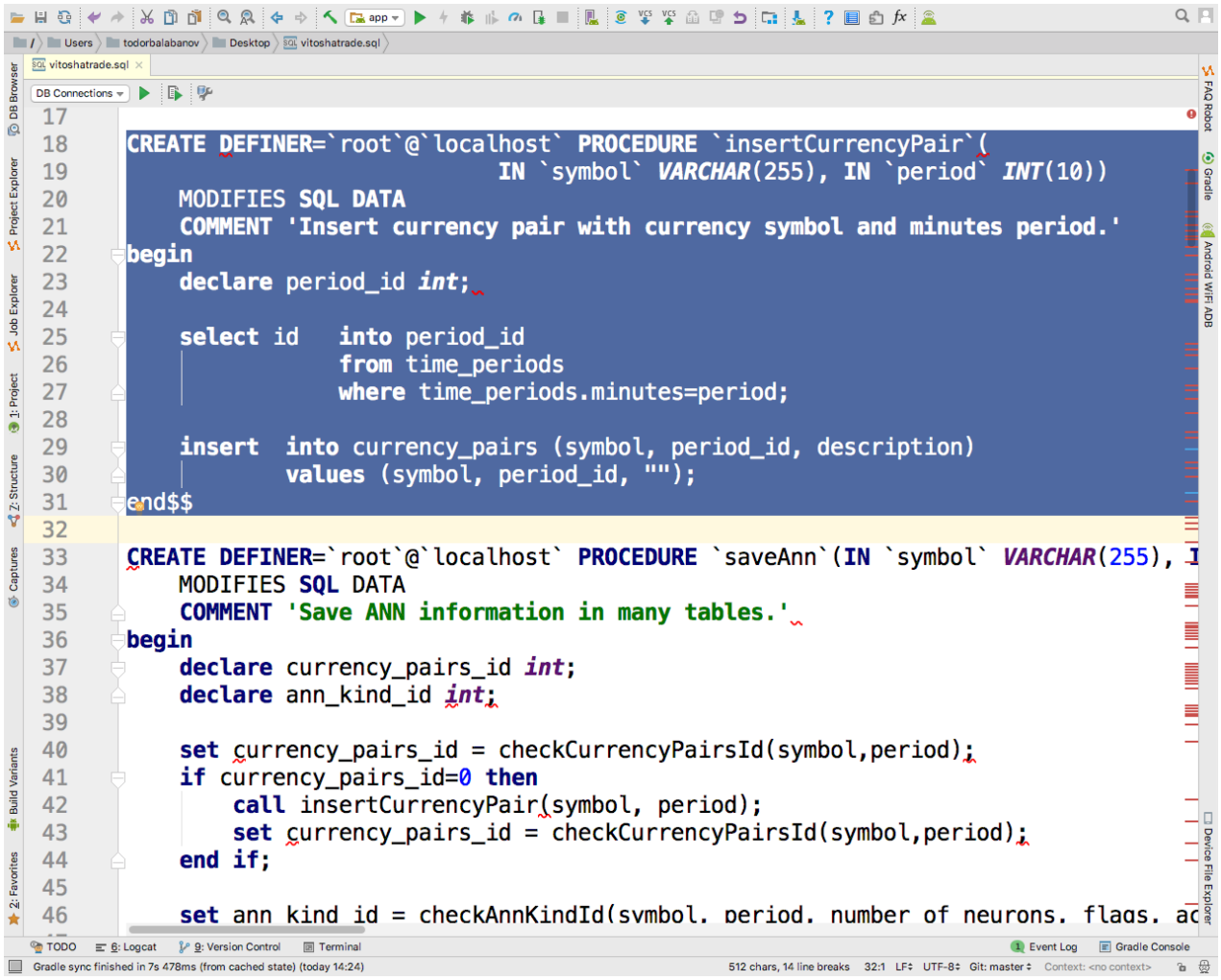
Последната, но изключително важна таблица, е таблицата за съхранение на обучаващите примери, съдържащи информацията за финансовия времеви ред (Фиг. 5.9). Тази таблица съдържа следните колони - служебен идентификатор, външен ключ към валутната двойка, брой измервания, последователност от времеви маркери,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



Фигура 5.9: Таблица с опции за протичане на обучението

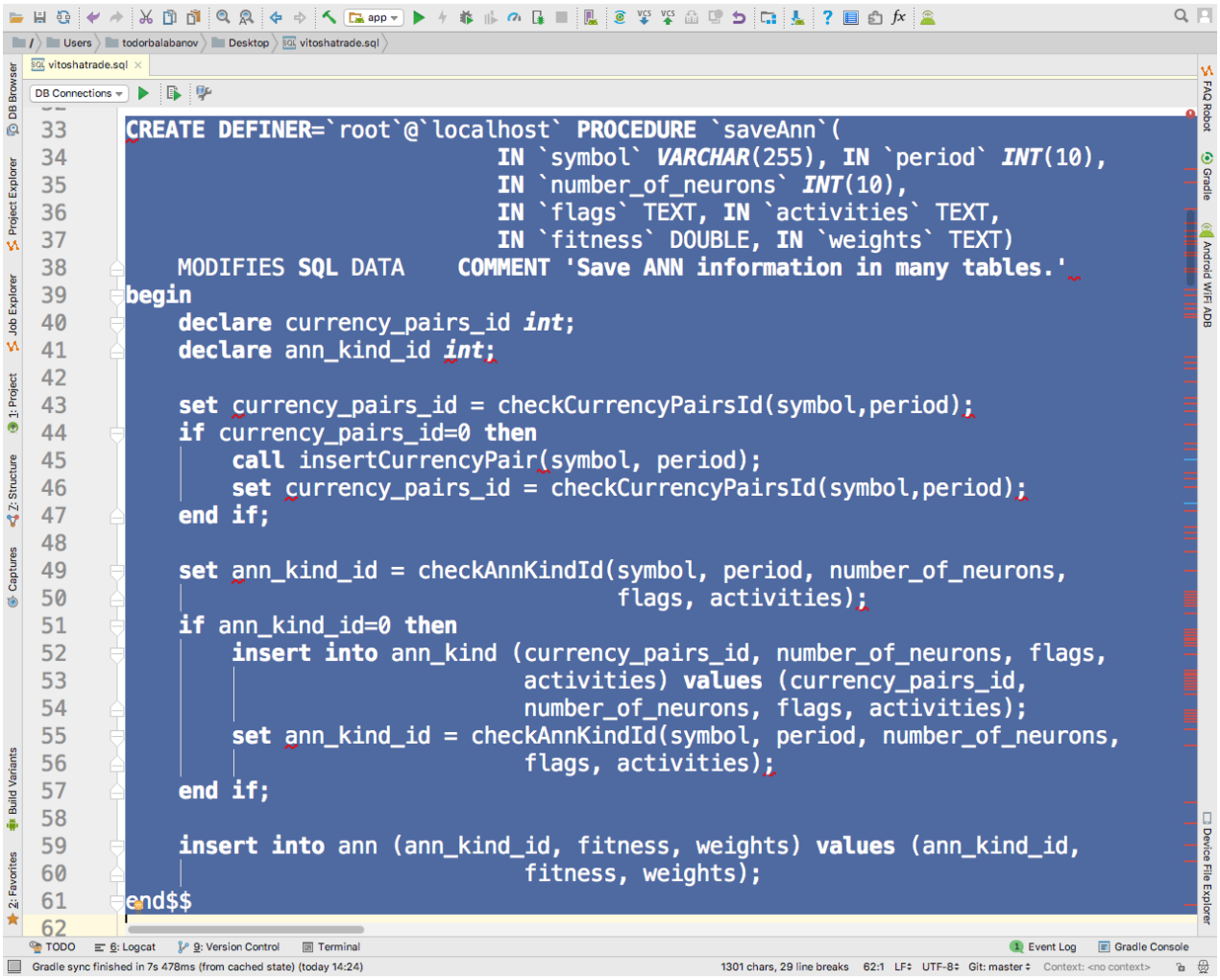
5.2 Алгоритмична обработка на суровите данни

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.



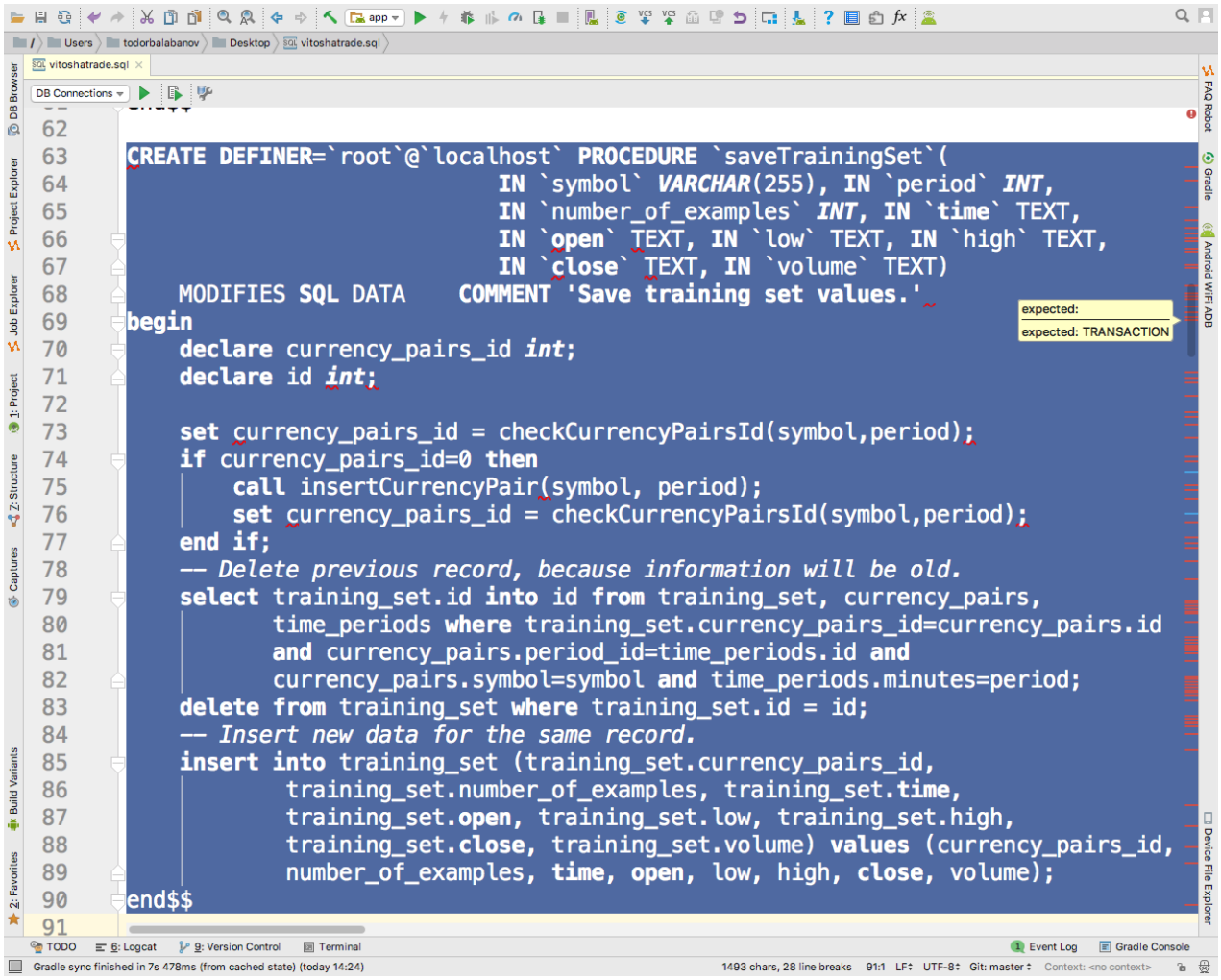
Фигура 5.10: Процедура за добавяне на валутна двойка

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).



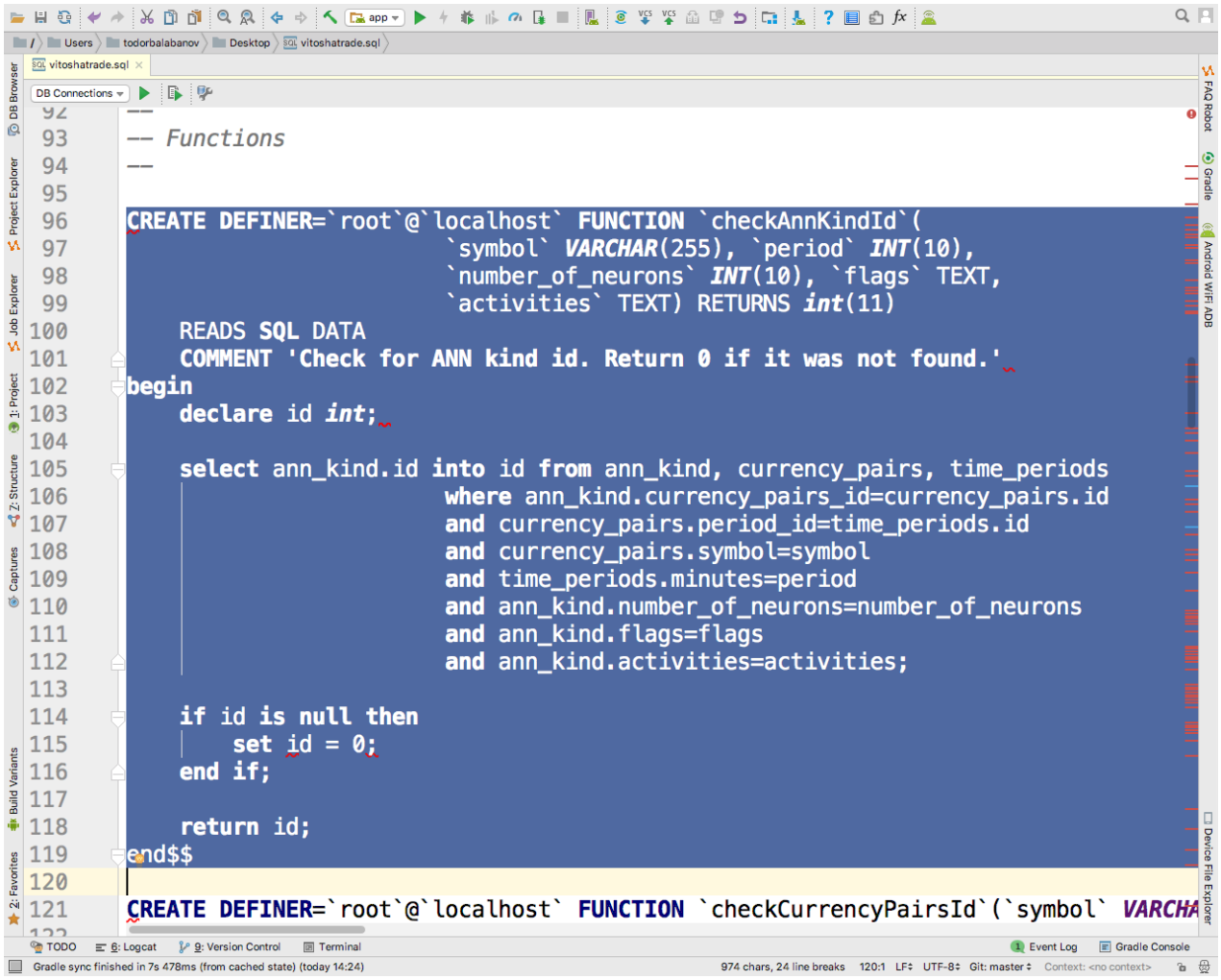
Фигура 5.11: Процедура за запис на екземпляр от изкуствена невронна мрежа

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 дойка 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.



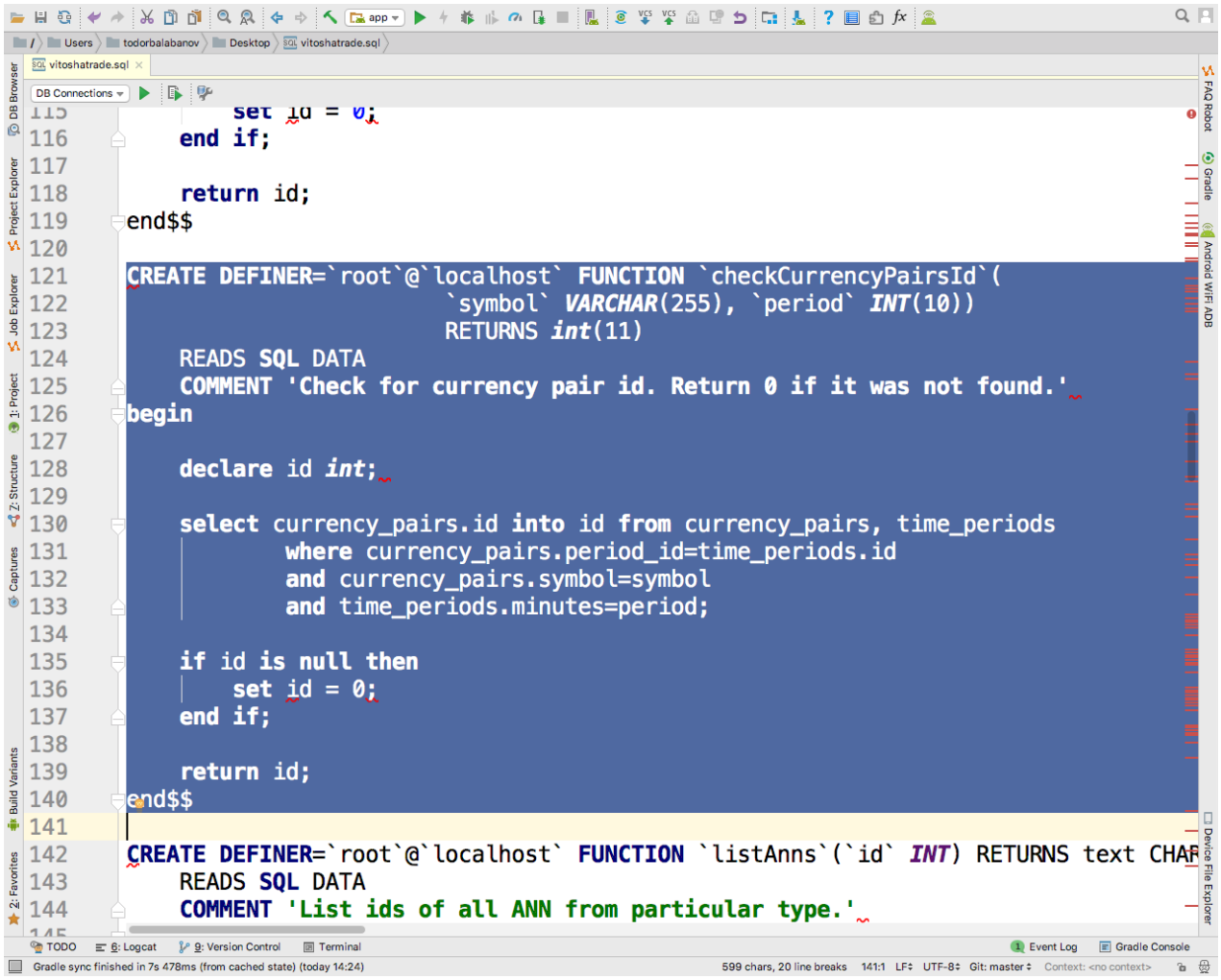
Фигура 5.12: Процедура за запис на тренировъчни примери

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.



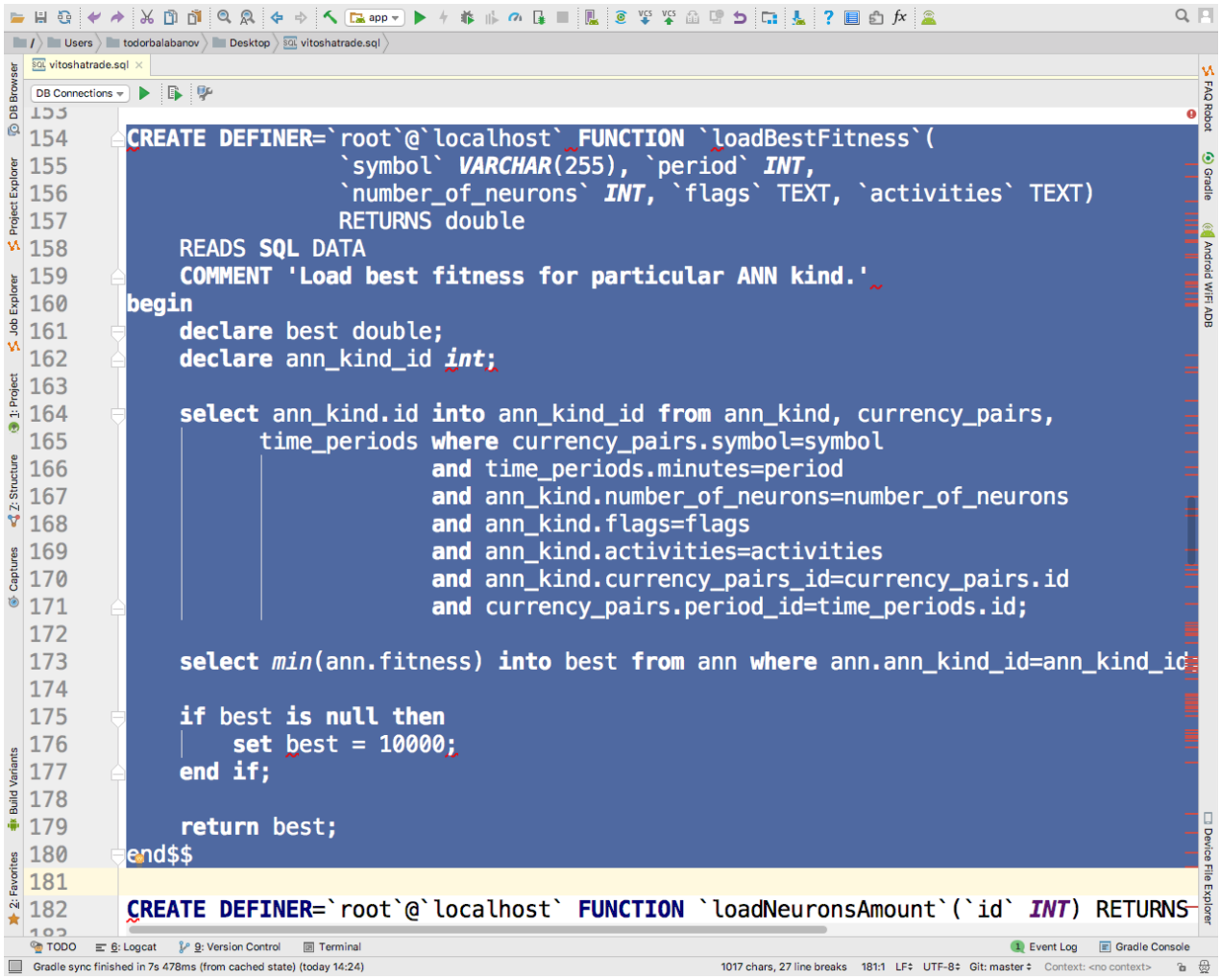
Фигура 5.13: Функция за определяне на типа мрежа

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.



Фигура 5.14: Функция за определяне на типа валутна двойка

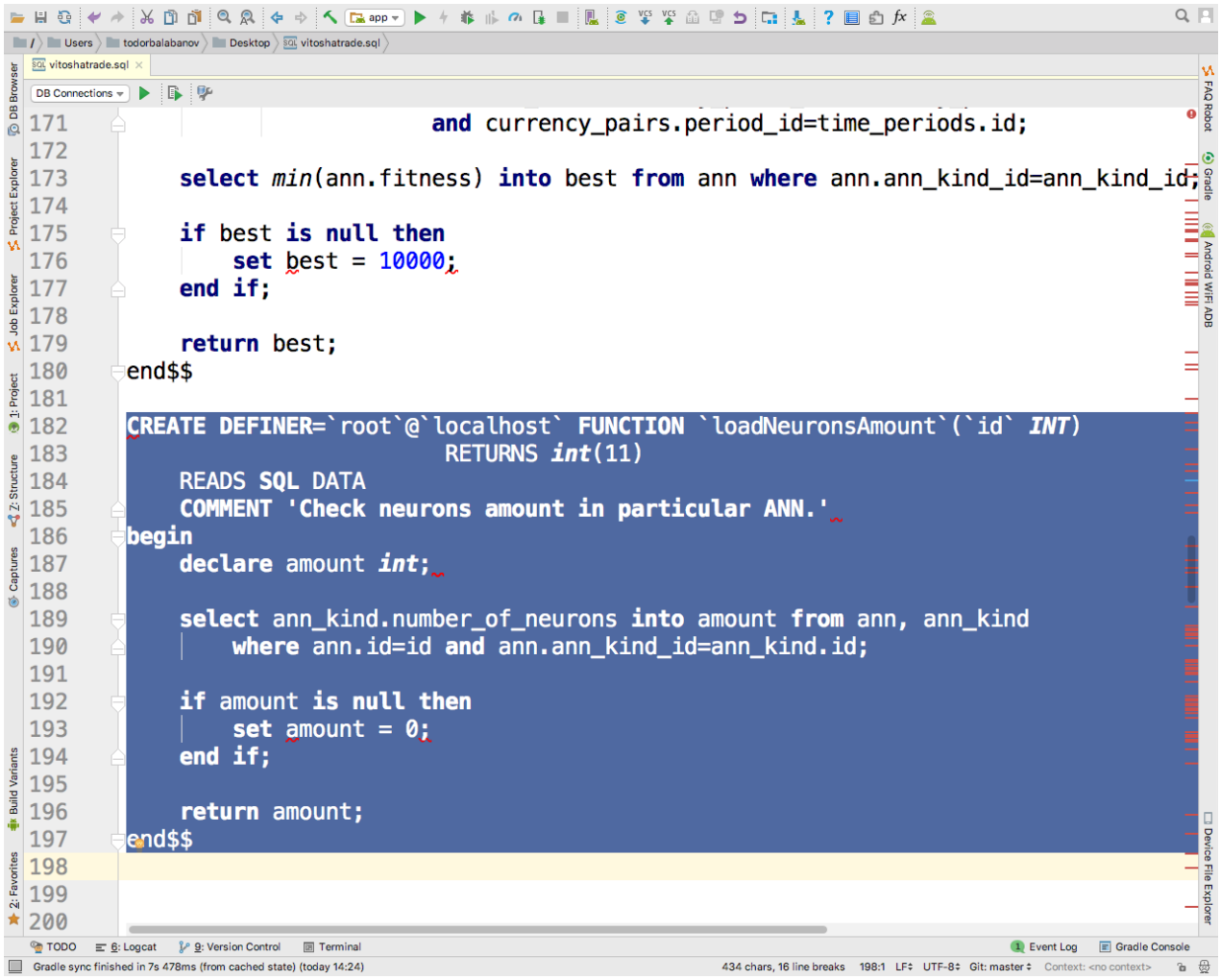
По аналогичен начин, помощна функция служи за определяне на служебния идентификатор за валутна двойка, чрез зададено символно име и период (Фиг. 5.14). Ако валутната двойка не бъде открита се връща служебна нула.



```
CREATE DEFINER=`root`@`localhost` FUNCTION `loadBestFitness`(
                `symbol` VARCHAR(255), `period` INT,
                `number_of_neurons` INT, `flags` TEXT, `activities` TEXT)
                RETURNS double
    READS SQL DATA
    COMMENT 'Load best fitness for particular ANN kind.'
begin
    declare best double;
    declare ann_kind_id int;

    select ann_kind.id into ann_kind_id from ann_kind, currency_pairs,
          time_periods where currency_pairs.symbol=symbol
                      and time_periods.minutes=period
                      and ann_kind.number_of_neurons=number_of_neurons
                      and ann_kind.flags=flags
                      and ann_kind.activities=activities
                      and ann_kind.currency_pairs_id=currency_pairs.id
                      and currency_pairs.period_id=time_periods.id;

    select min(ann.fitness) into best from ann where ann.ann_kind_id=ann_kind_id

    if best is null then
        set best = 10000;
    end if;

    return best;
end$$

CREATE DEFINER=`root`@`localhost` FUNCTION `loadNeuronsAmount`(`id` INT) RETURNS
```

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).

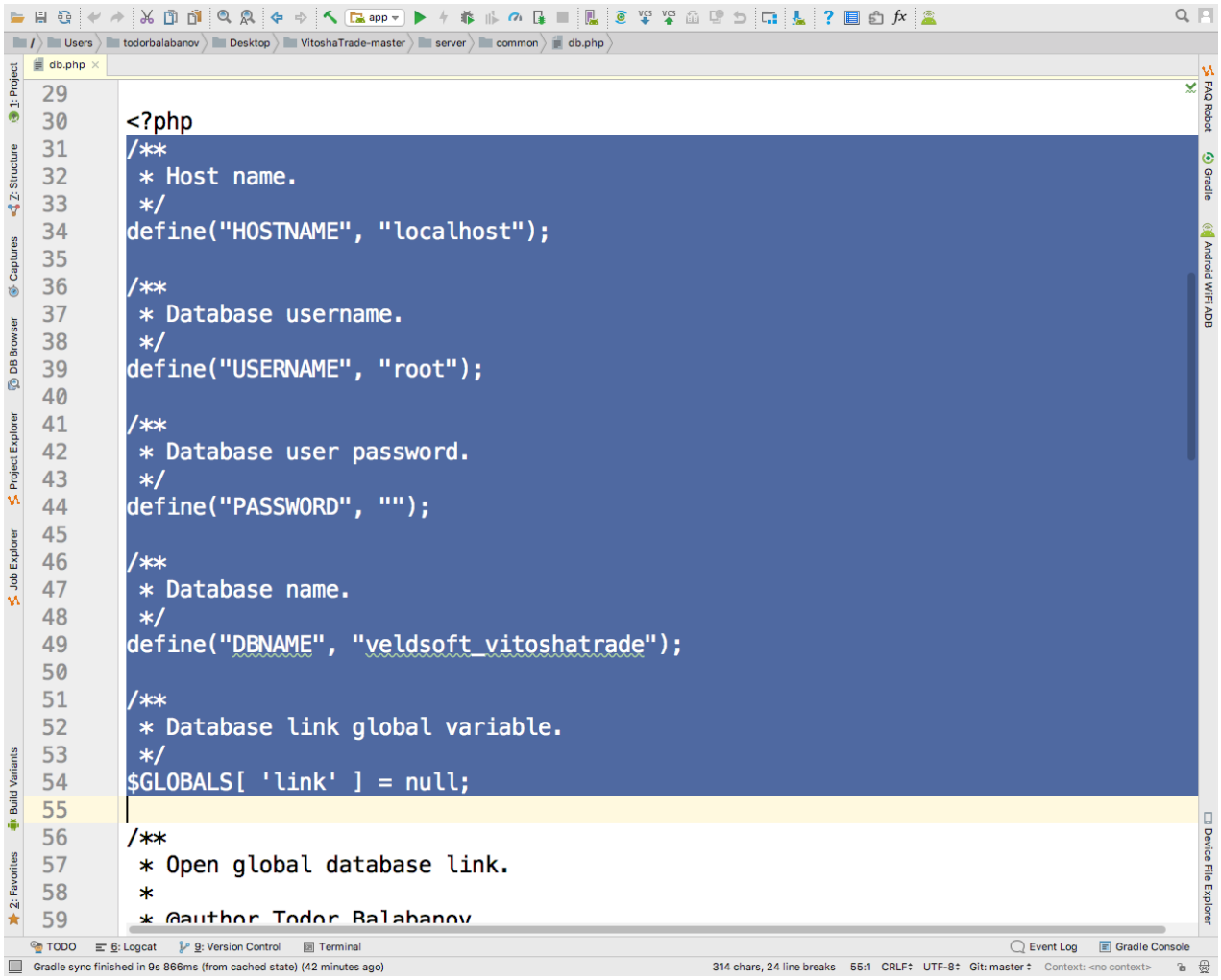


Фигура 5.16: Функция за определяне на броя неврони за определен екземпляр мрежа

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

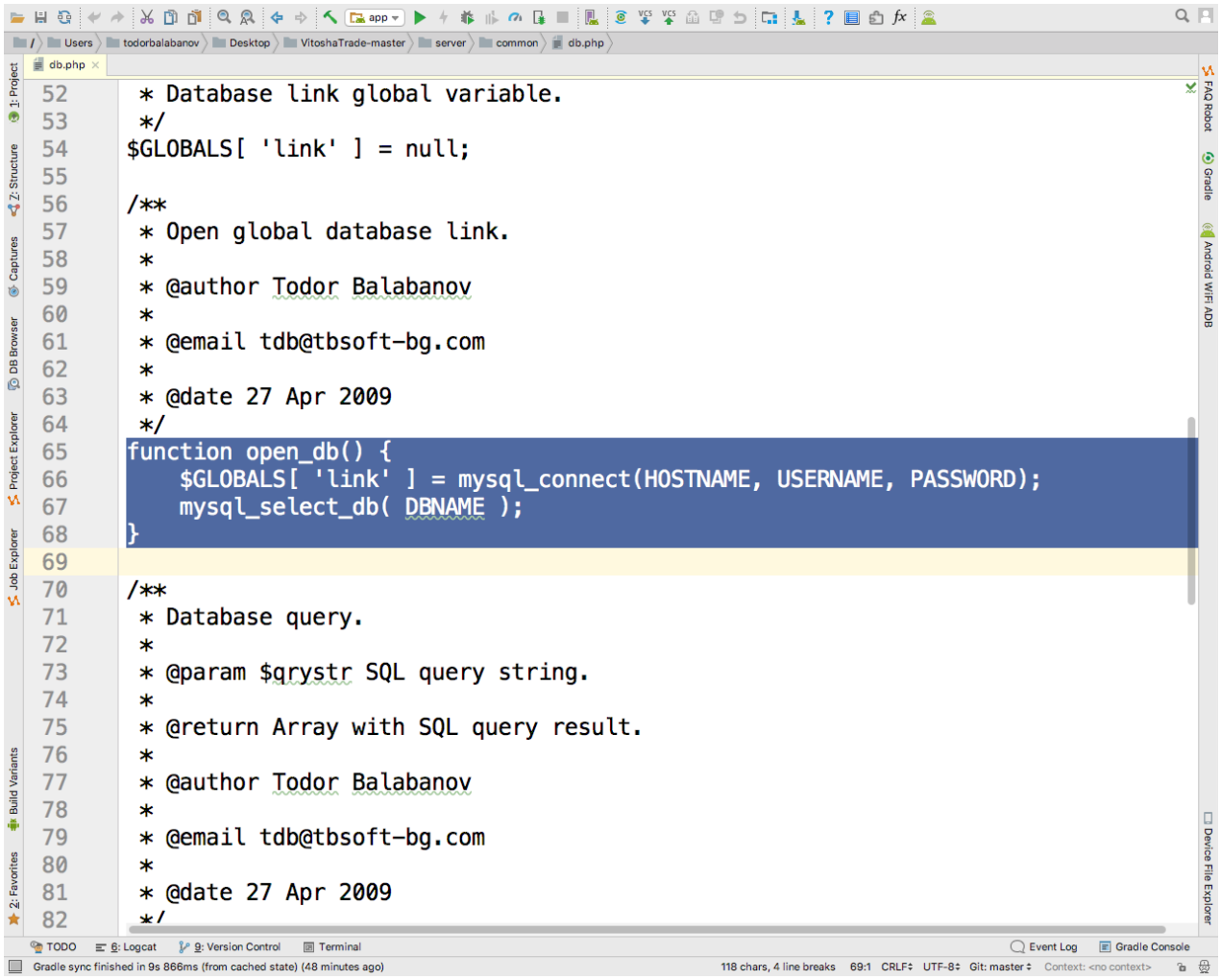
5.3 Сървър скриптове

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.



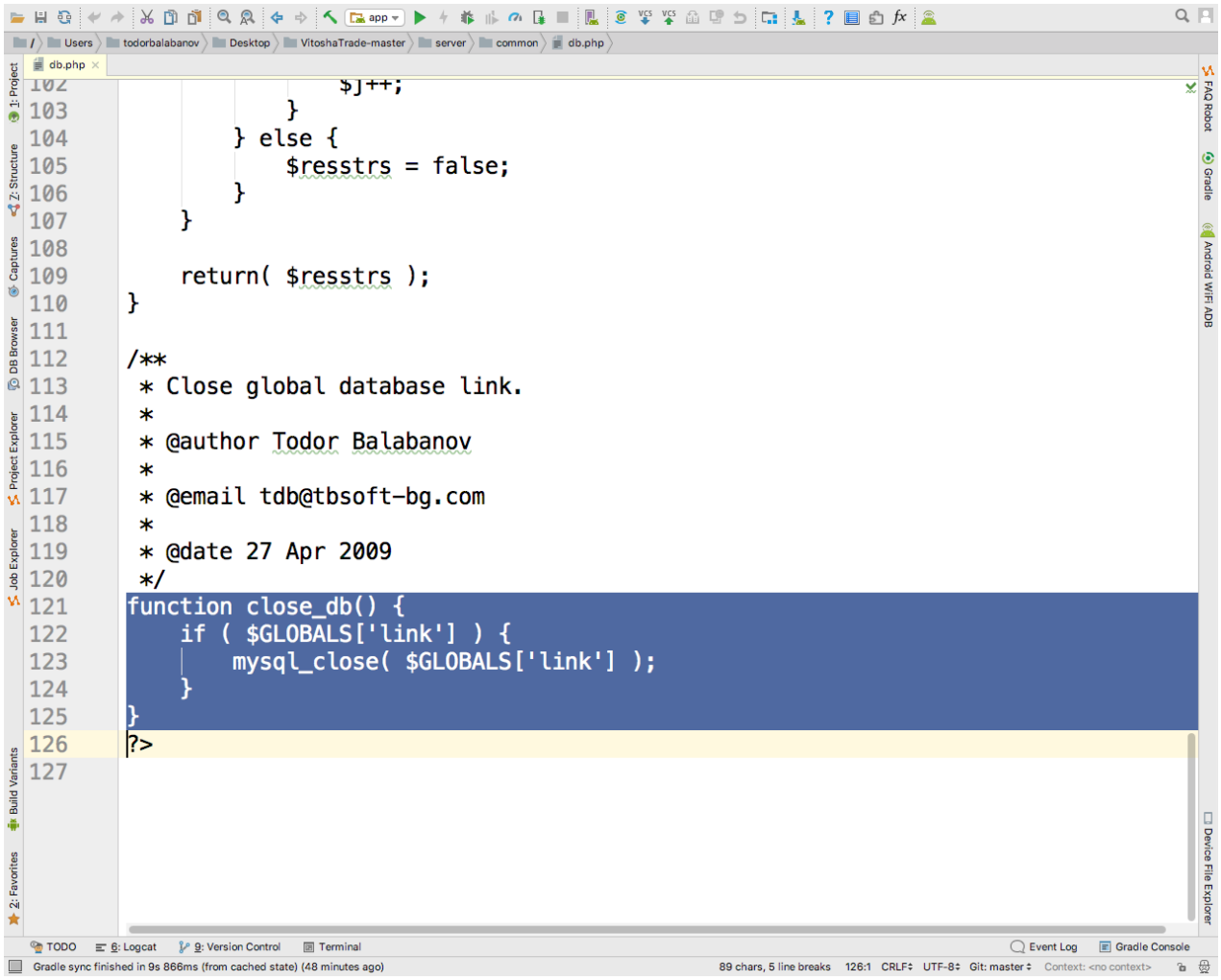
Фигура 5.17: Глобални променливи за достъп до базата данни

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).



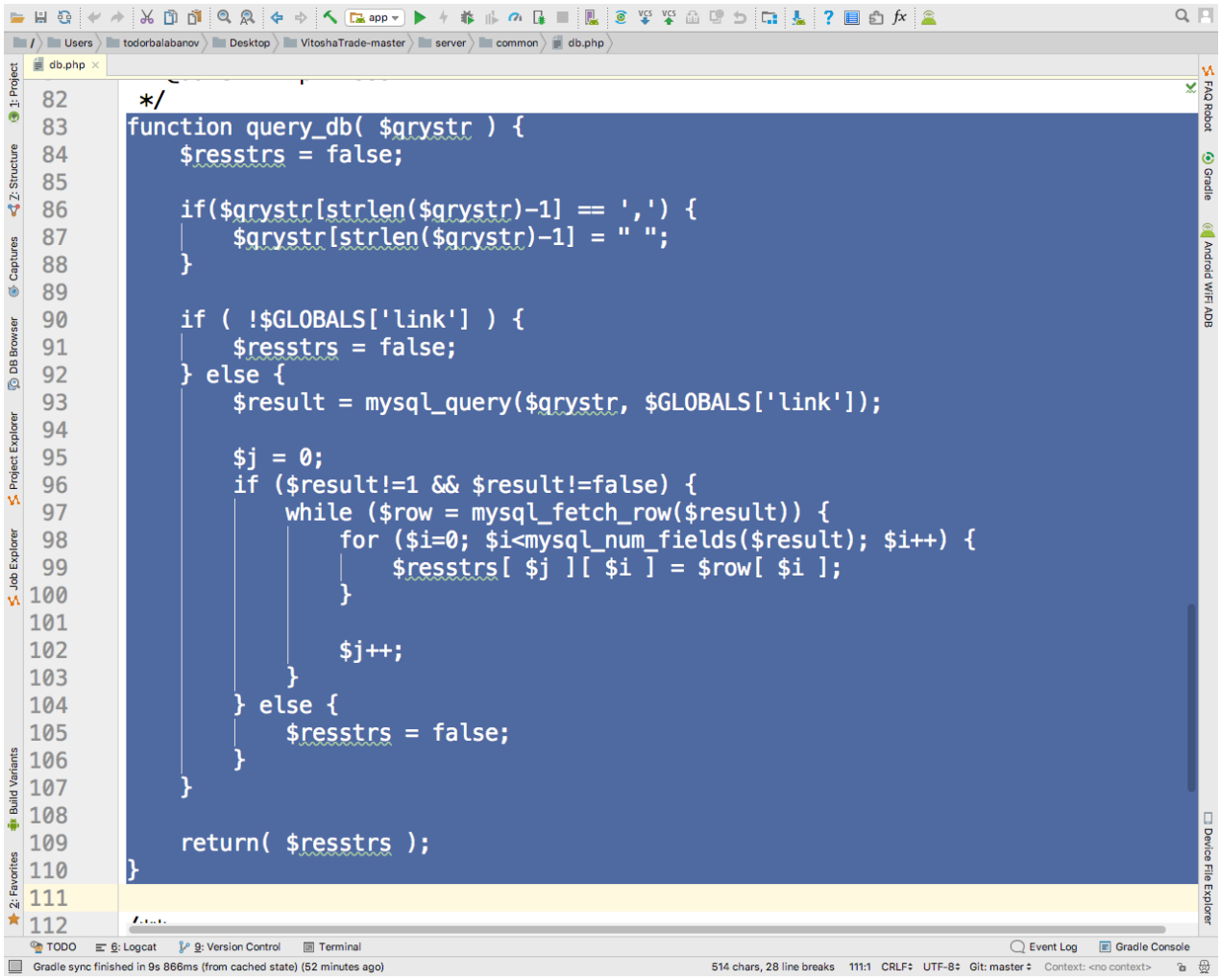
Фигура 5.18: Отваряне на връзка към базата данни

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



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).

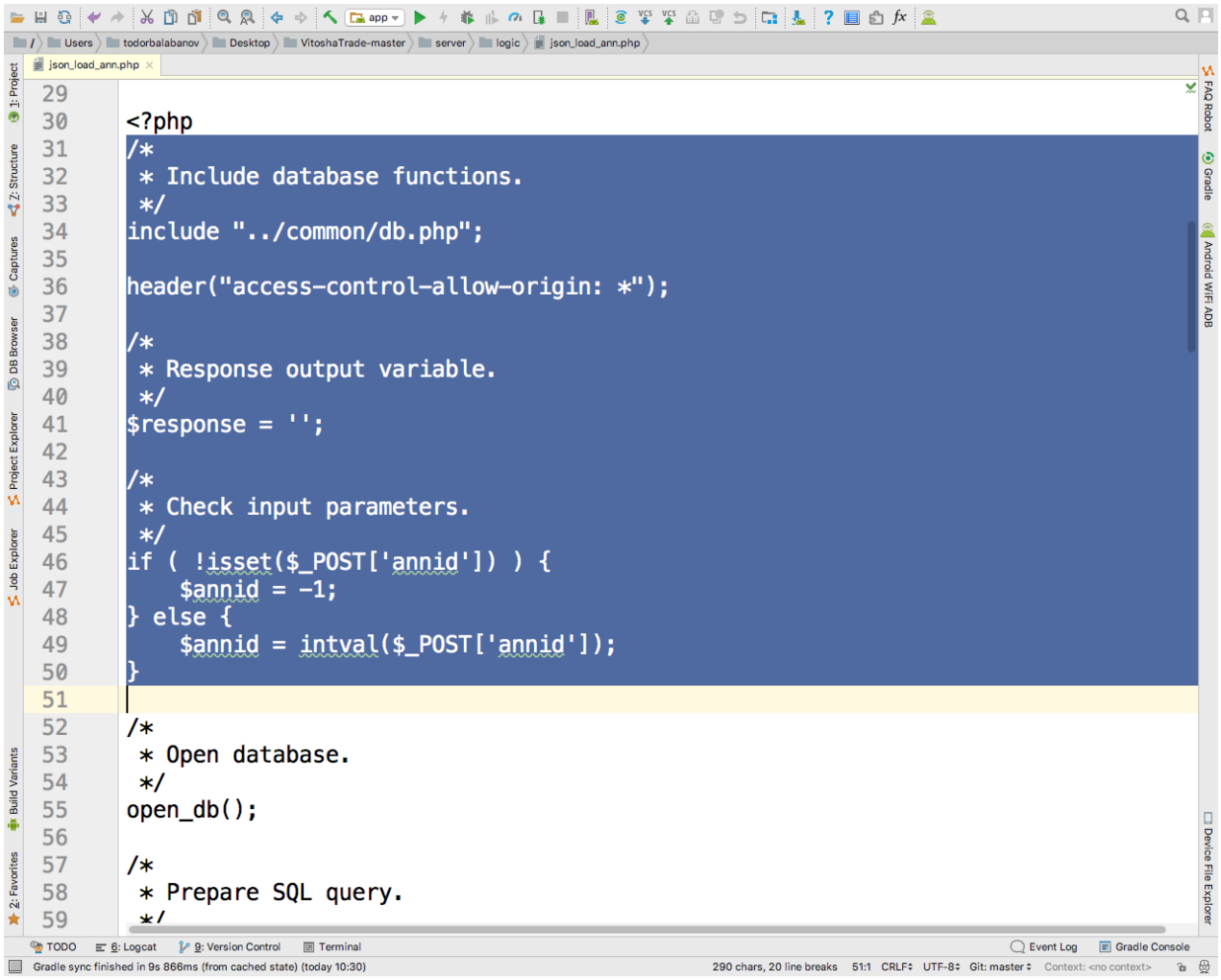


Фигура 5.20: Изпълнение на заявки към базата данни

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.

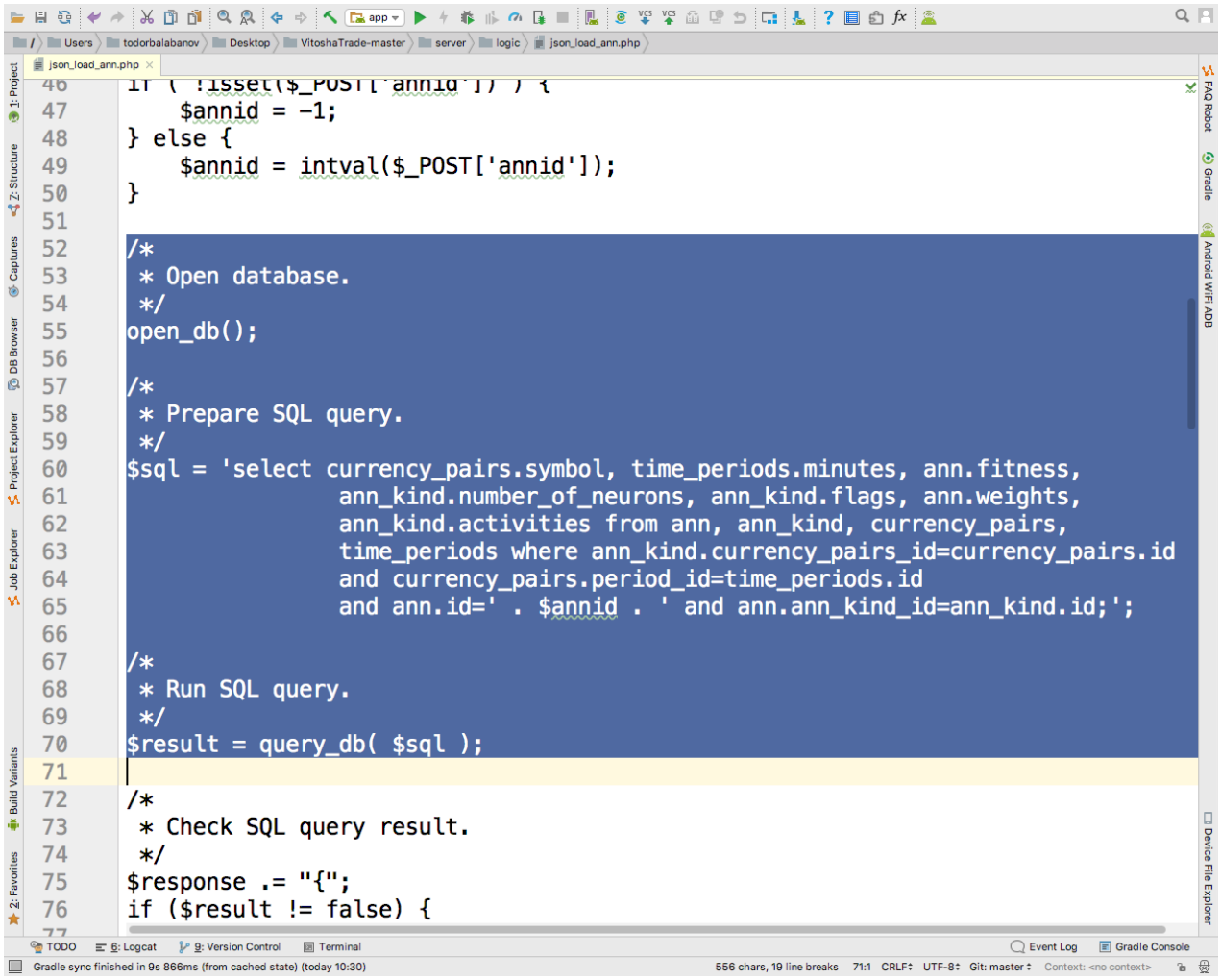
5.3.1 Зареждане на мрежа по идентификатор

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.



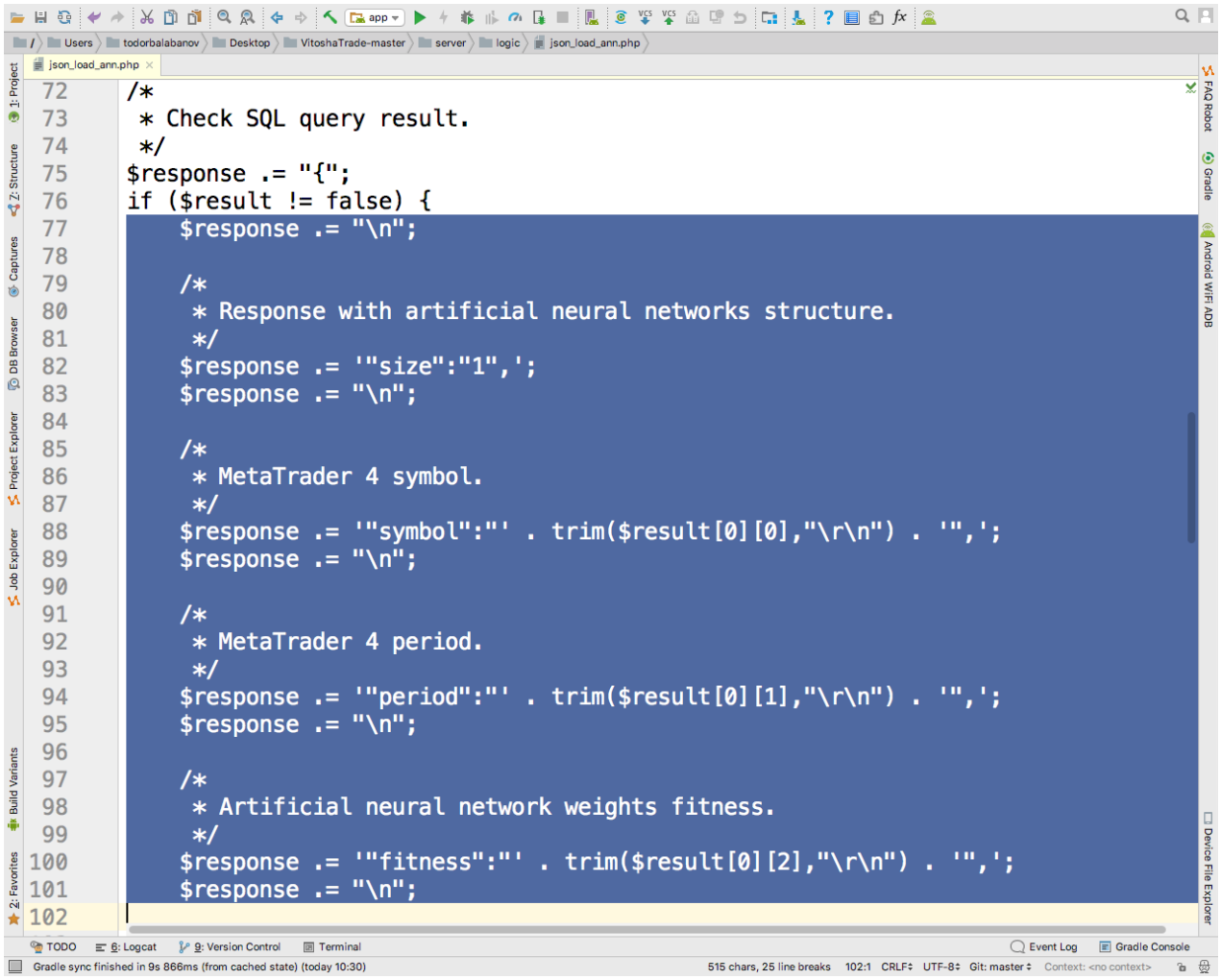
Фигура 5.21: Проверка на входните данни

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).



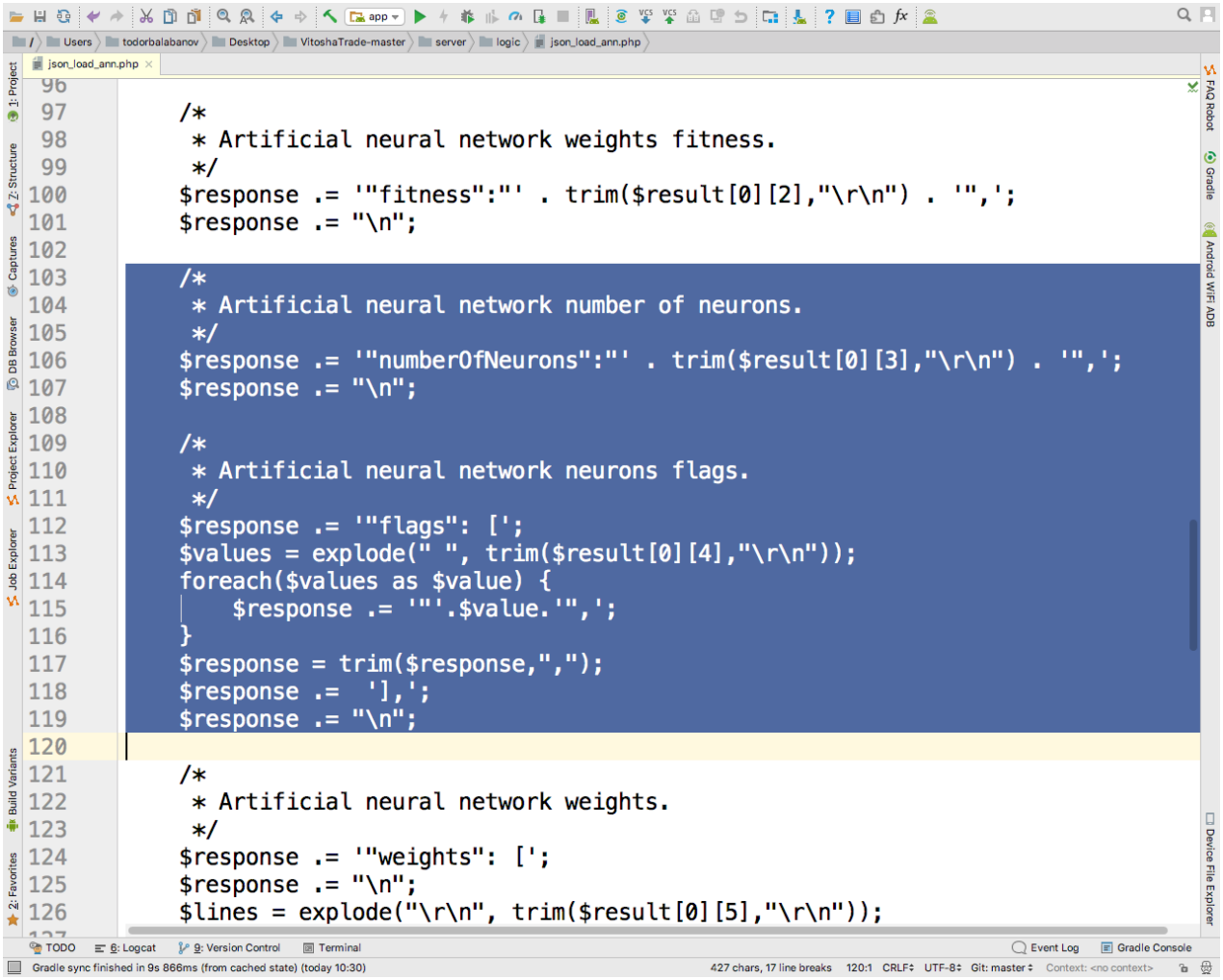
Фигура 5.22: Заявка за извличане на екземпляр по идентификатор

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

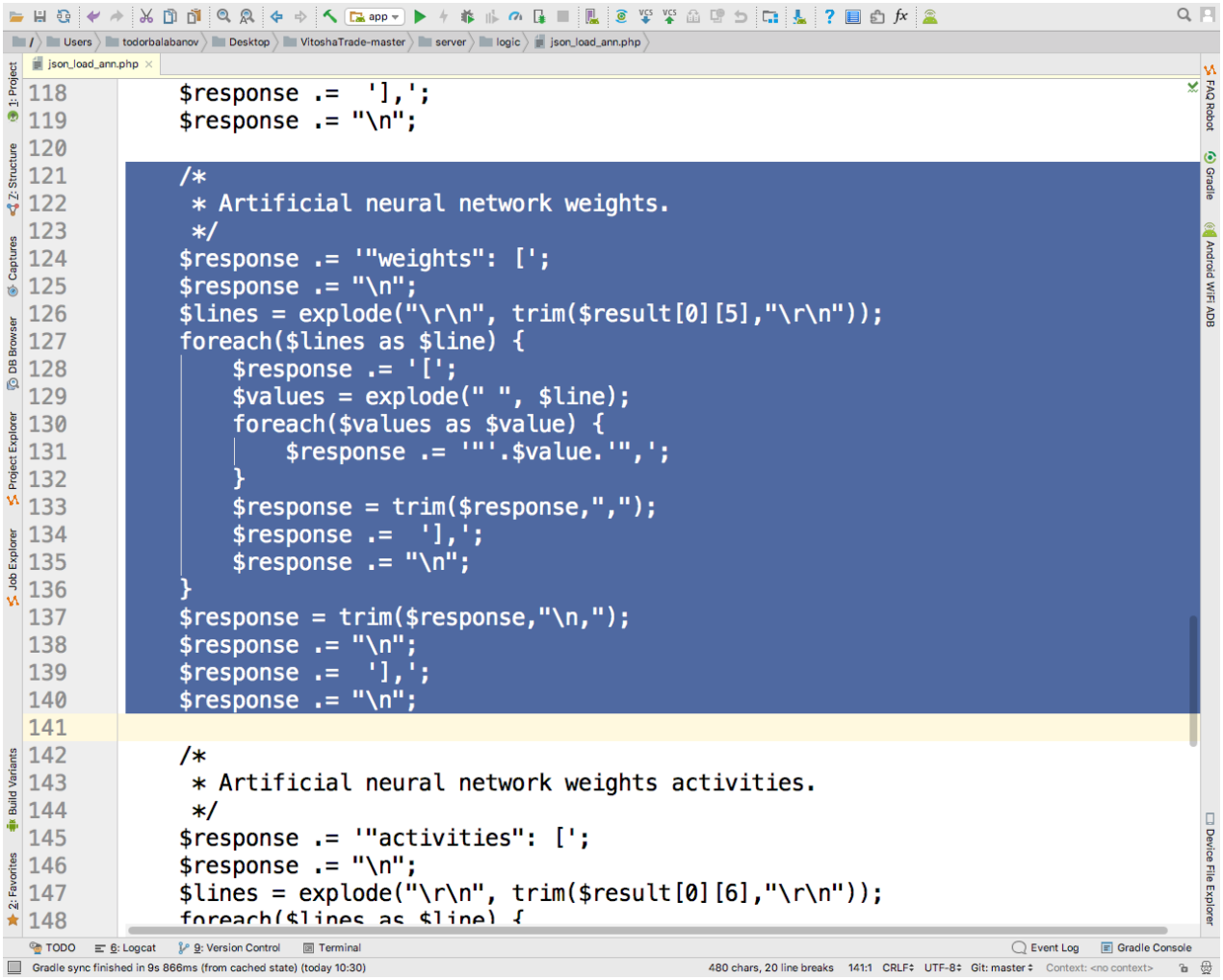


Фигура 5.23: Проверка за наличност на екземпляра

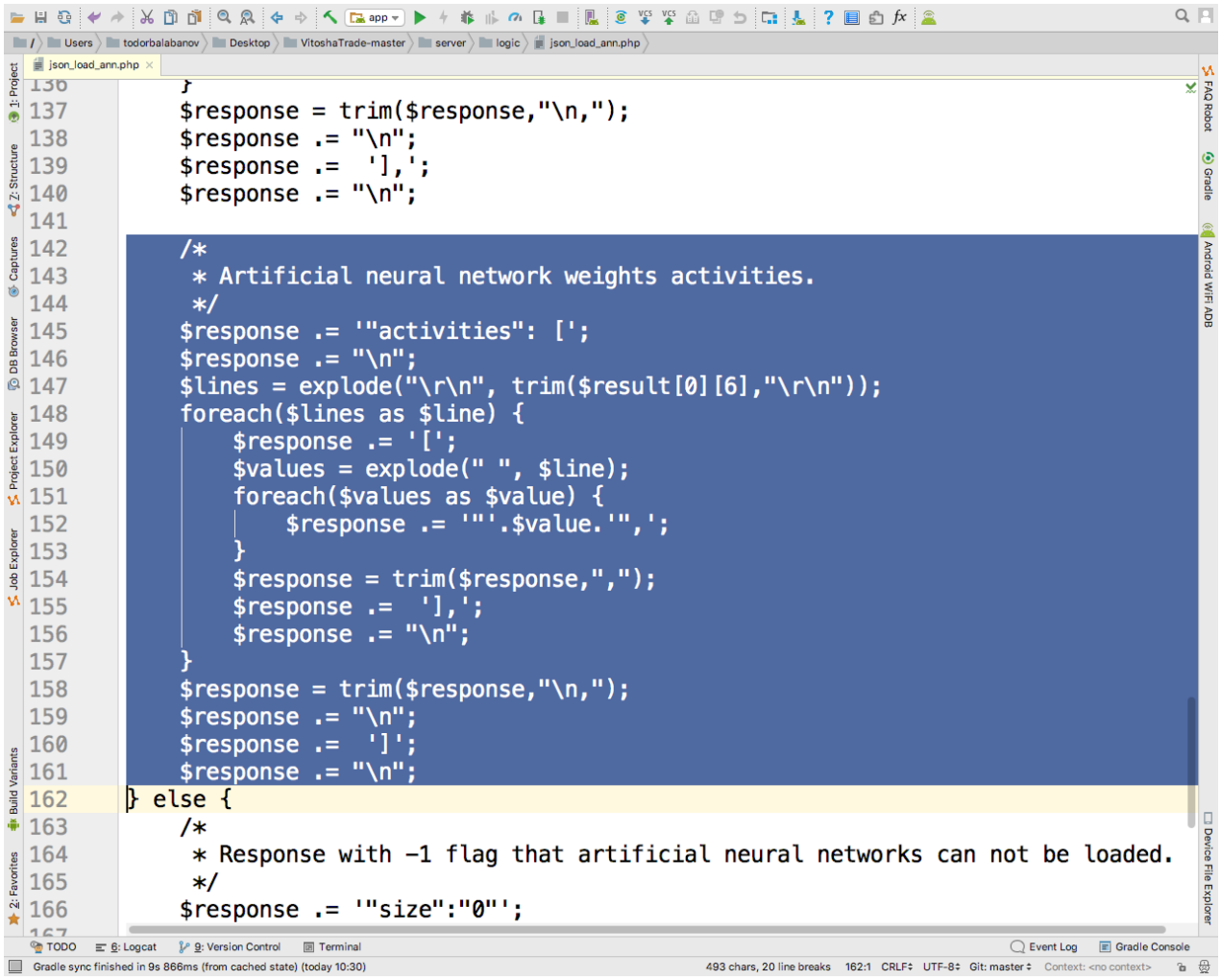
Възможни са два изхода от изпълнението на заявката, спрямо това дали такъв екземпляр присъства в базата данни или не присъства. Ако екземплярът бъде открит започва изграждането на JSON съобщение с отговор към клиента. Отговорът съдържа единица, за намерен екземпляр, названието на валутната двойка, интервала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



Фигура 5.24: Информация за невроните

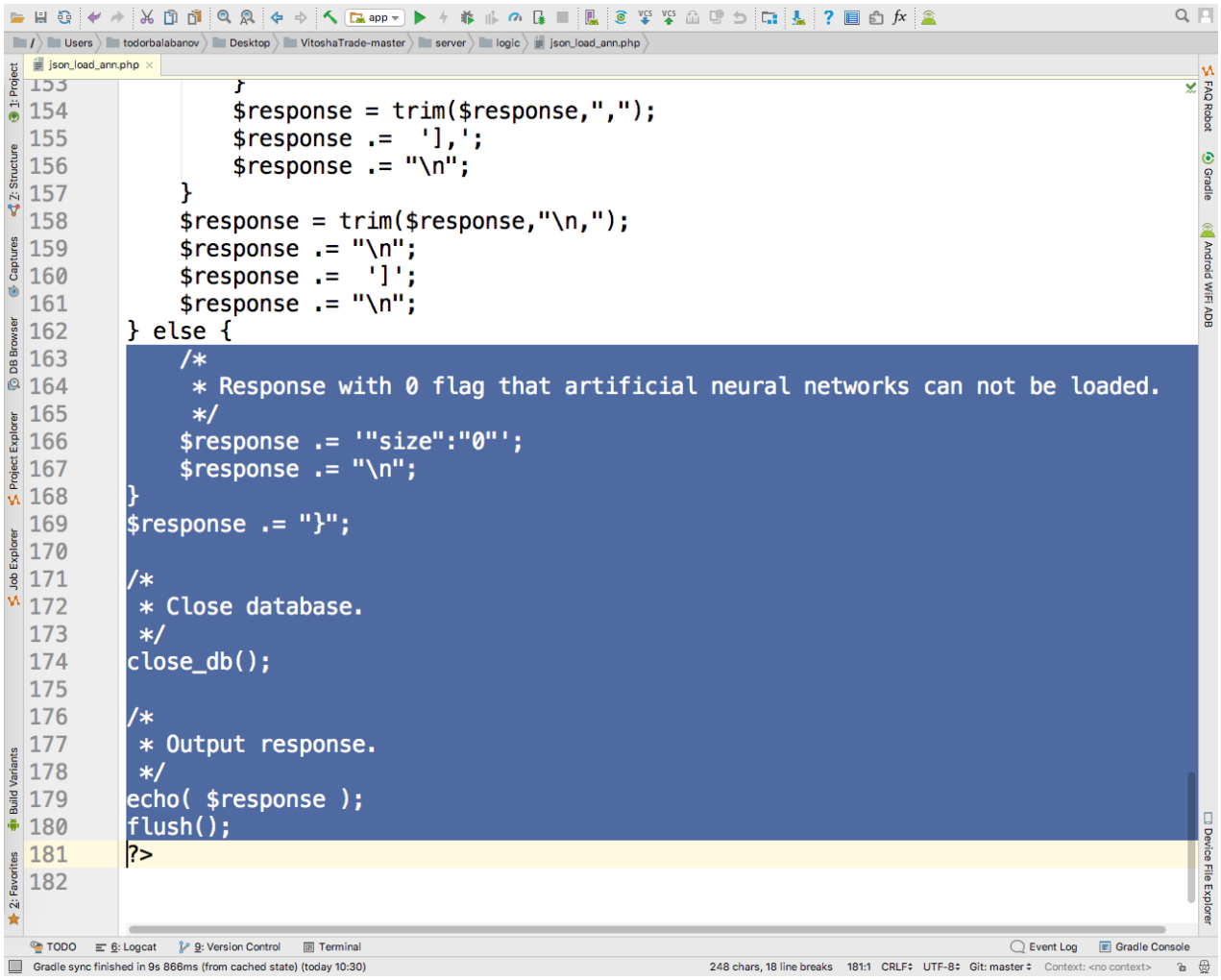


Фигура 5.25: Информация за връзките



Фигура 5.26: Информация за теглата

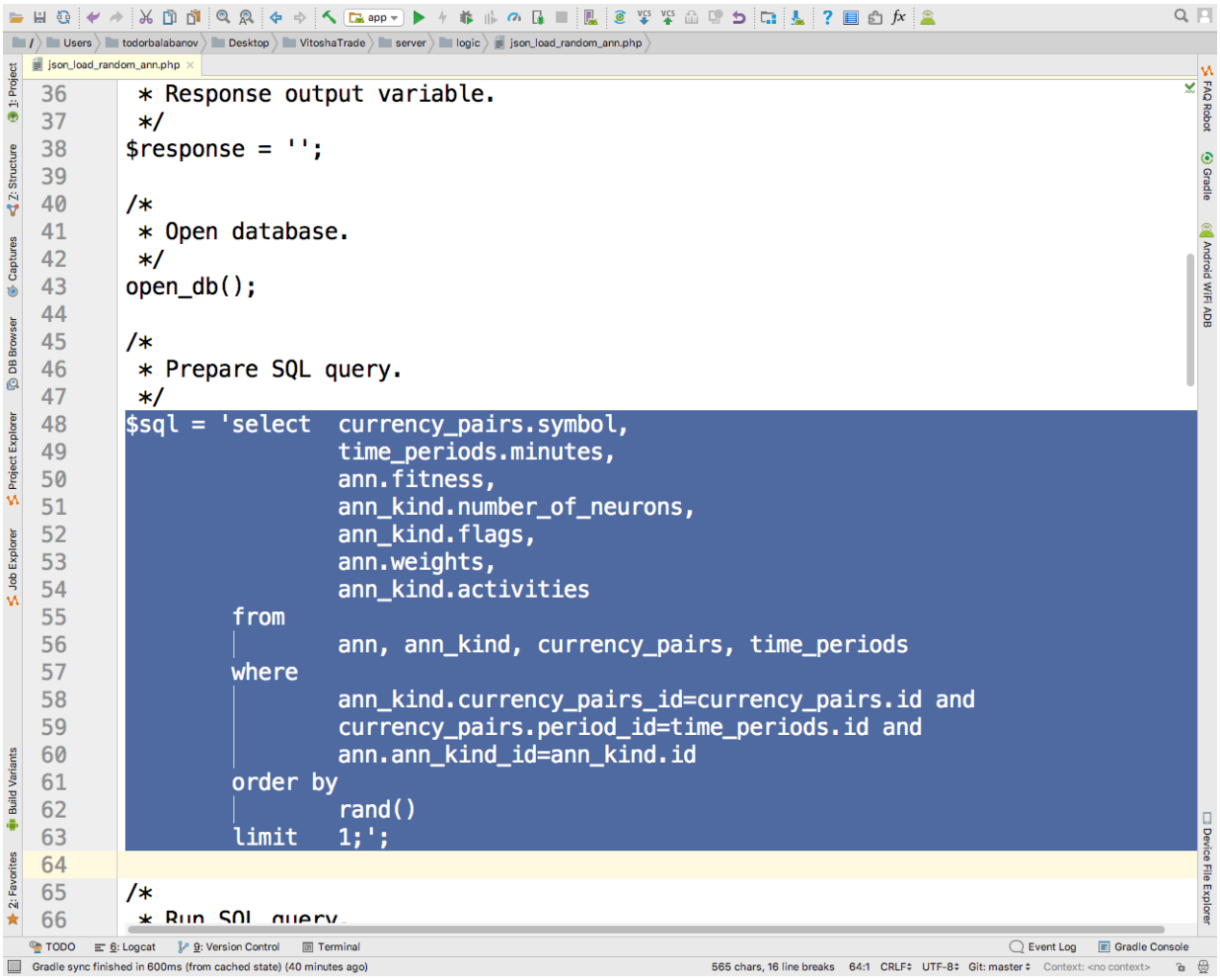
В ситуацията, когато екземплярът не е открит се връща нулев флаг. И при двата случая (открит или не) JSON съобщението се завършва, базата данни се затваря и информацията се изпраща към клиента (Фиг. 5.27).



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.

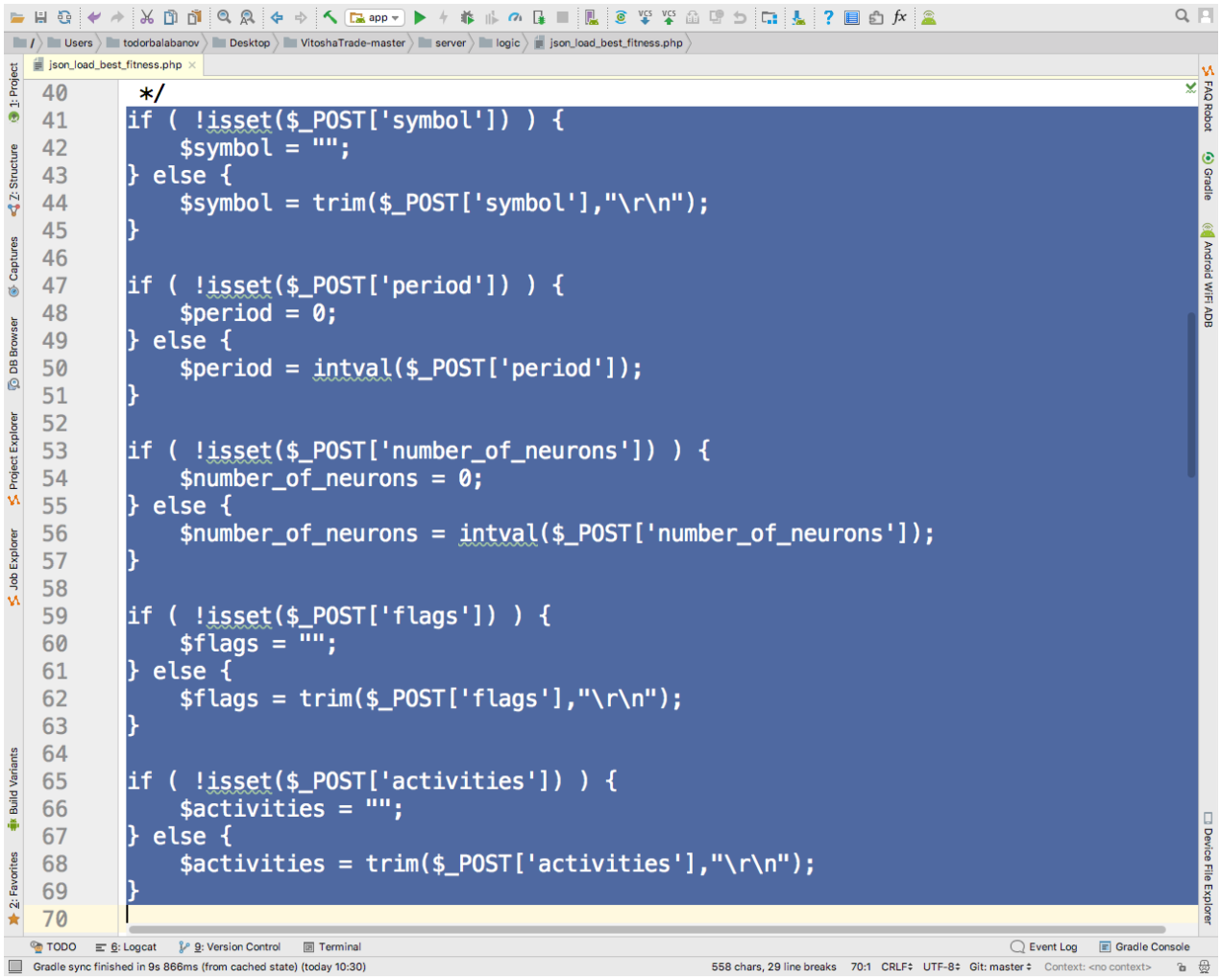


Фигура 5.28: Заявка за зареждане на случайна мрежа

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

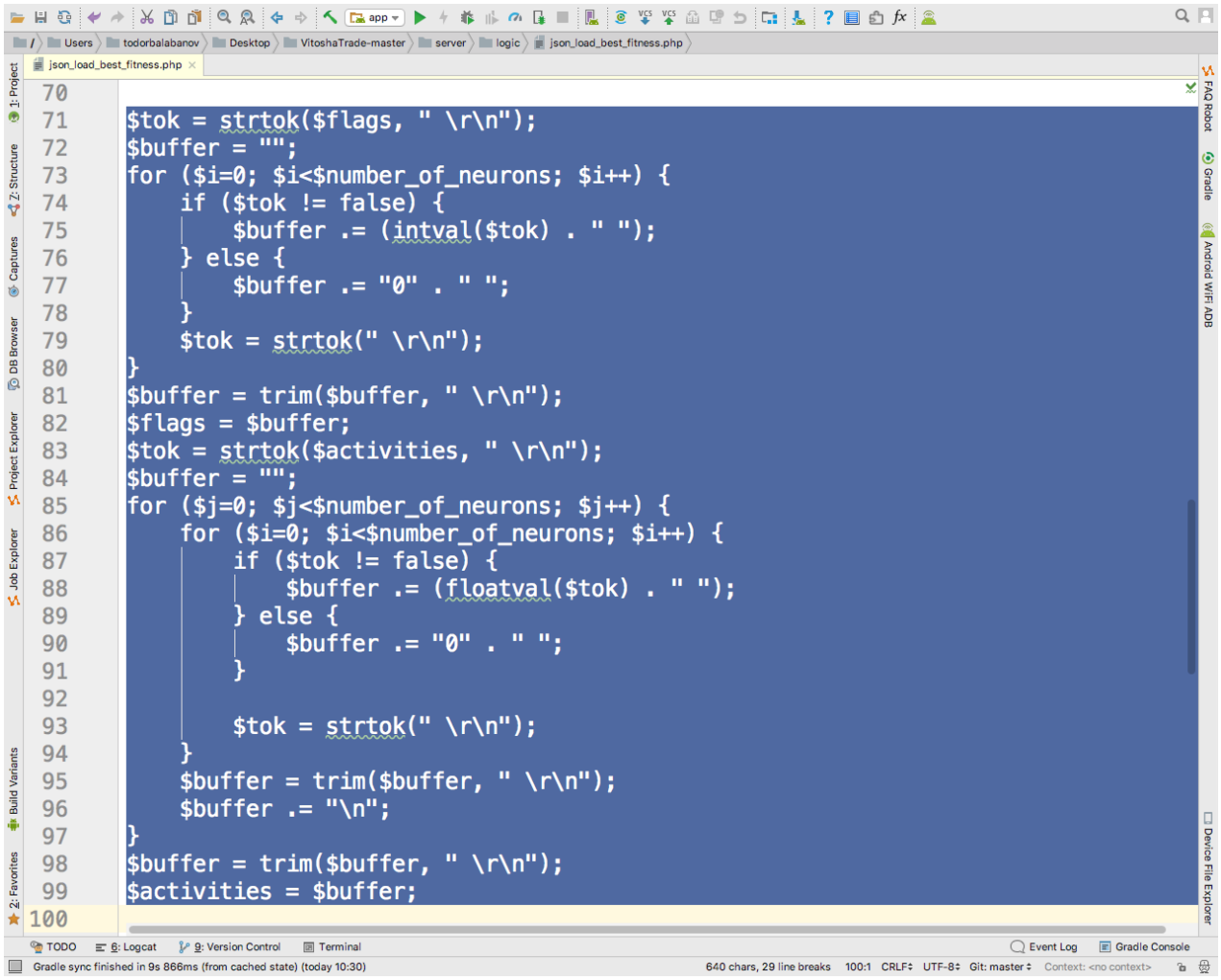
5.3.3 Зареждане на най-добрата жизненост от глобалната популация

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.



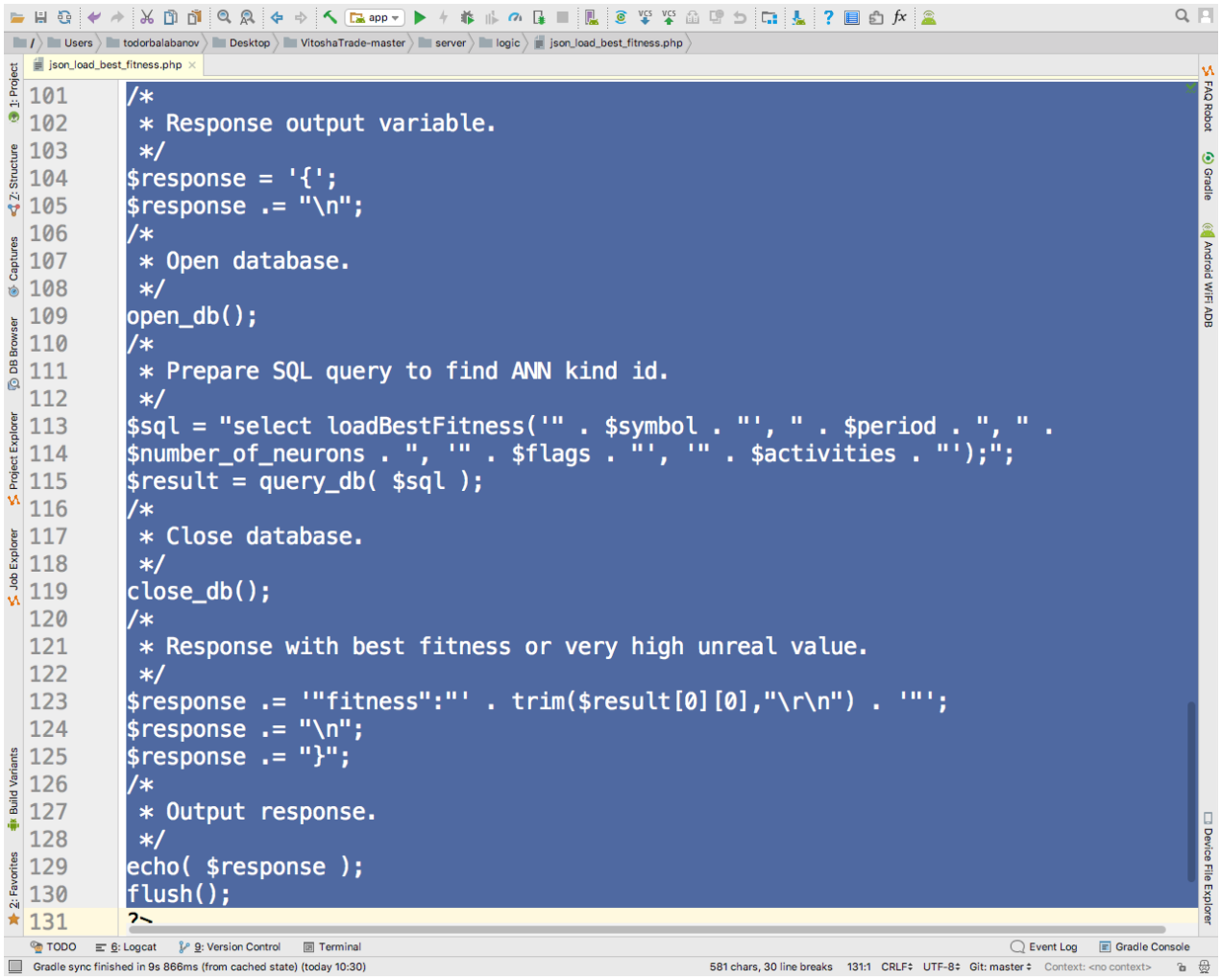
Фигура 5.29: Входни параметри за конкретна топология на невронна мрежа

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



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).

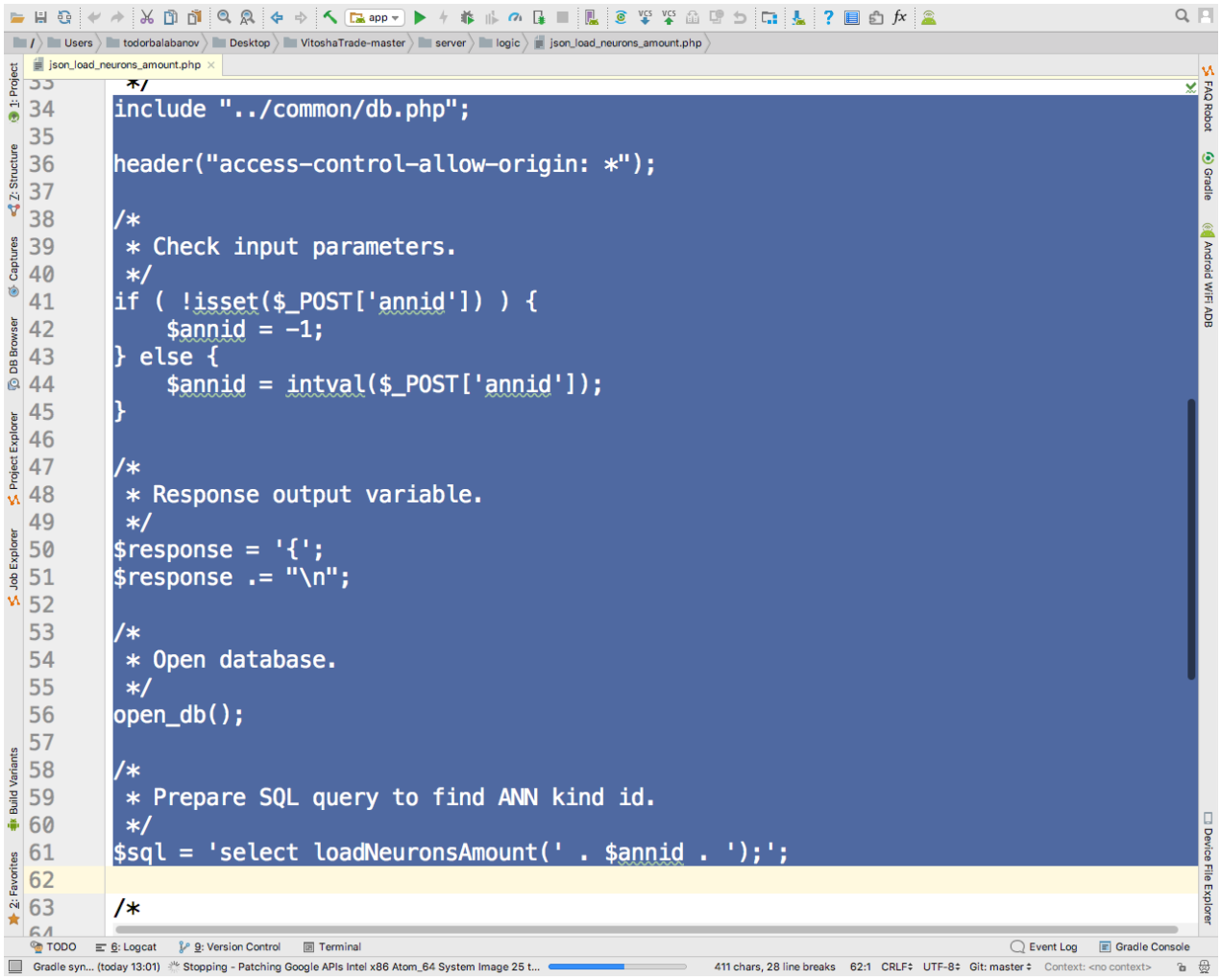


Фигура 5.31: Заявка за най-добра жизненост

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 пакетиран отговор (Фиг. 5.31).

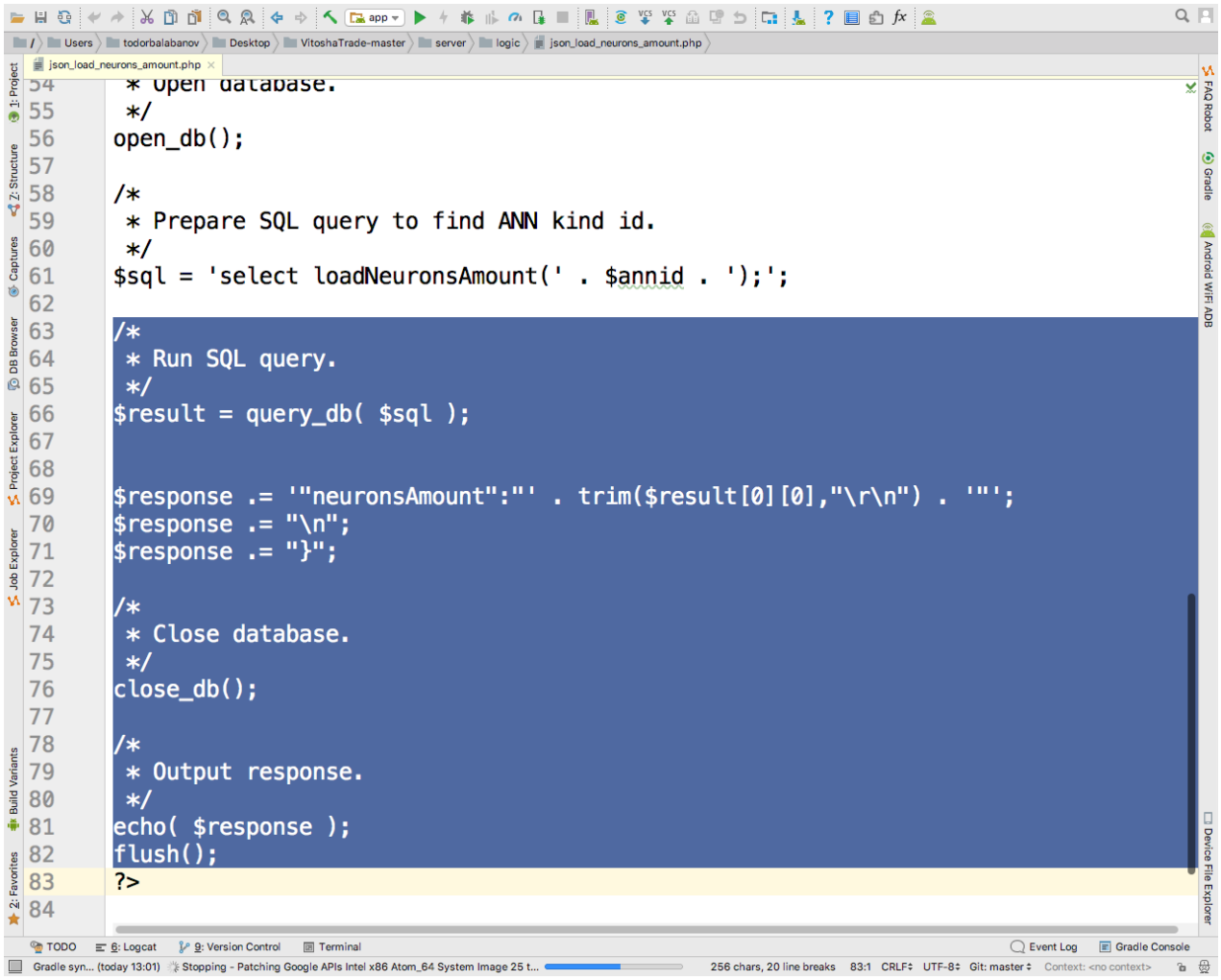
5.3.4 Зареждане на брой неврони по идентификатор

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



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).

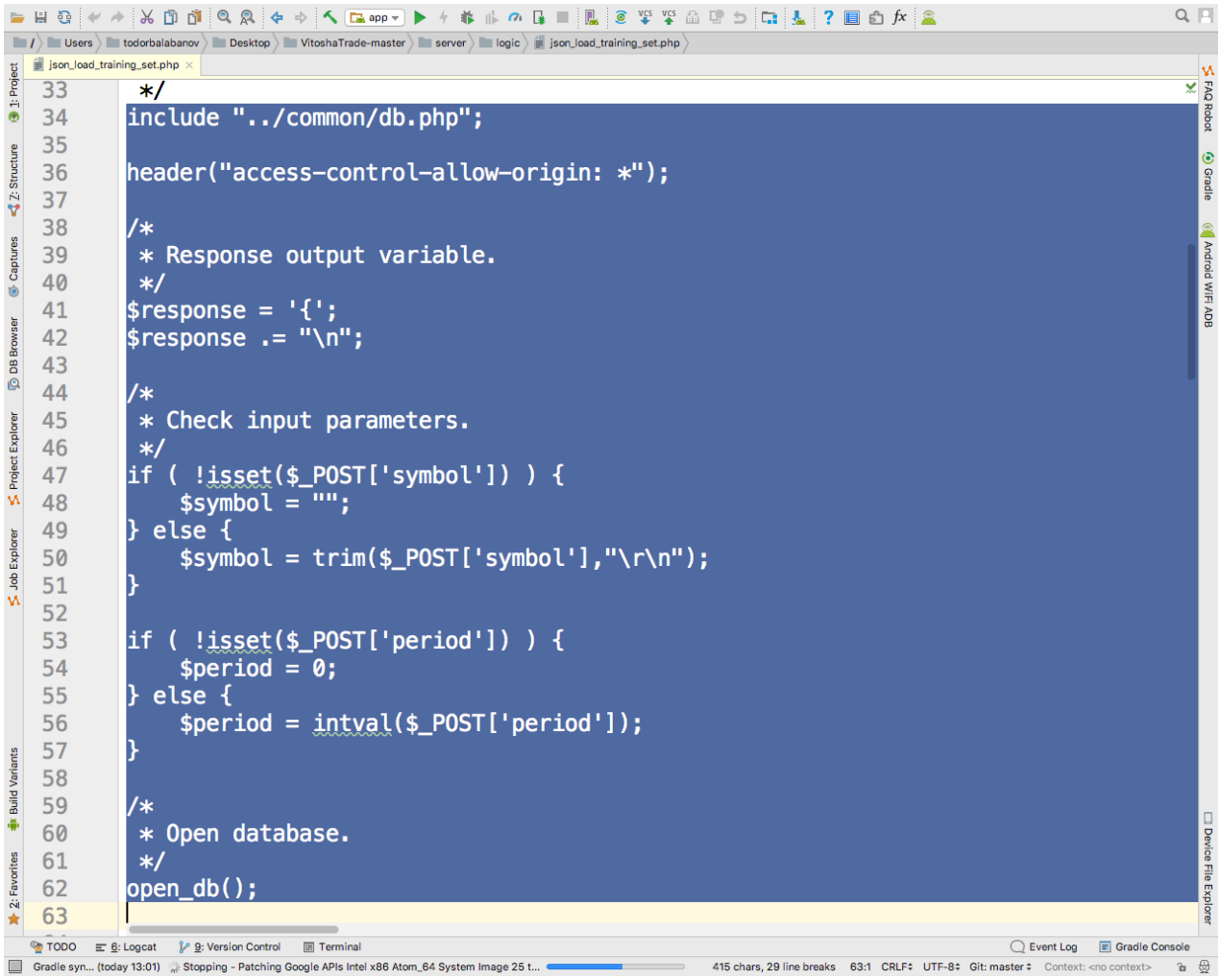


Фигура 5.33: Отговор към клиента

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

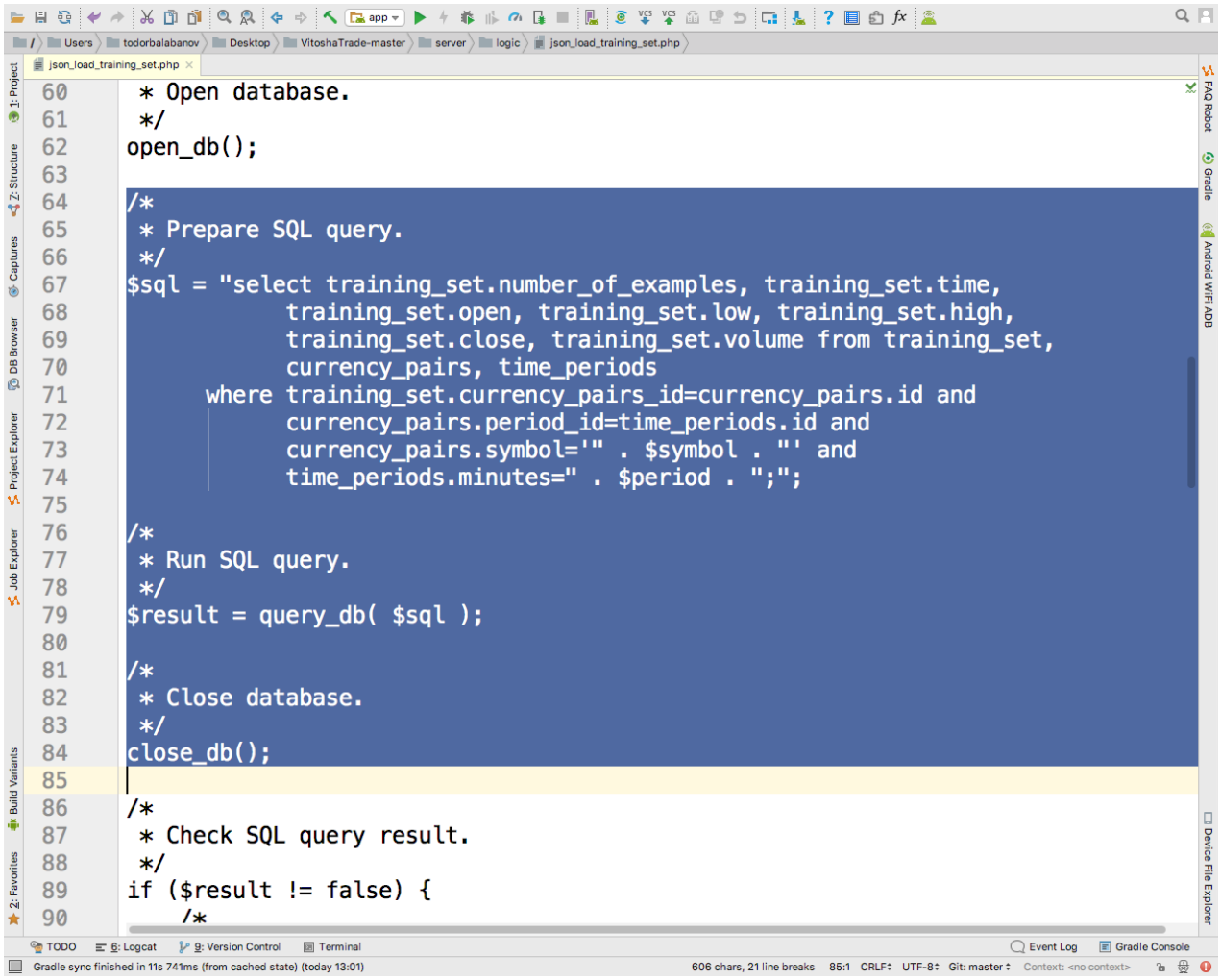
5.3.5 Зареждане на тренировъчно множество по информация за валутна двойка

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



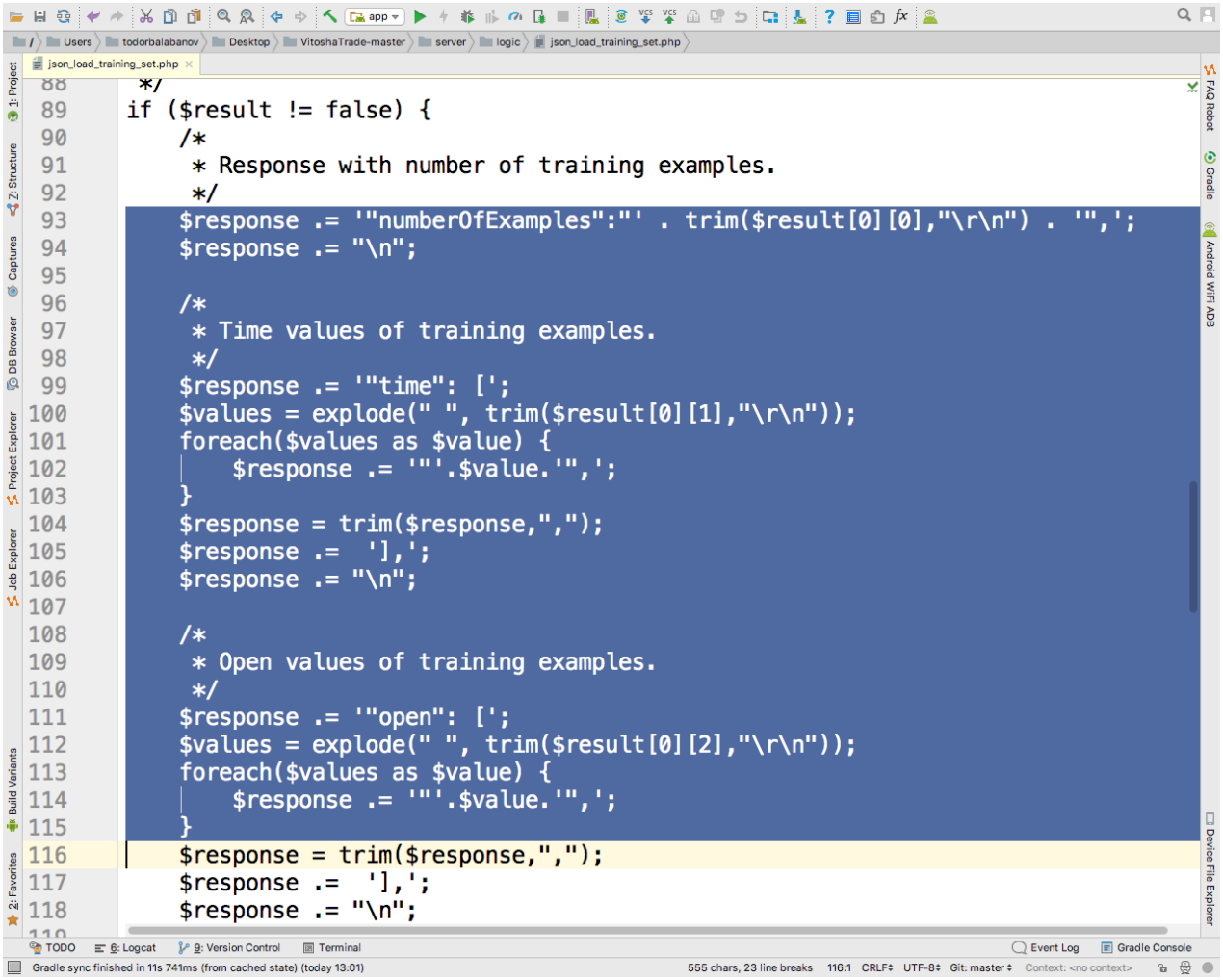
Фигура 5.34: Проверка на входните параметри

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).

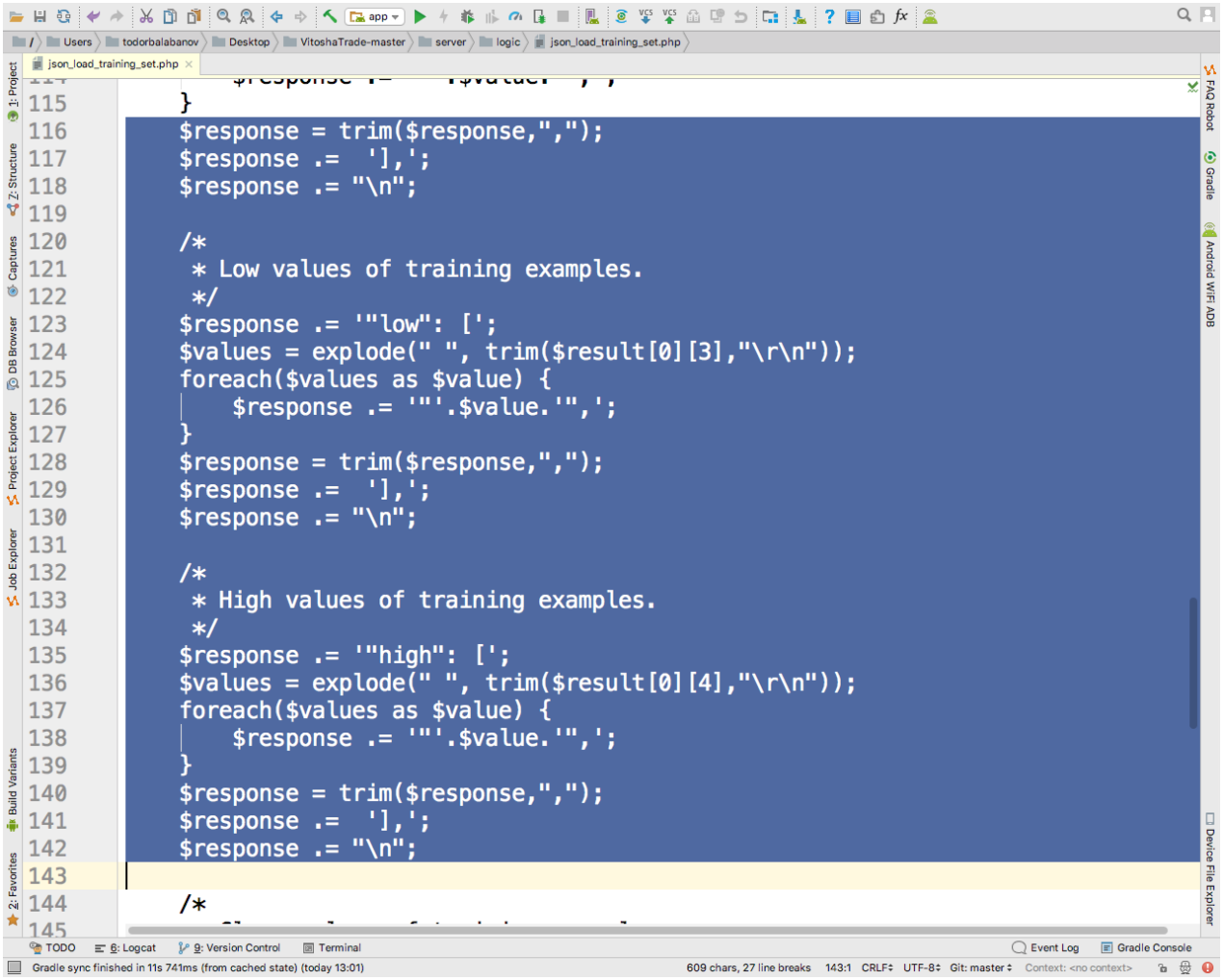


Фигура 5.35: Заявка за тренировъчно множество

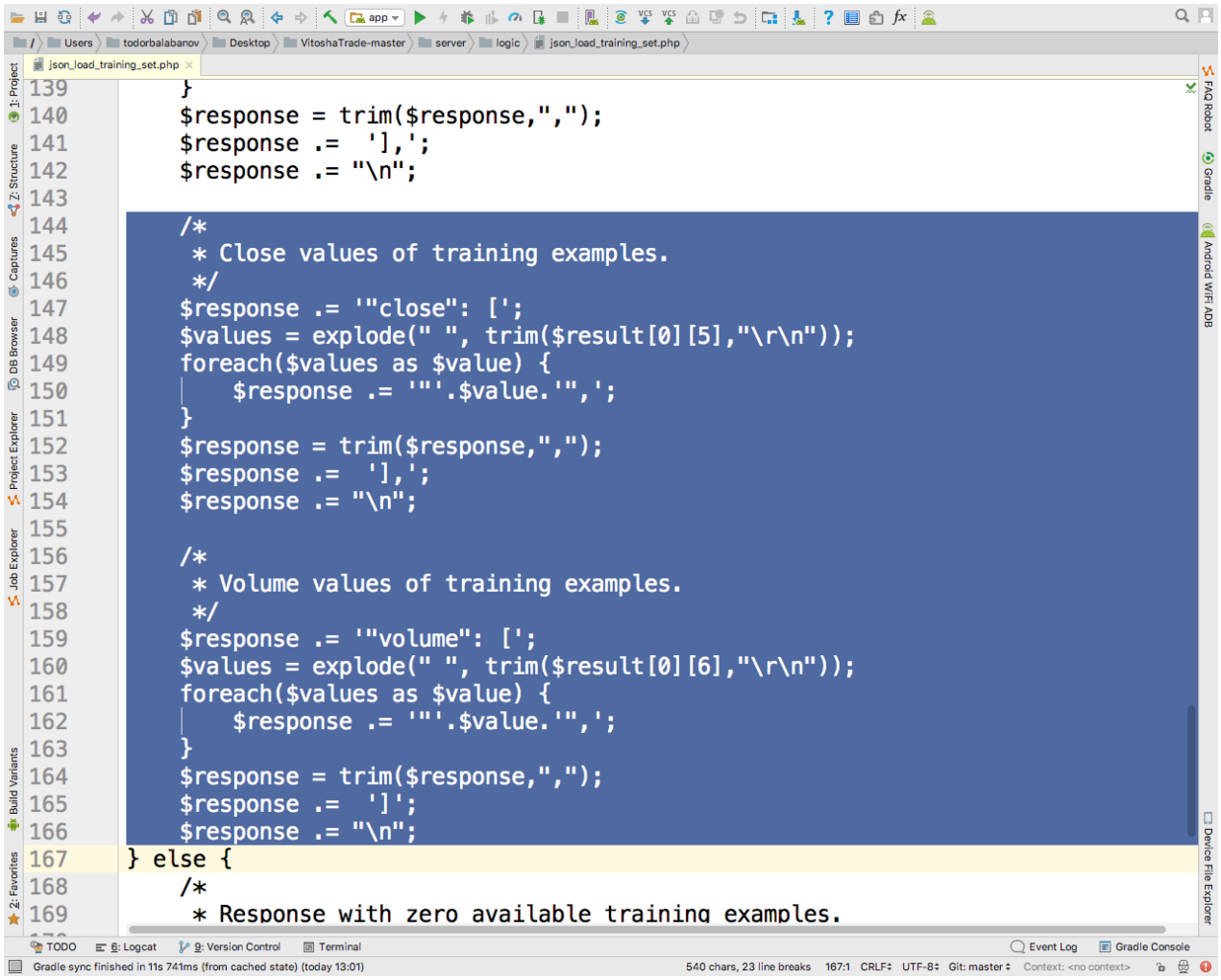
Следва заявка, извличане на резултата и затваряне на връзката към базата данни (Фиг. 5.35). Макар и да не е направено, добрият стил за разделяне на слоевете предполага вместо заявка да се извика съхранена функция.



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



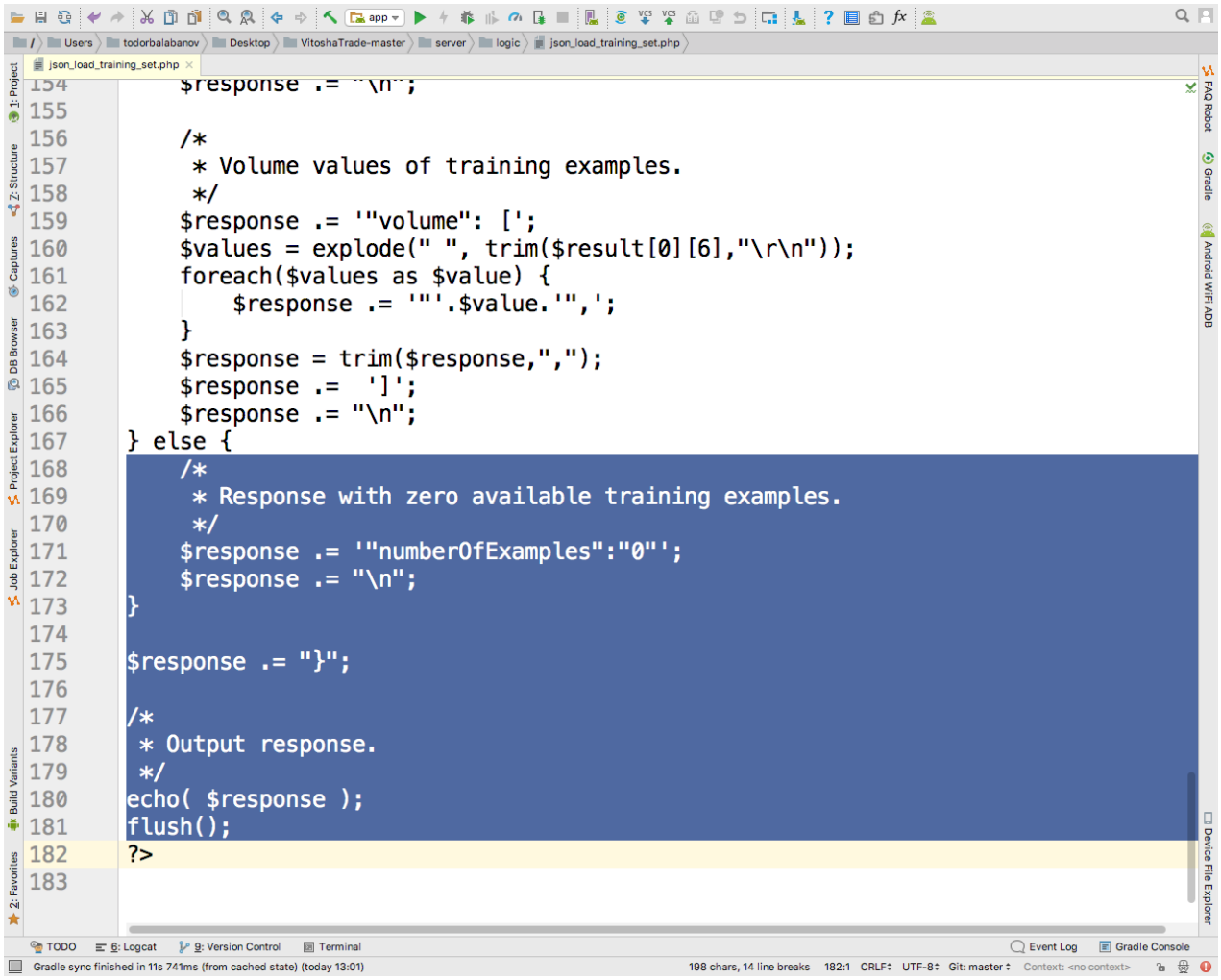
Фигура 5.37: Най-ниска и най-висока постигната стойност




```
    }
    $response = trim($response,",");
    $response .=  '],';
    $response .= "\n";

    /*
     * Close values of training examples.
     */
    $response .= '"close": [';
    $values = explode(" ", trim($result[0][5],"\r\n"));
    foreach($values as $value) {
        $response .= '"'.$value.'",';
    }
    $response = trim($response,",");
    $response .=  '],';
    $response .= "\n";

    /*
     * Volume values of training examples.
     */
    $response .= '"volume": [';
    $values = explode(" ", trim($result[0][6],"\r\n"));
    foreach($values as $value) {
        $response .= '"'.$value.'",';
    }
    $response = trim($response,",");
    $response .=  ']';
    $response .= "\n";
} else {
    /*
     * 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).

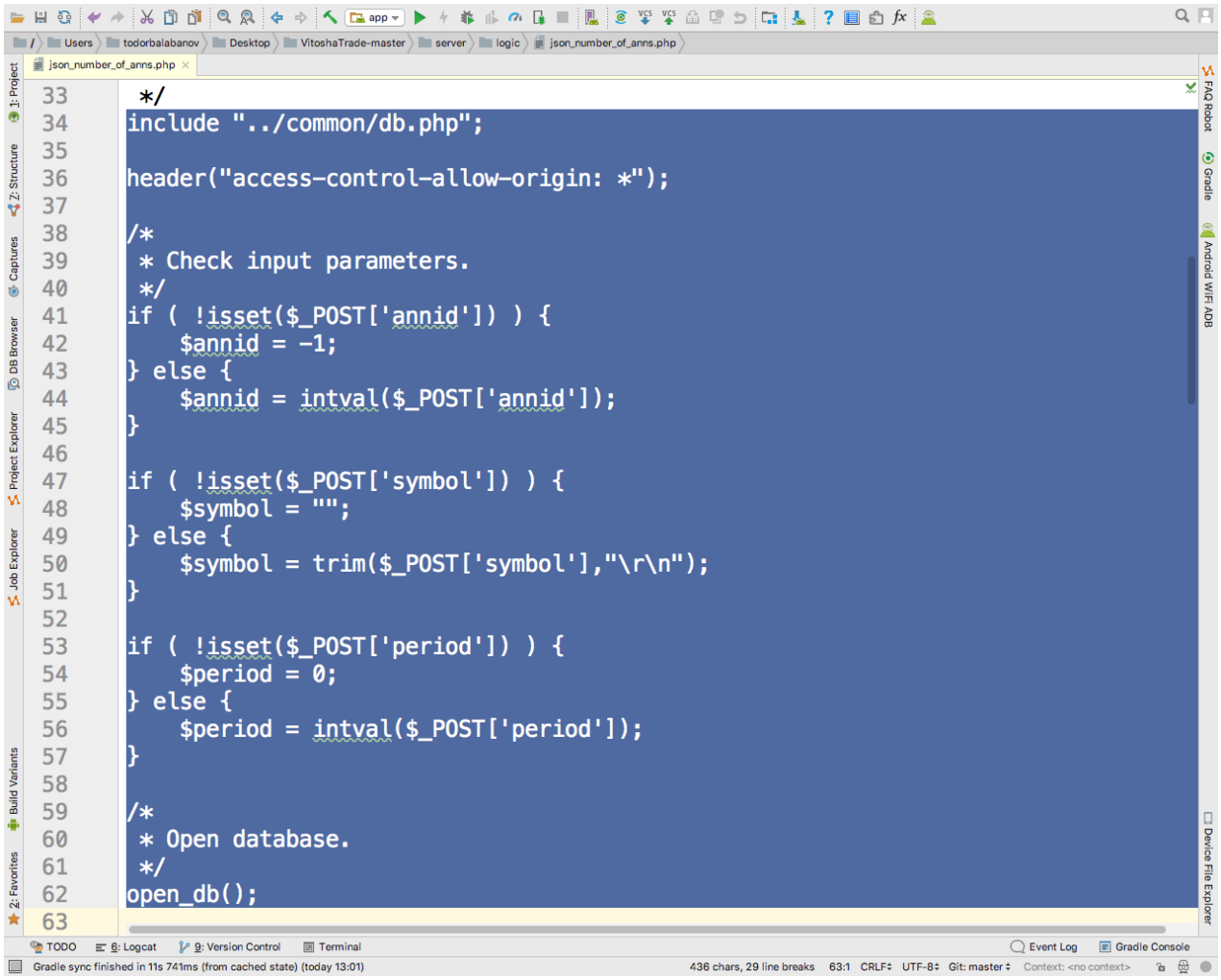


Фигура 5.39: Изпращане на отговор до клиента

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).

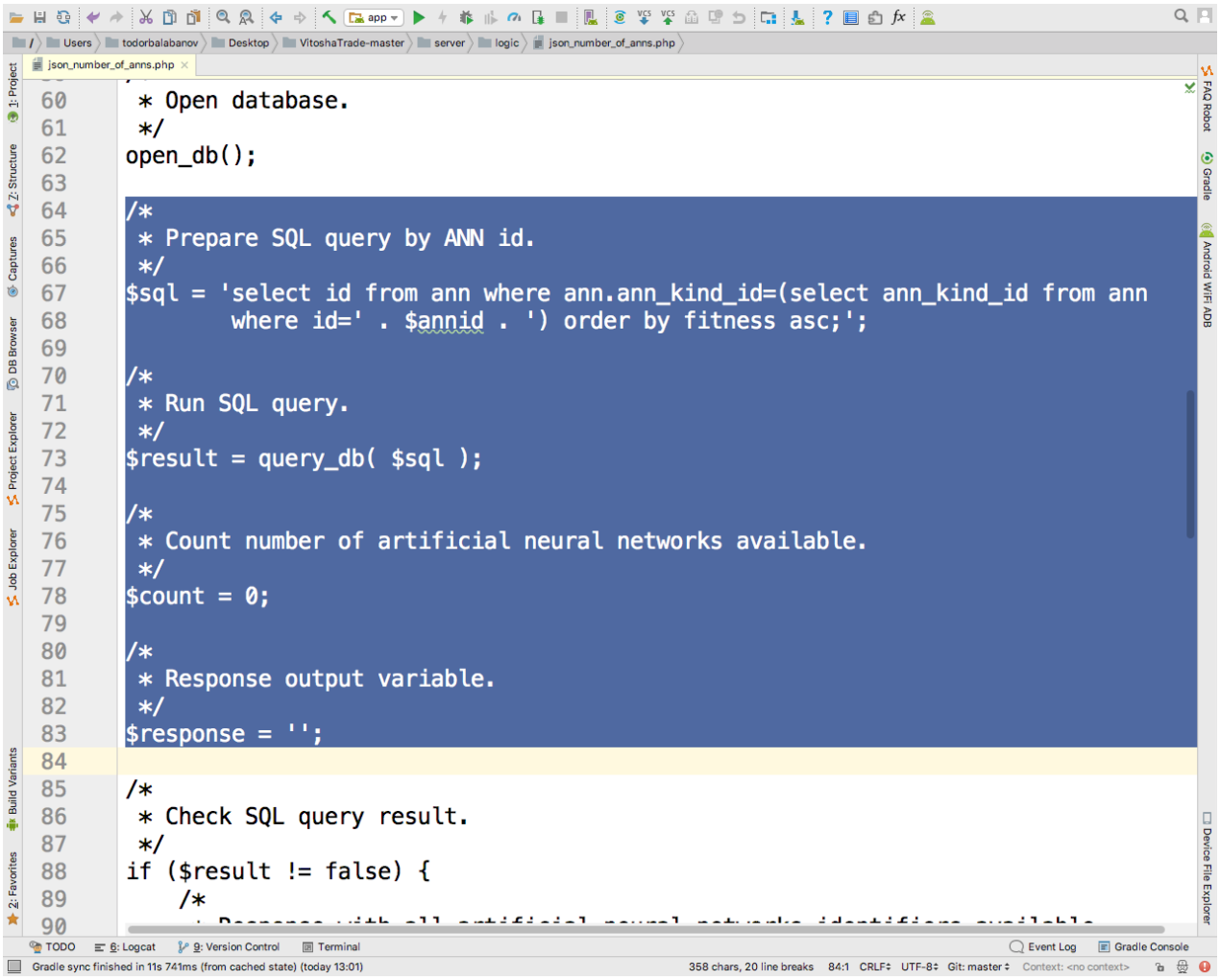
5.3.6 Зареждане на брой екземпляри по идентификатор или информация за валутна двойка

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.



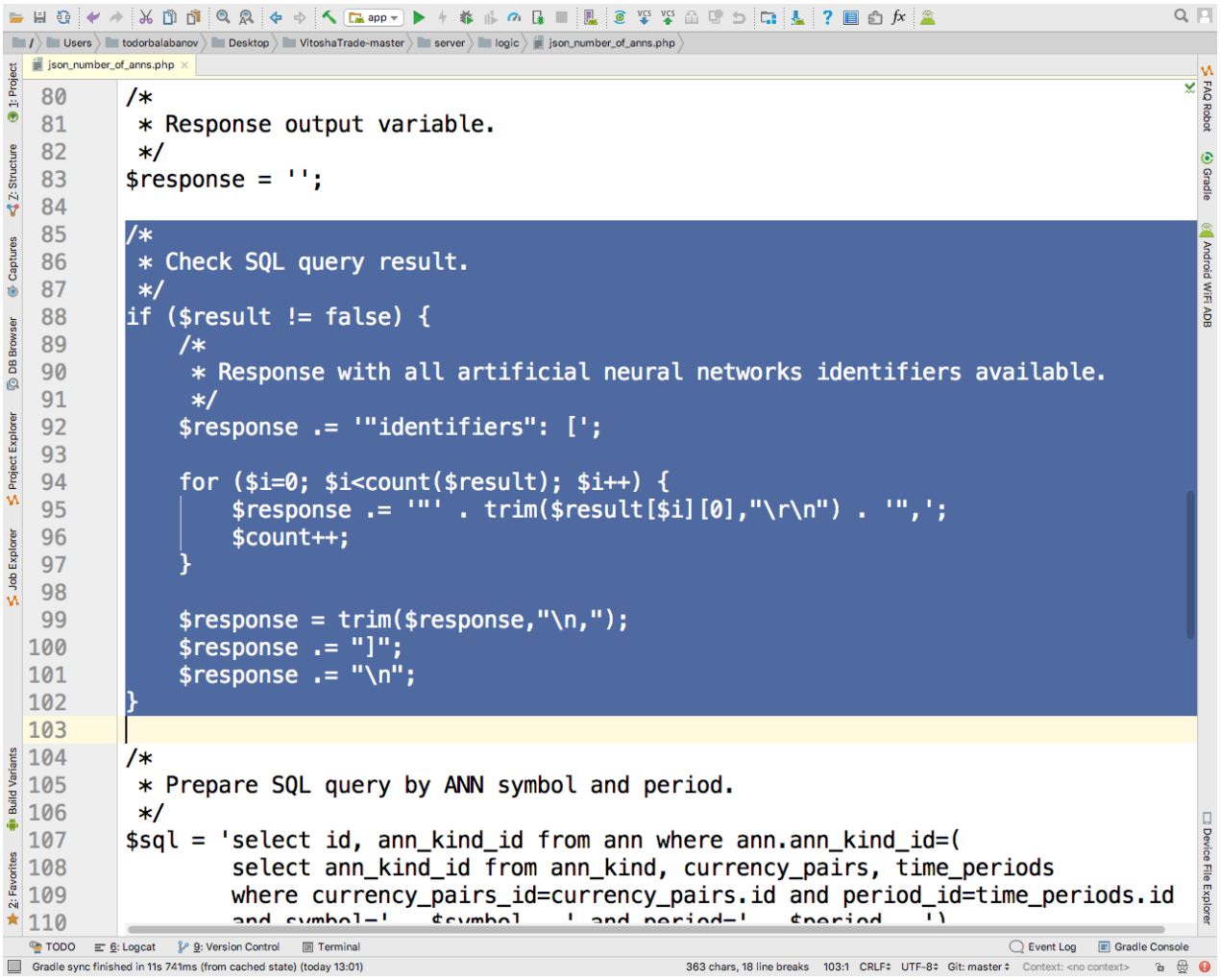
Фигура 5.40: Проверка на входните аргументи

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).



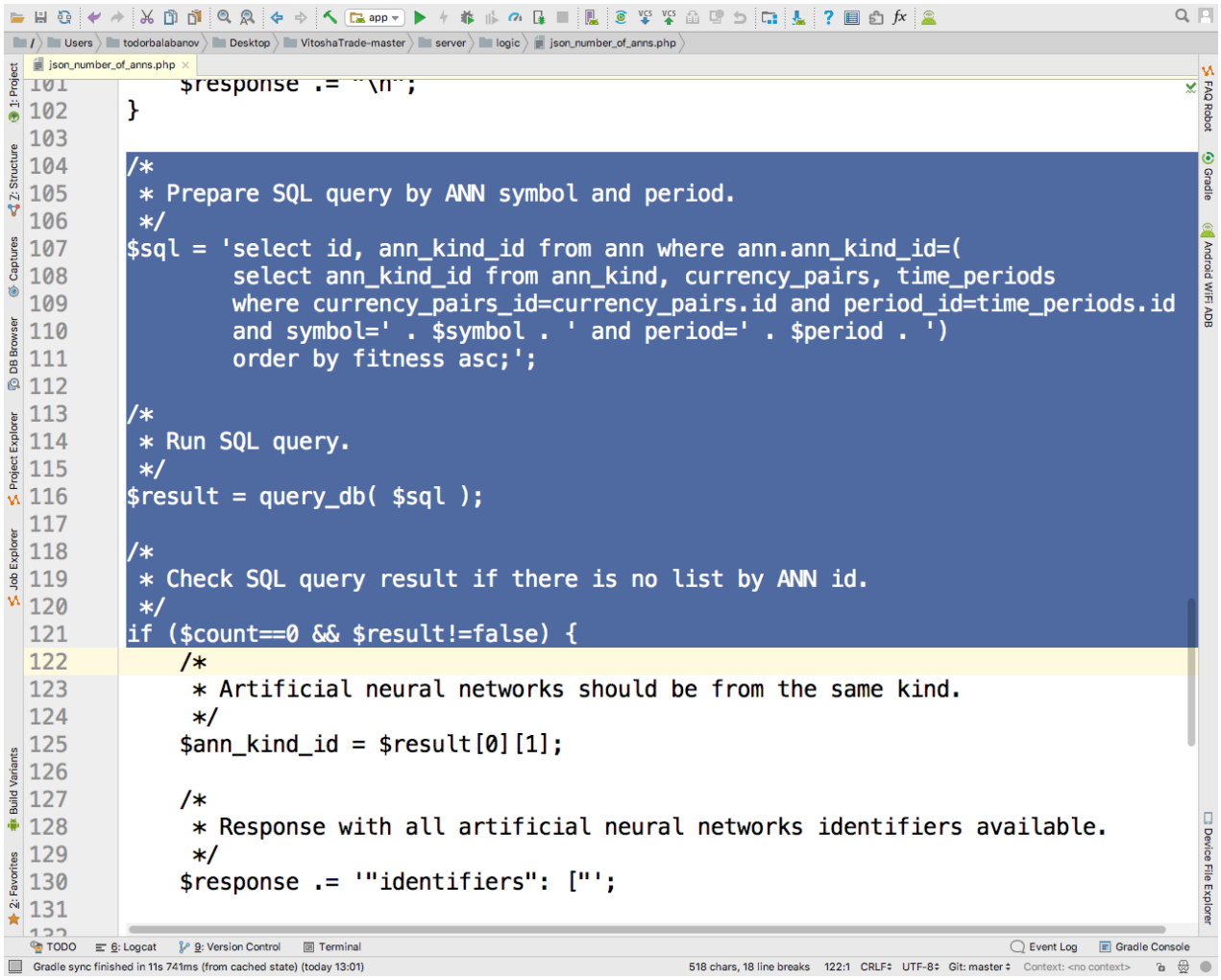
Фигура 5.41: Еземпляри от типа на подадения идентификатор

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



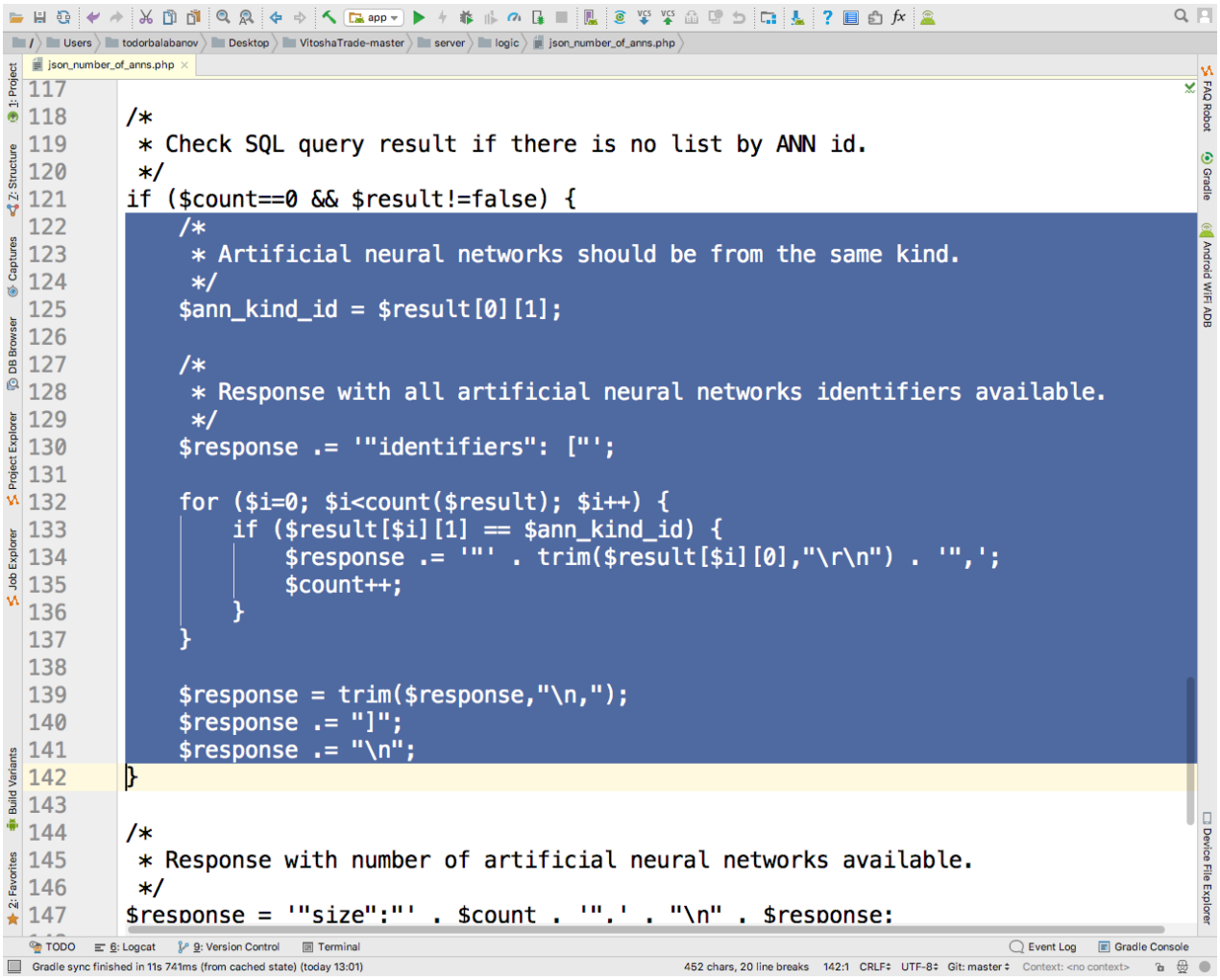
Фигура 5.42: Пакетиране на получените стойности в JSON съобщение

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



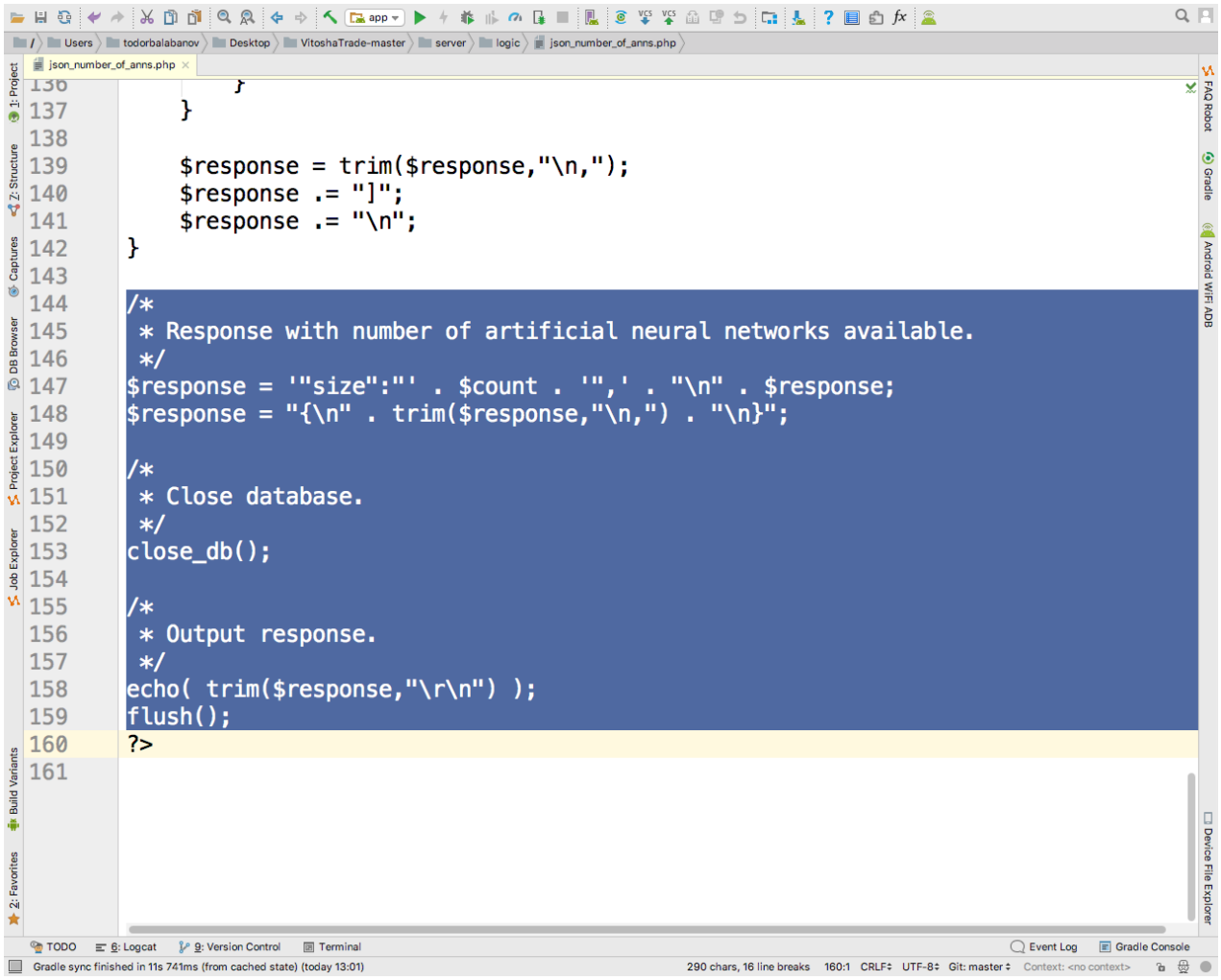
Фигура 5.43: Проверка за екземпляри по название на валутна двойка и период

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).



Фигура 5.44: Пакетиране на получените стойности в JSON съобщение

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

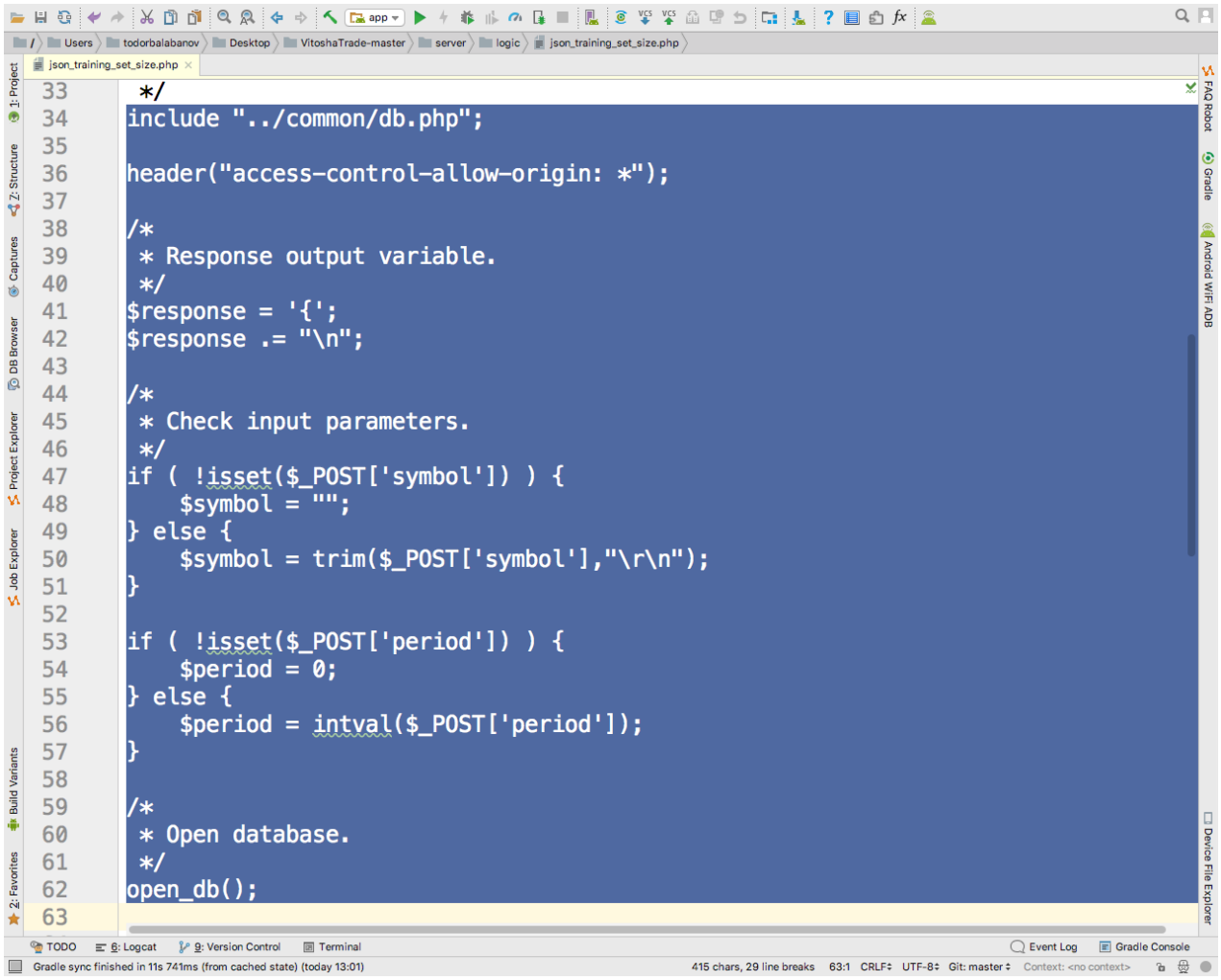


Фигура 5.45: Изпращане на резултата

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

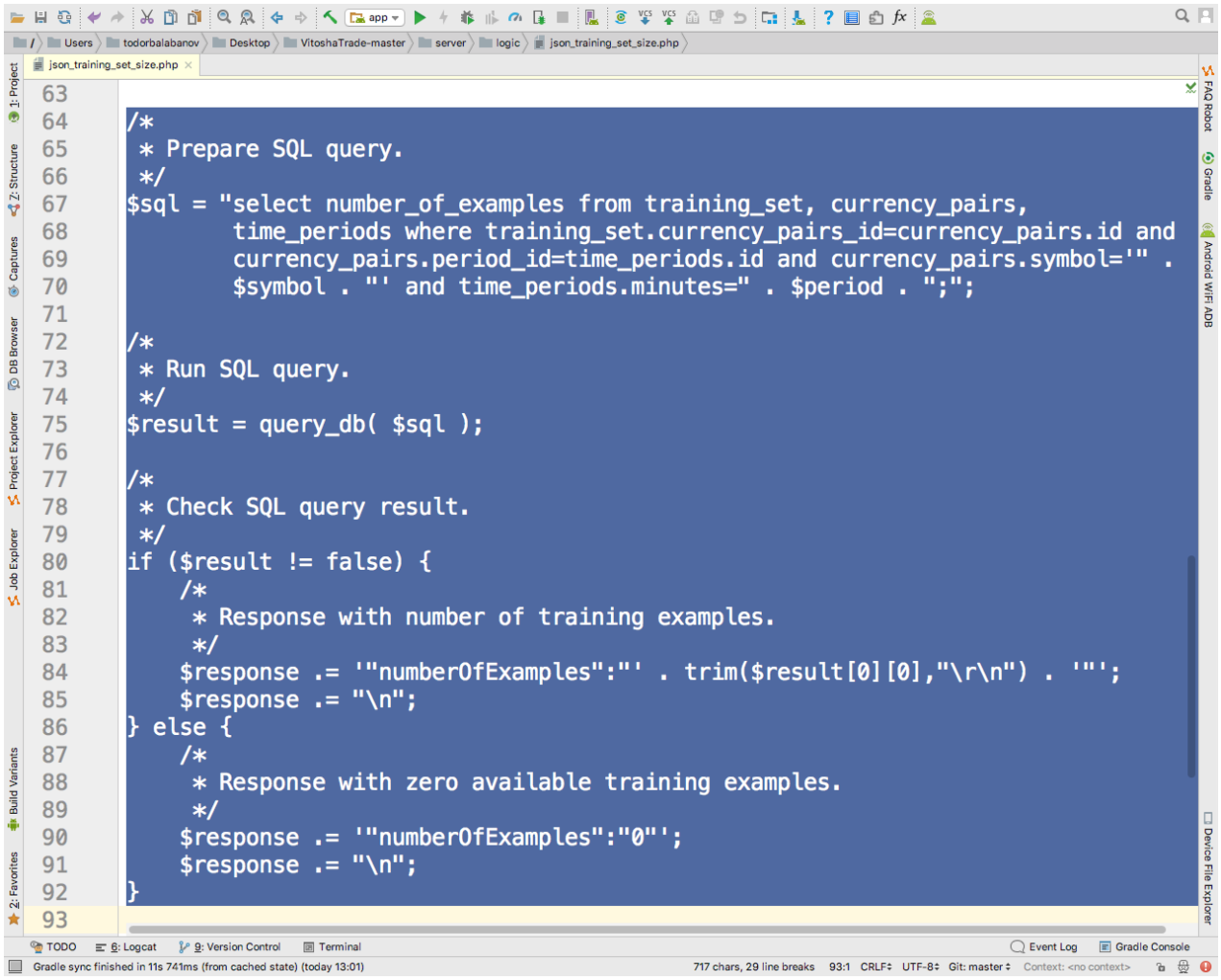
5.3.7 Зареждане на брой тренировъчни примери по информация за валутна двойка

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).



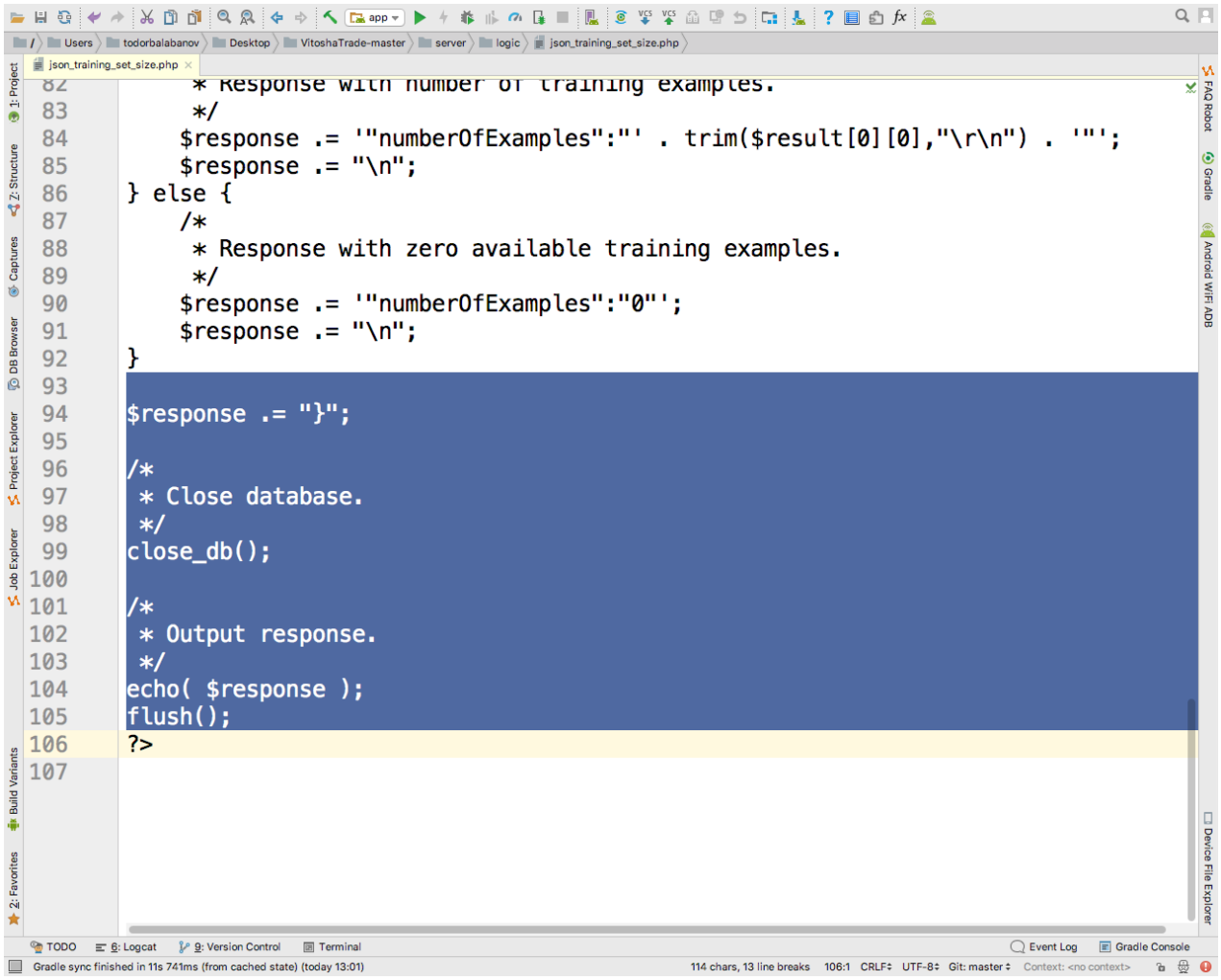
Фигура 5.46: Брой тренировъчни примери по валутна информация

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



Фигура 5.47: Заявка за брой тренировъчни примери

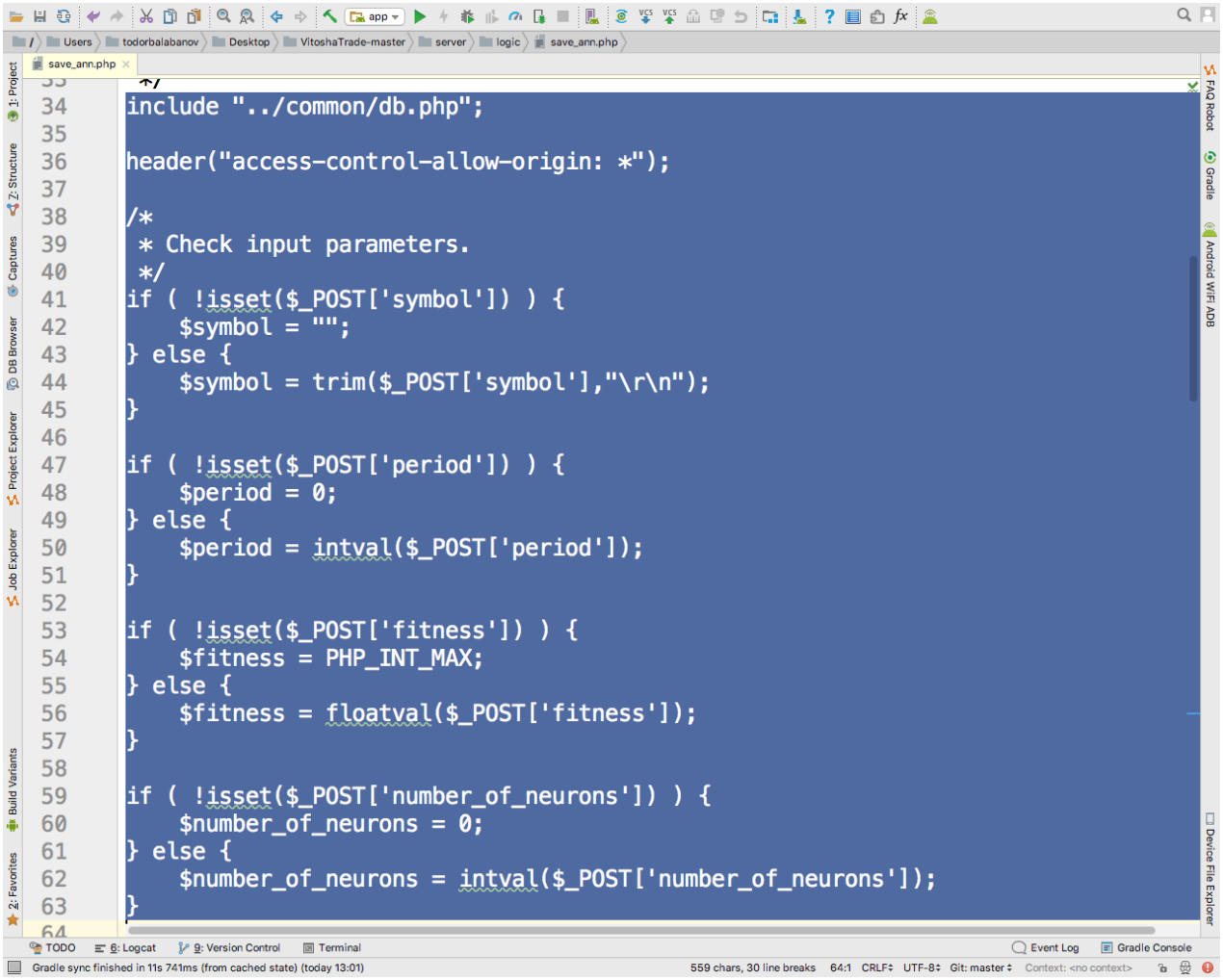
Скриптът приключва със затваряне на връзката към базата данни, пакетиране и изпращане на отговора до клиента (Фиг. 5.48).



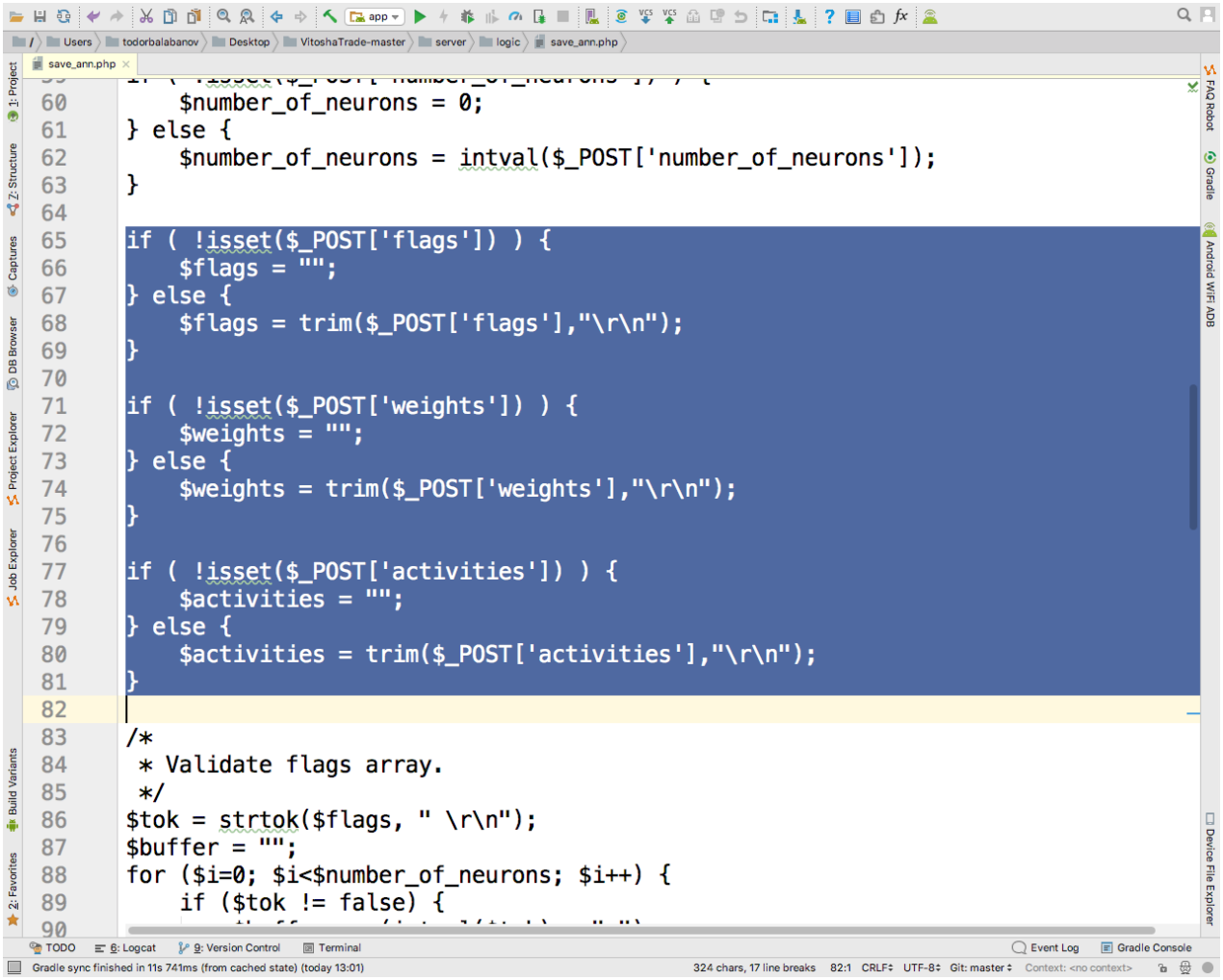
Фигура 5.48: Изпращане на отговор до клиента

5.3.8 Съхраняване на екземпляр изкуствена невронна мрежа

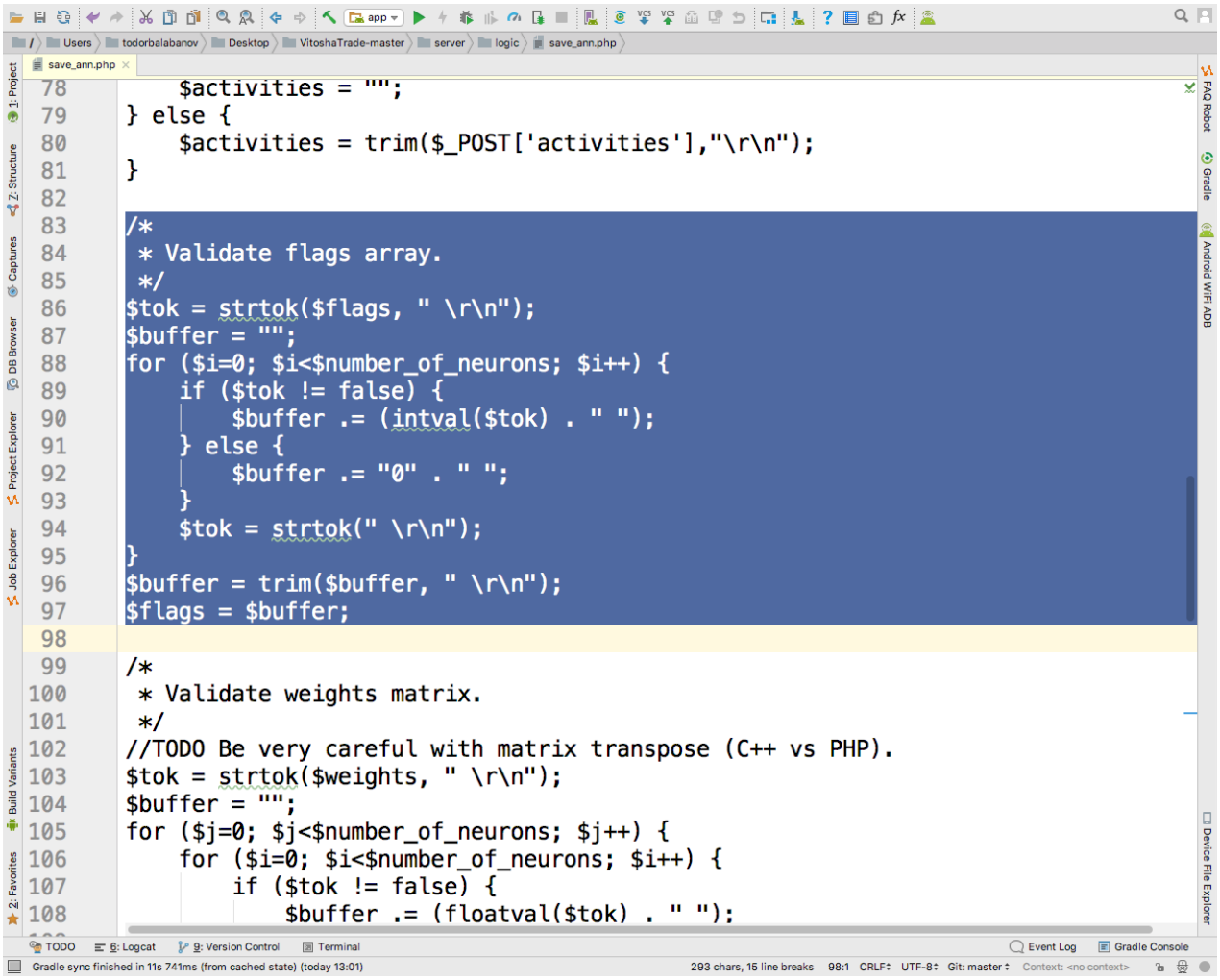
Съхраняването на екземпляр от изкуствената невронна мрежа не връща резултат за изпълнението на операцията и поради тази причина не се използва JSON. След включването на модула за работа с базата от данни се изисква по-сложна проверка на входните аргументи, тъй като те са повече на брой и по-сложни (Фиг.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



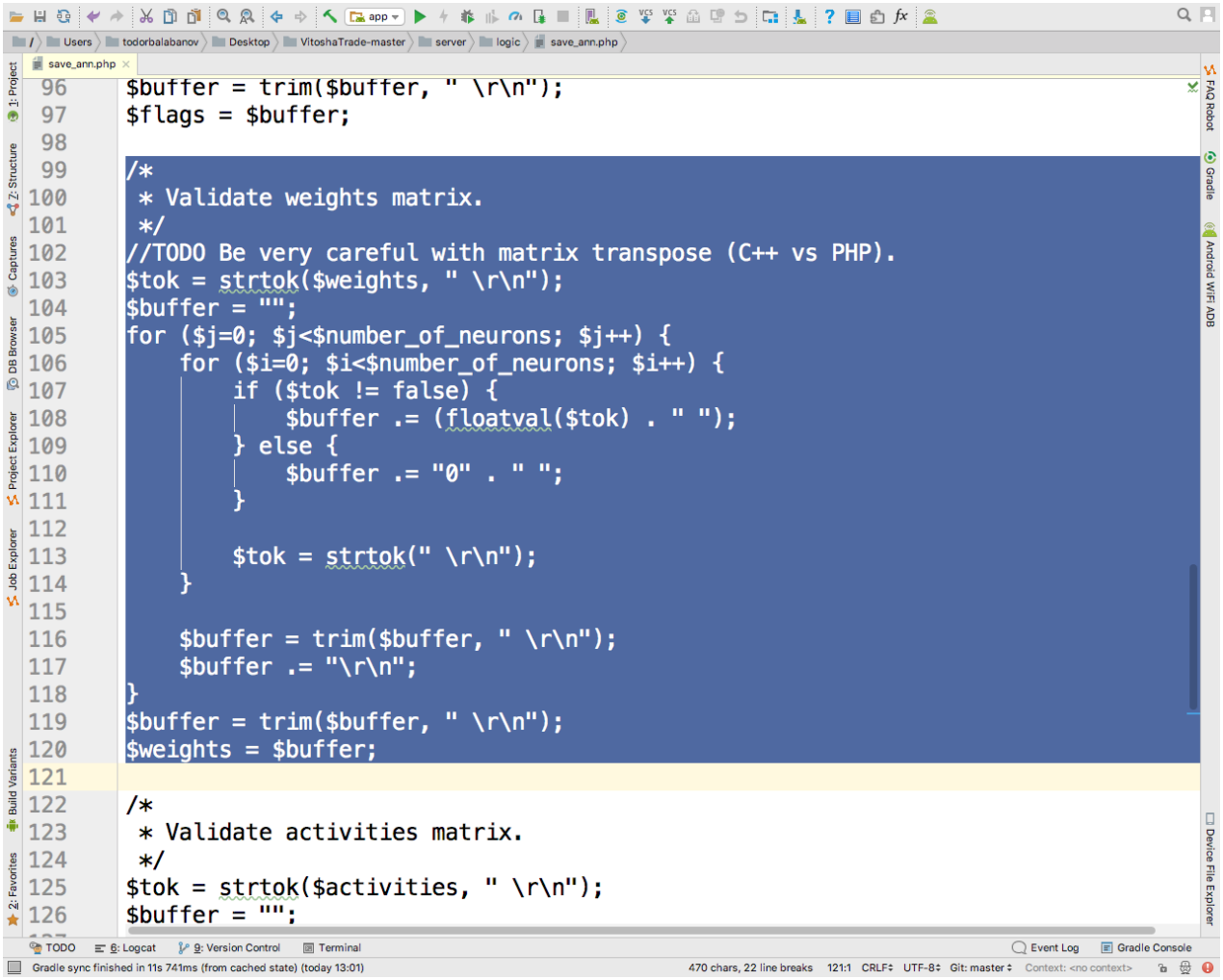
Фигура 5.49: Проверка на входните аргументи за валутна двойка, период, жизненост и брой неврони



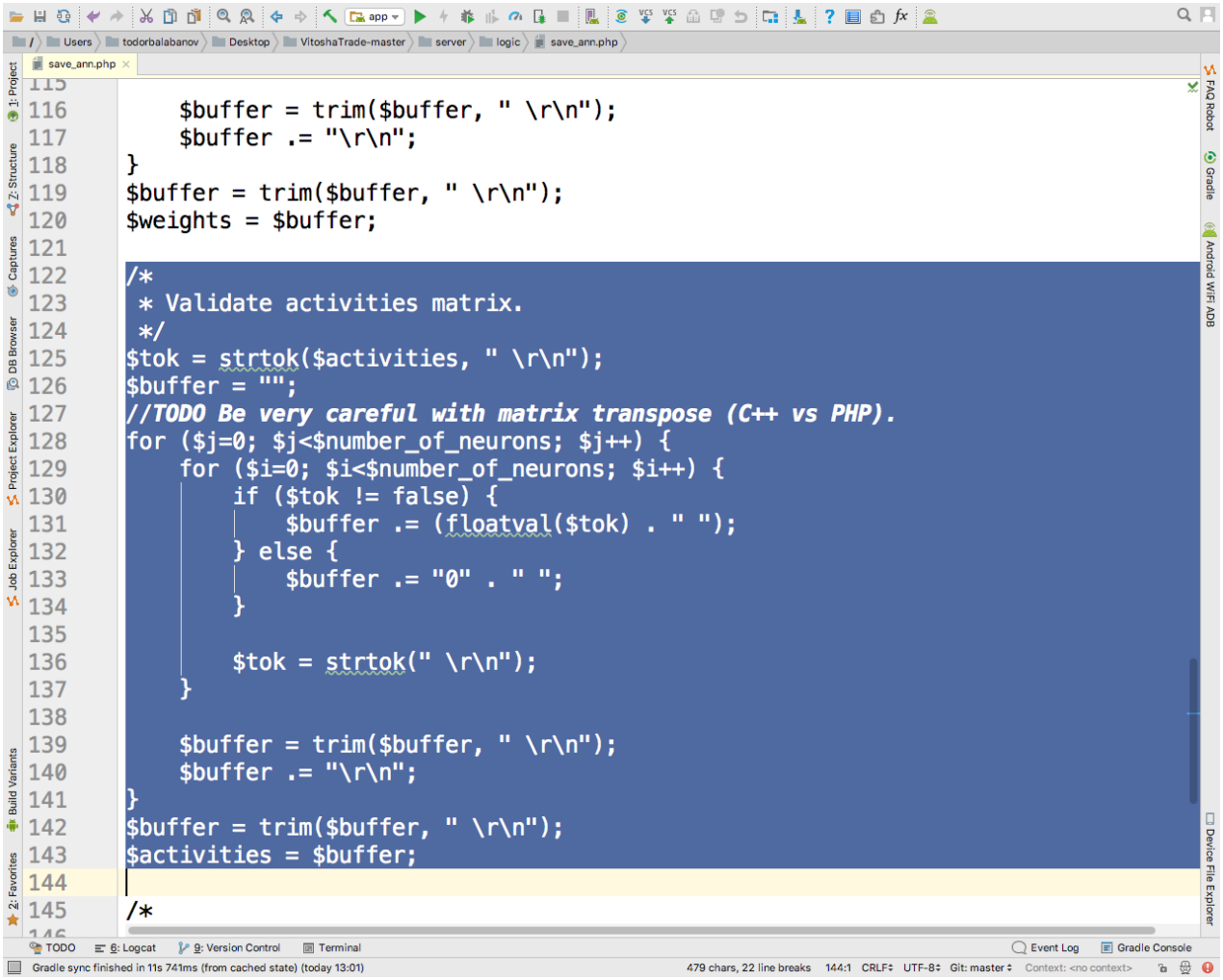
Фигура 5.50: Проверка на входните аргументи за флагове на невроните, връзки и тегла между невроните



Фигура 5.51: Проверка за типа на всеки неврон

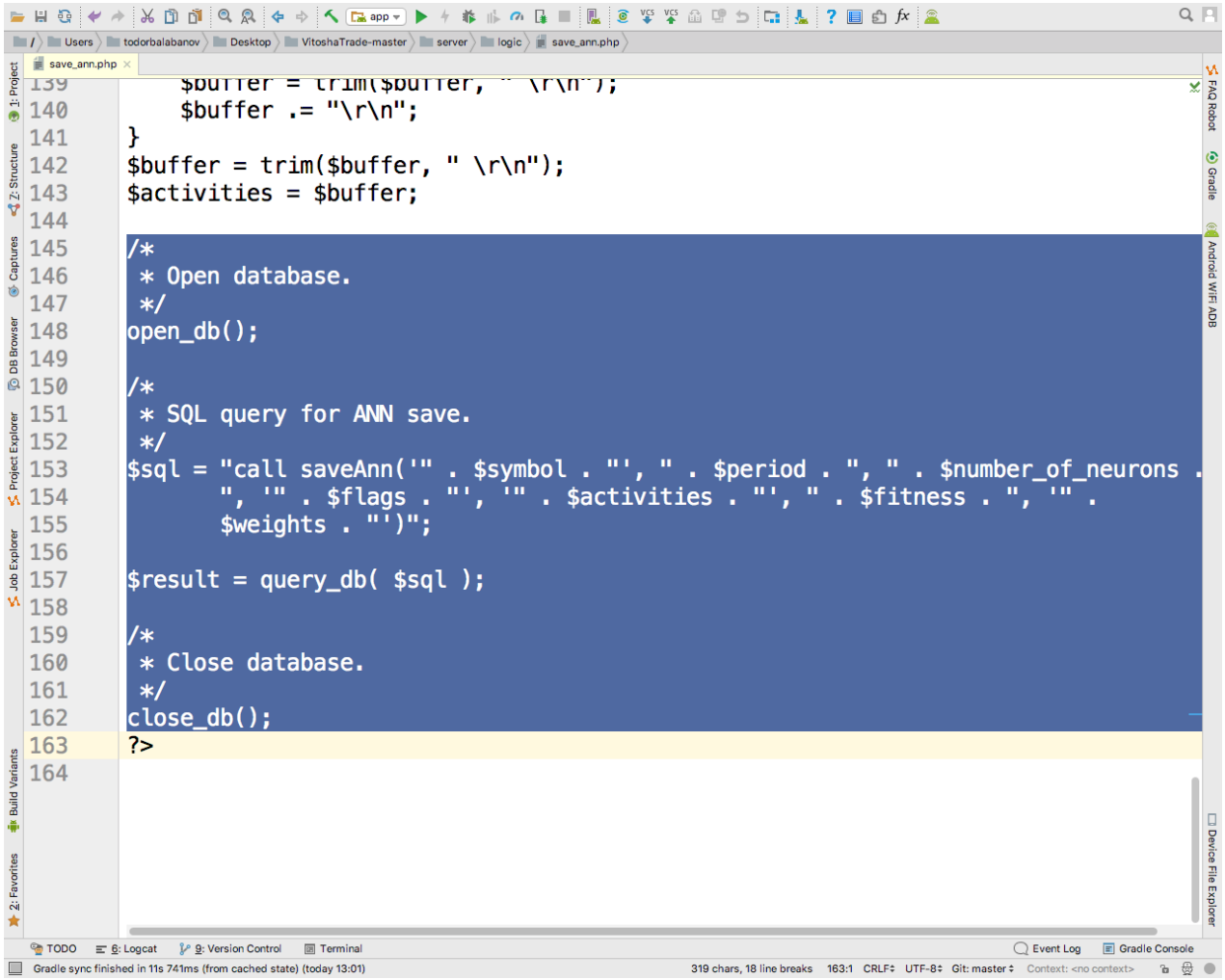


Фигура 5.52: Проверка за стойността на всяко тегло



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.

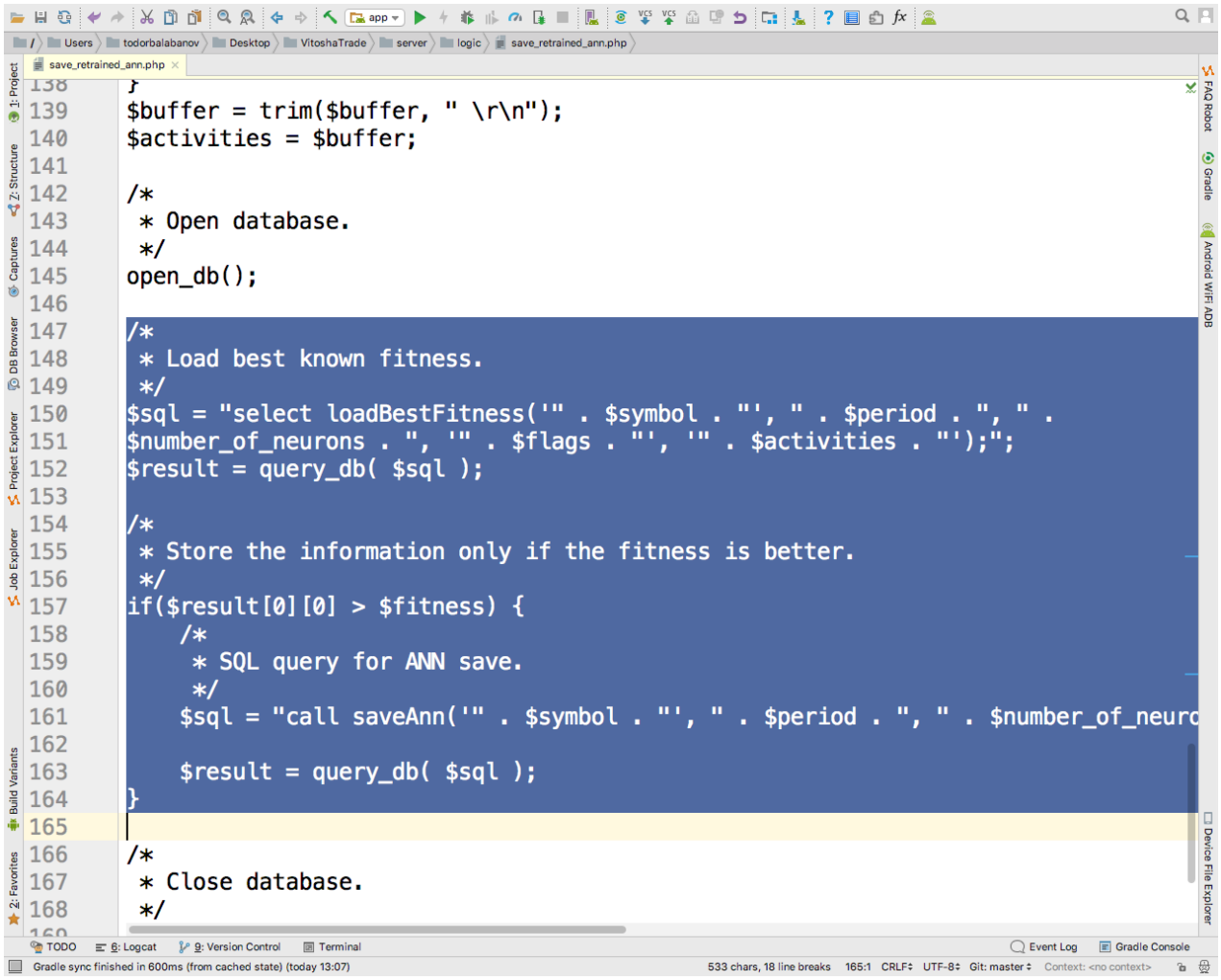


Фигура 5.54: Съхраняване на екземпляр изкуствена невронна мрежа

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.

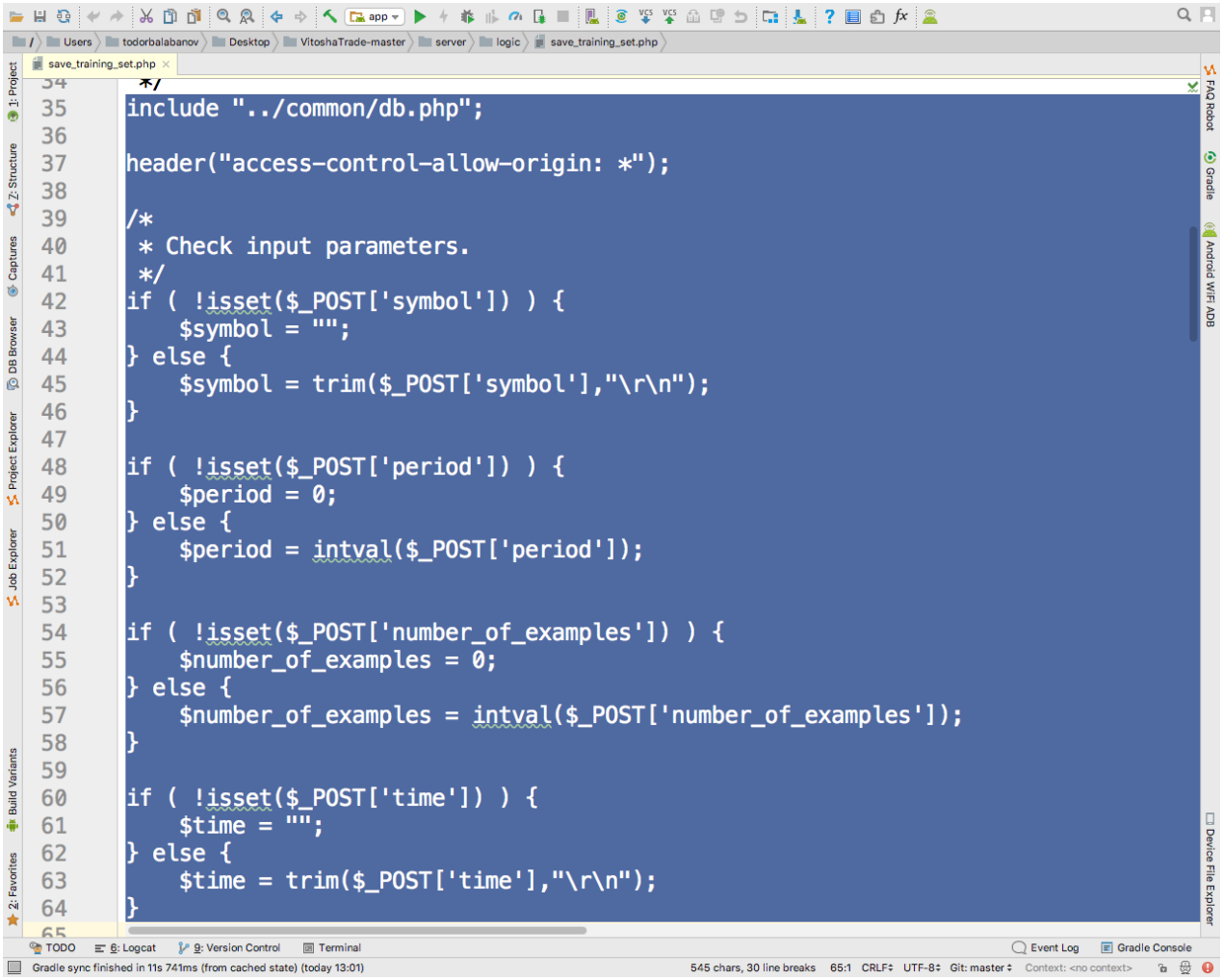


Фигура 5.55: Проверка на жизнеността преди съхраняване на информацията

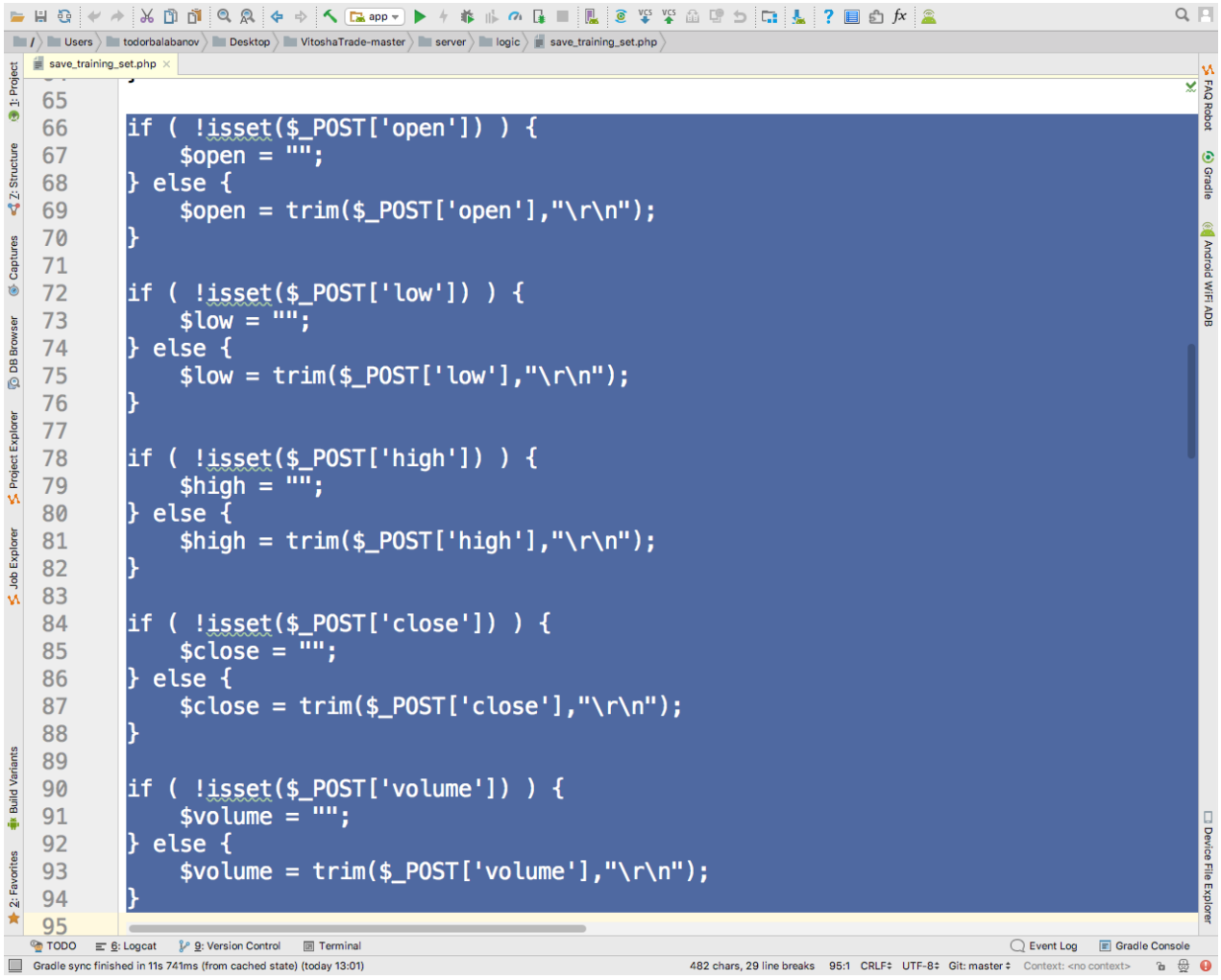
Разликата между общата процедура за съхраняване на изкуствена невронна мрежа и съхраняването след допълнително обучение е в проверката на най-добрата постигната жизнена стойност (Фиг. 5.55).

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).

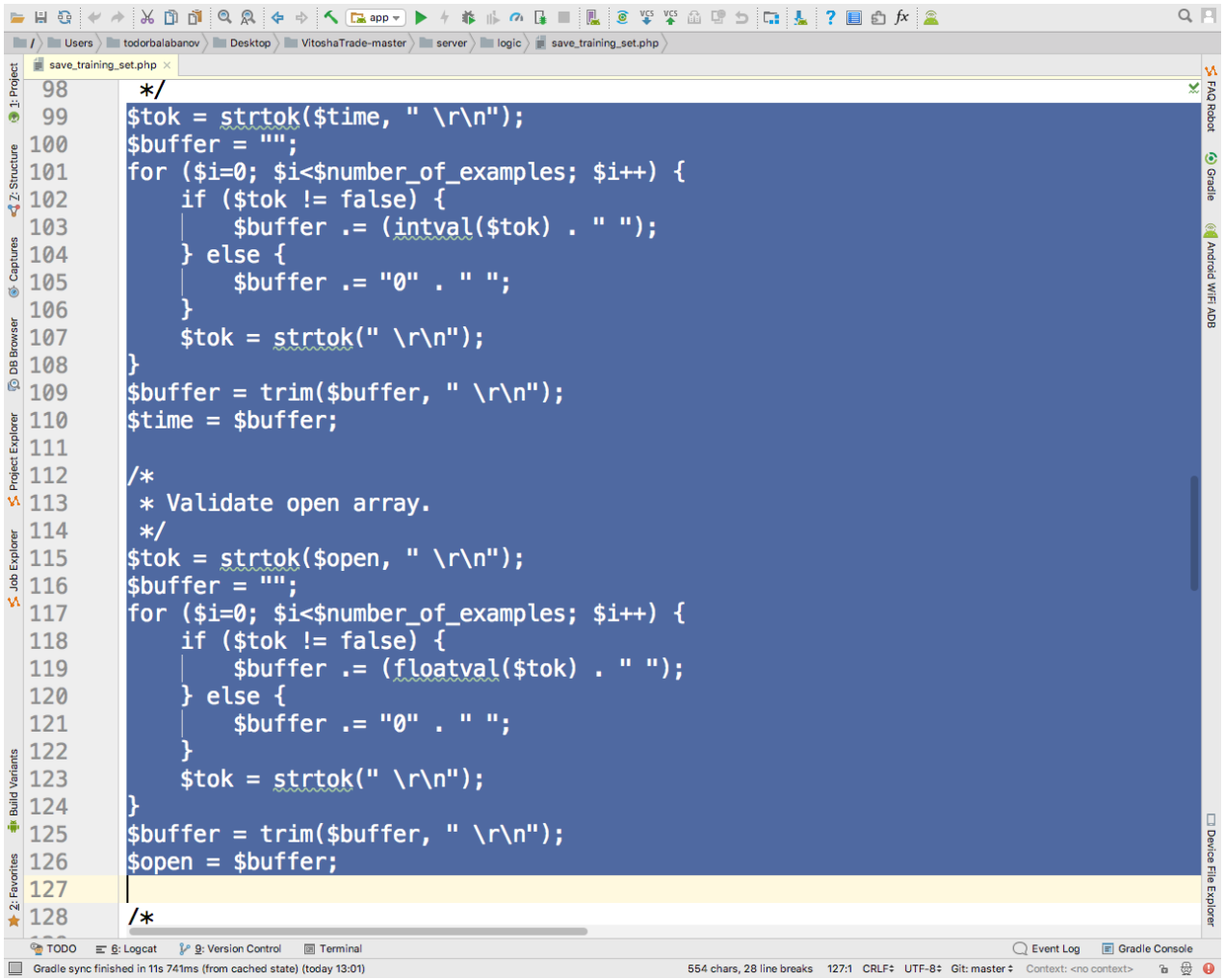


Фигура 5.56: Проверка на агументите за валутна двойка, период, брой примери и времеви маркери

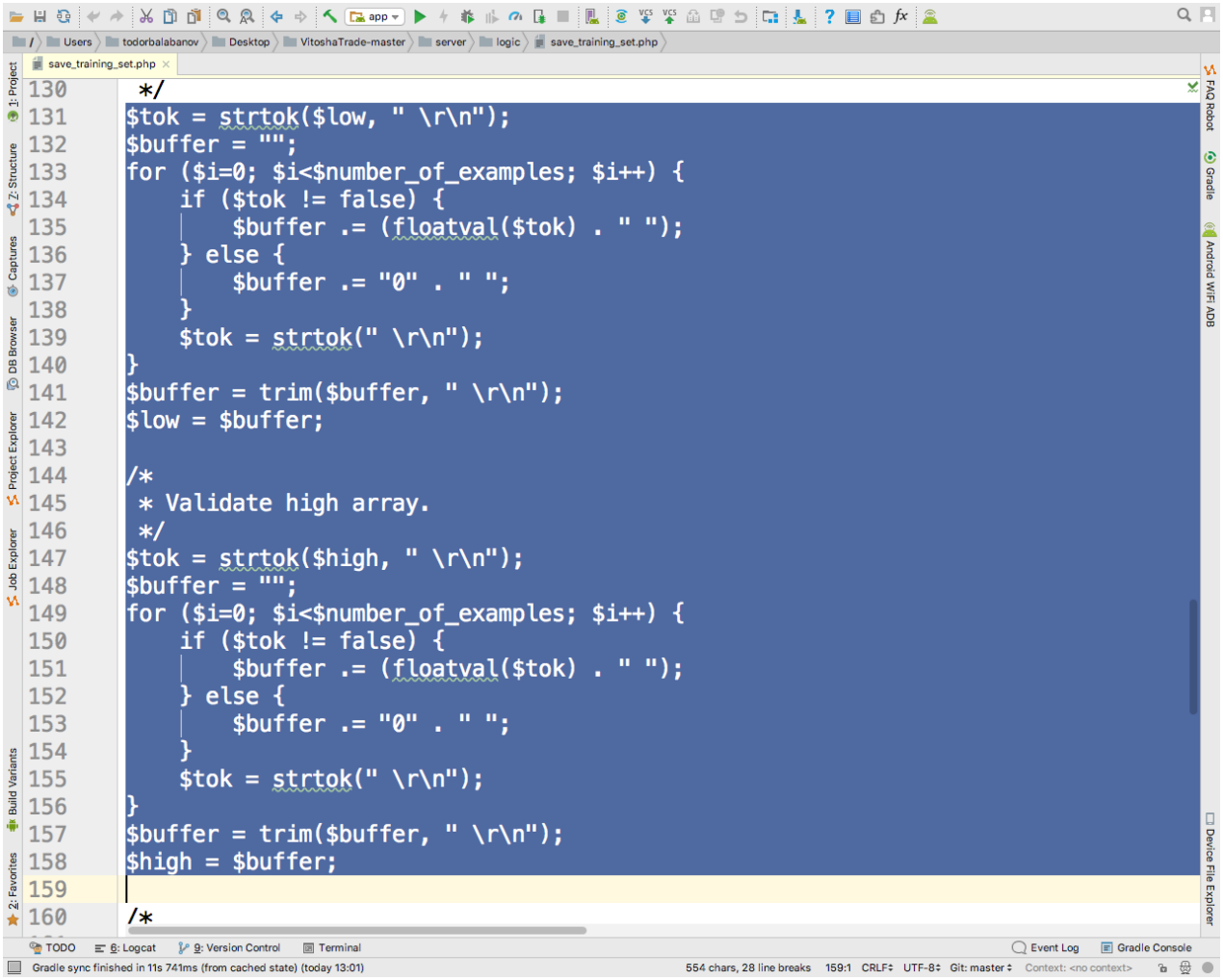


Фигура 5.57: Проверка на агументите за отваряне, най-ниска стойност, най-висока стойност, затваряне и изтъргуван обем

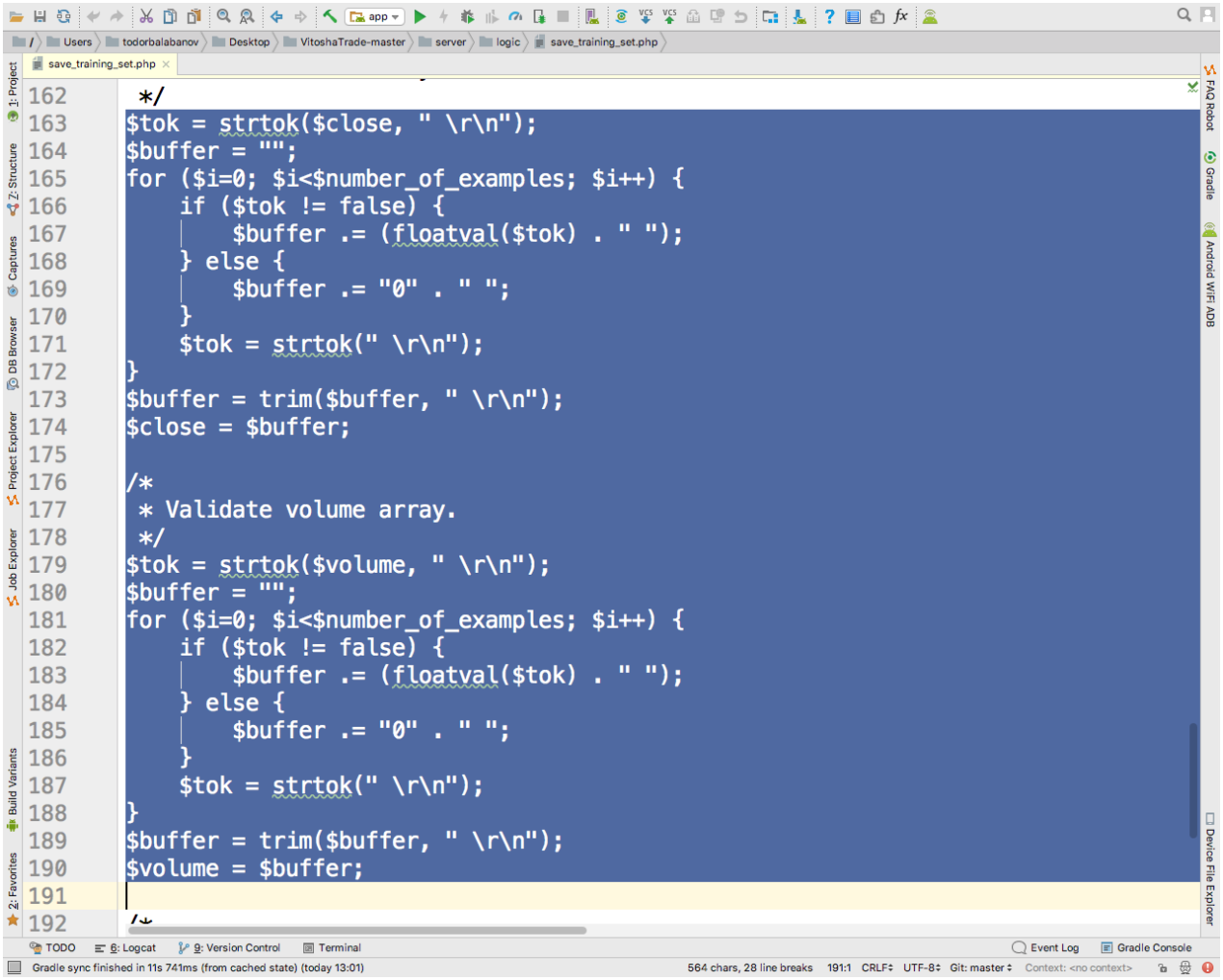
Всеки от масивите преминава допълнителна проверка за стойностите, които съдържа (Фиг. 5.58-5.60).



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

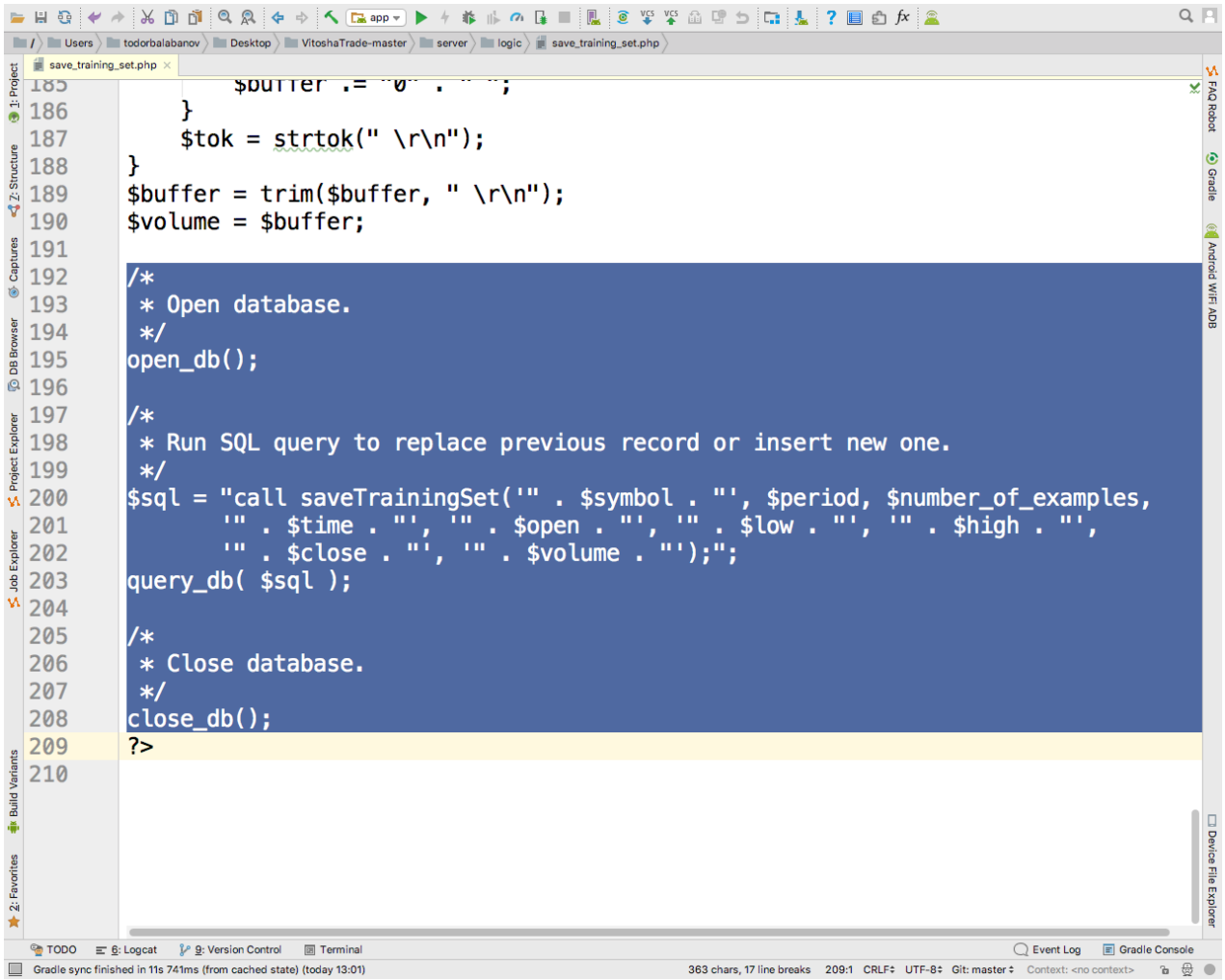


Фигура 5.59: Проверка на стойностите за най-ниско и най-високо постигнато ниво



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).



Фигура 5.61: Съхраняване на тренировъчно множество

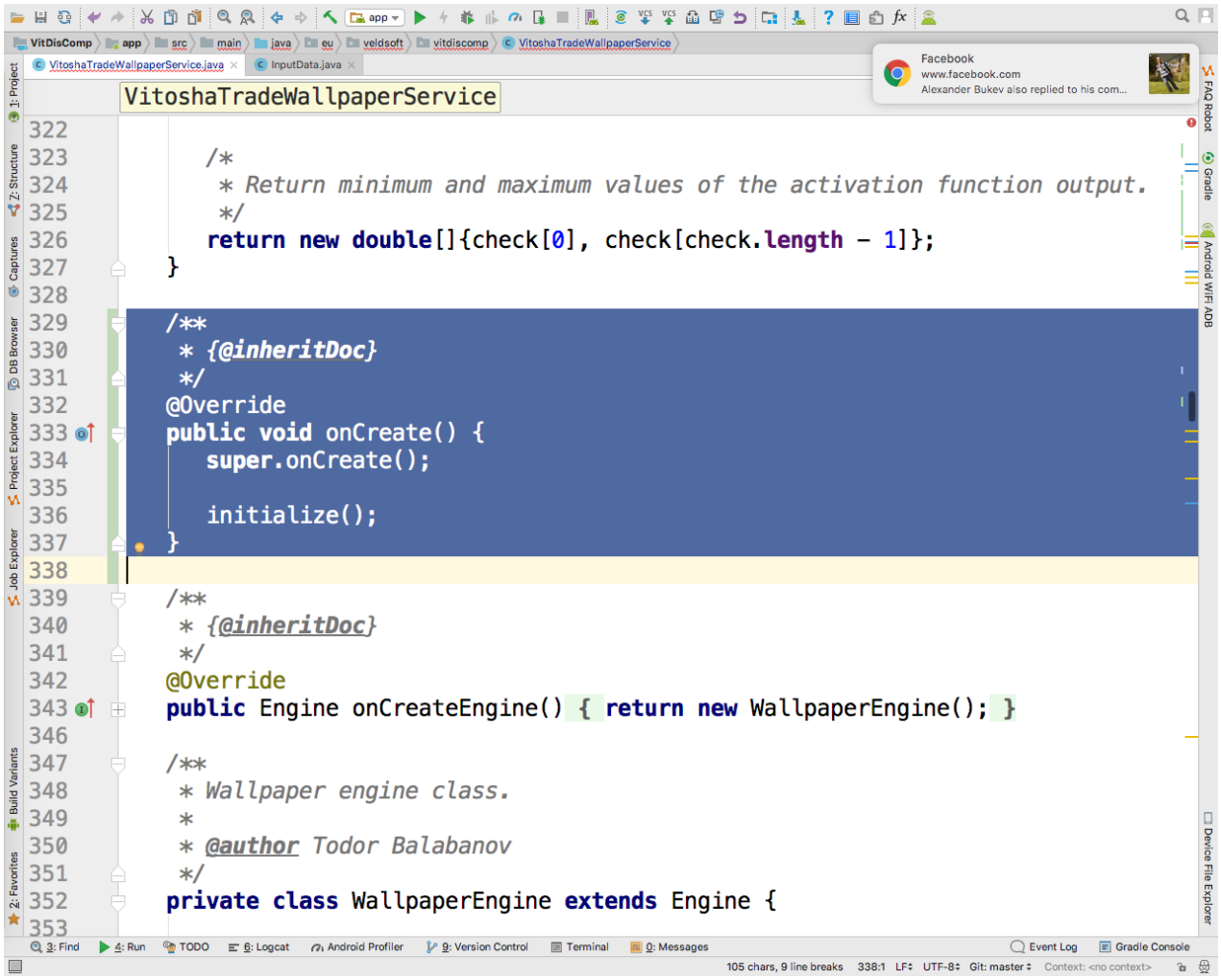
Глава 6

Комуникация клиент-сървър

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

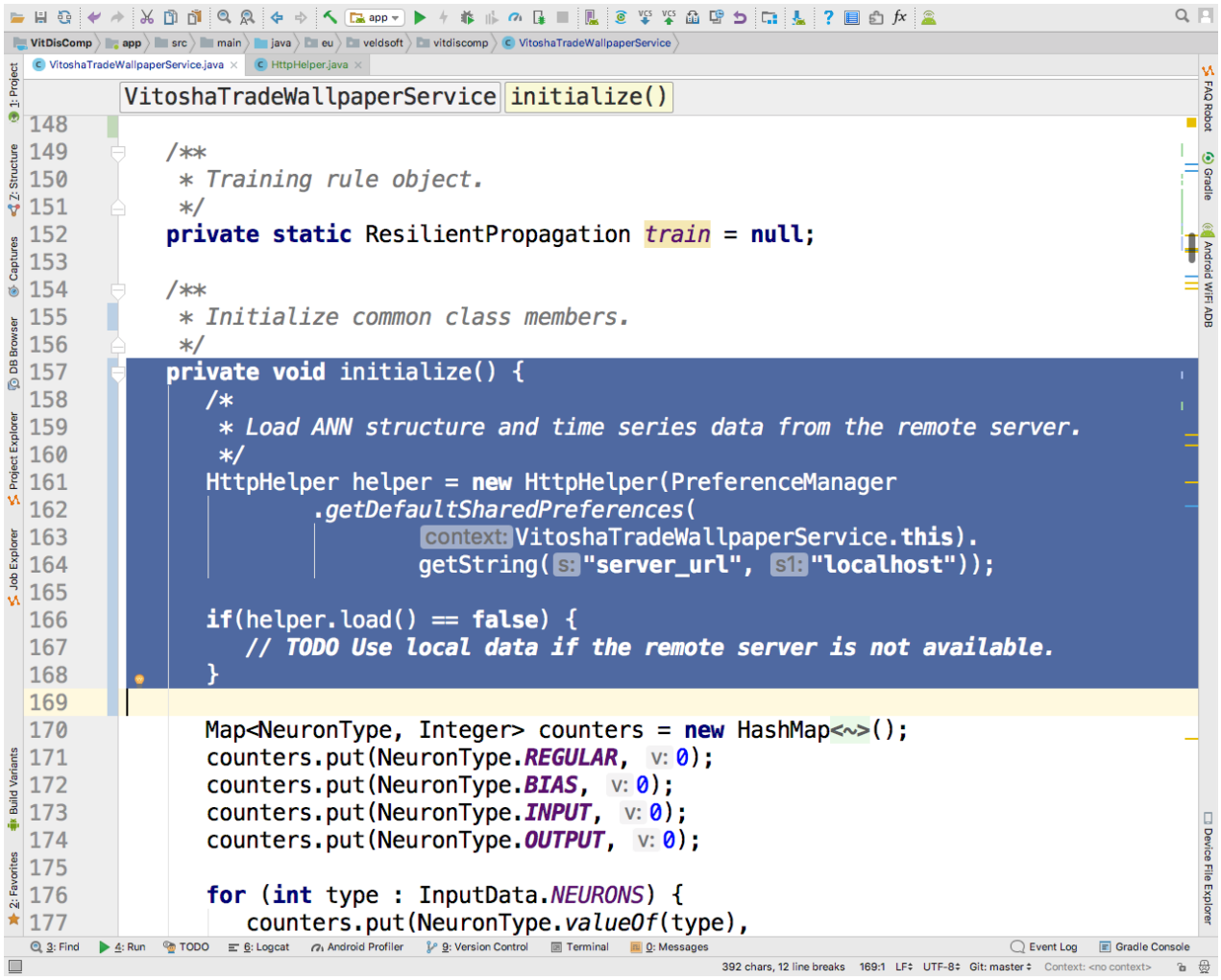
6.1 Hypertext Transfer Protocol

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.



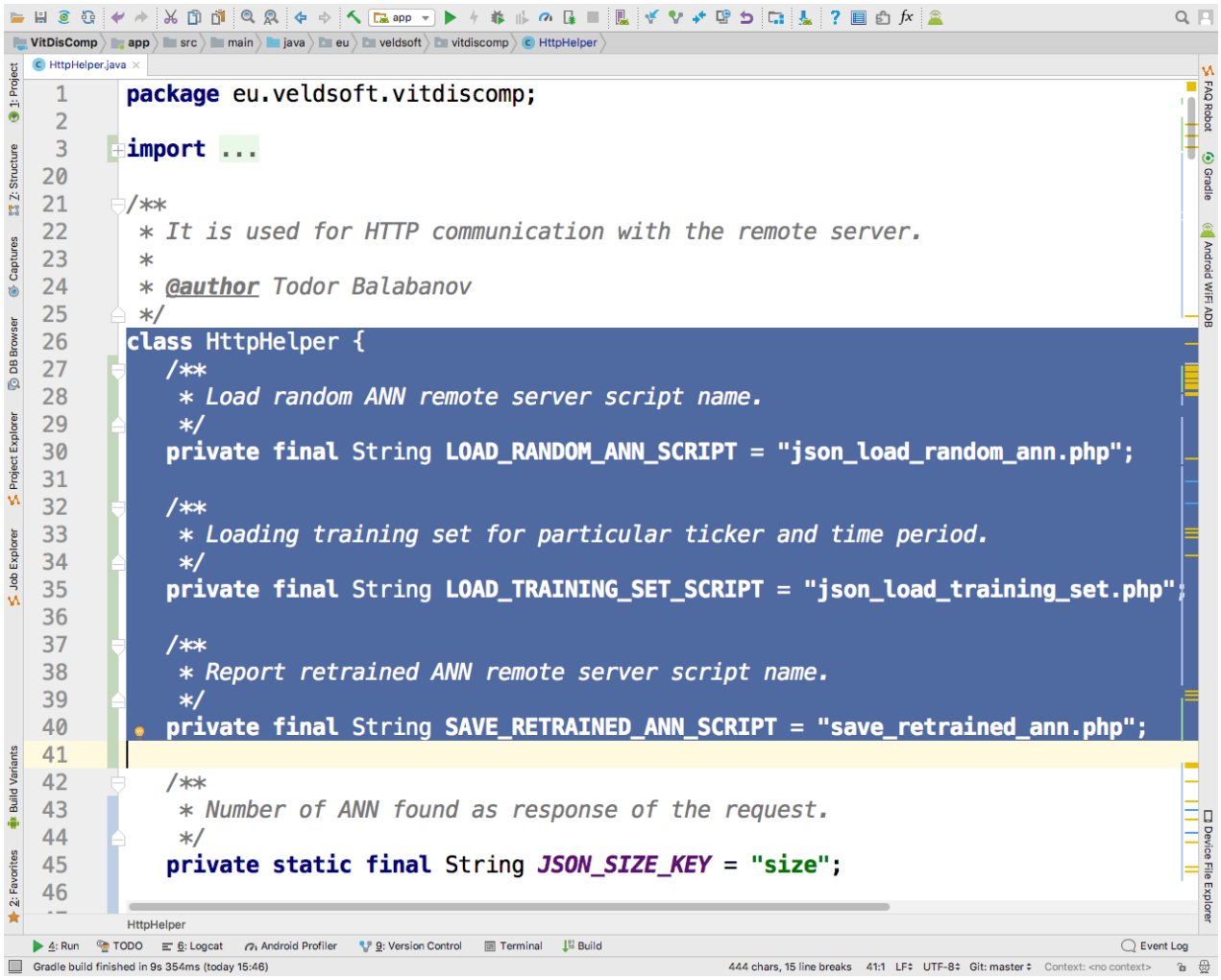
Фигура 6.1: Инициализация на вътрешните променливи

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



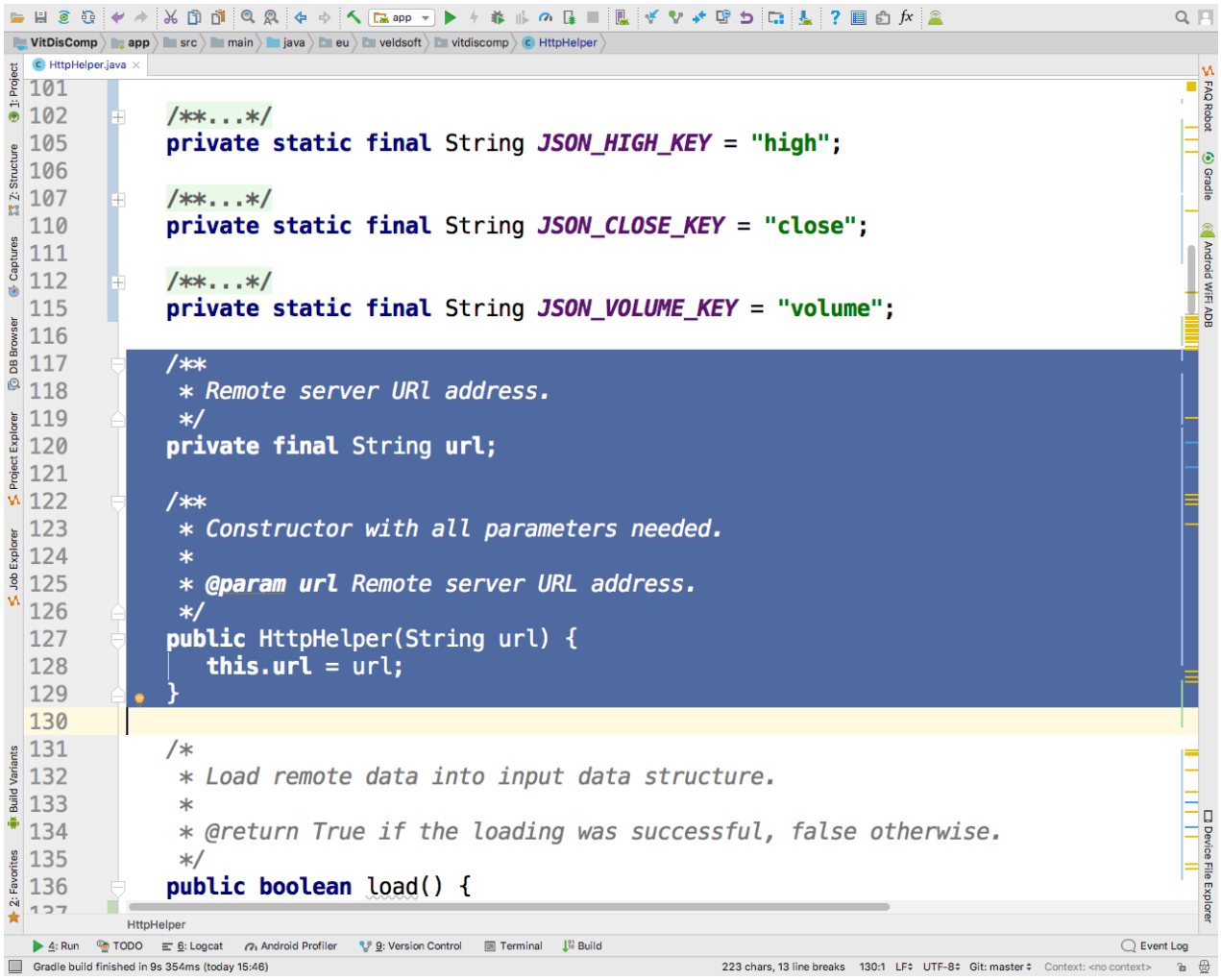
Фигура 6.2: Зареждане на данните с помощта на HTTP комуникация

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



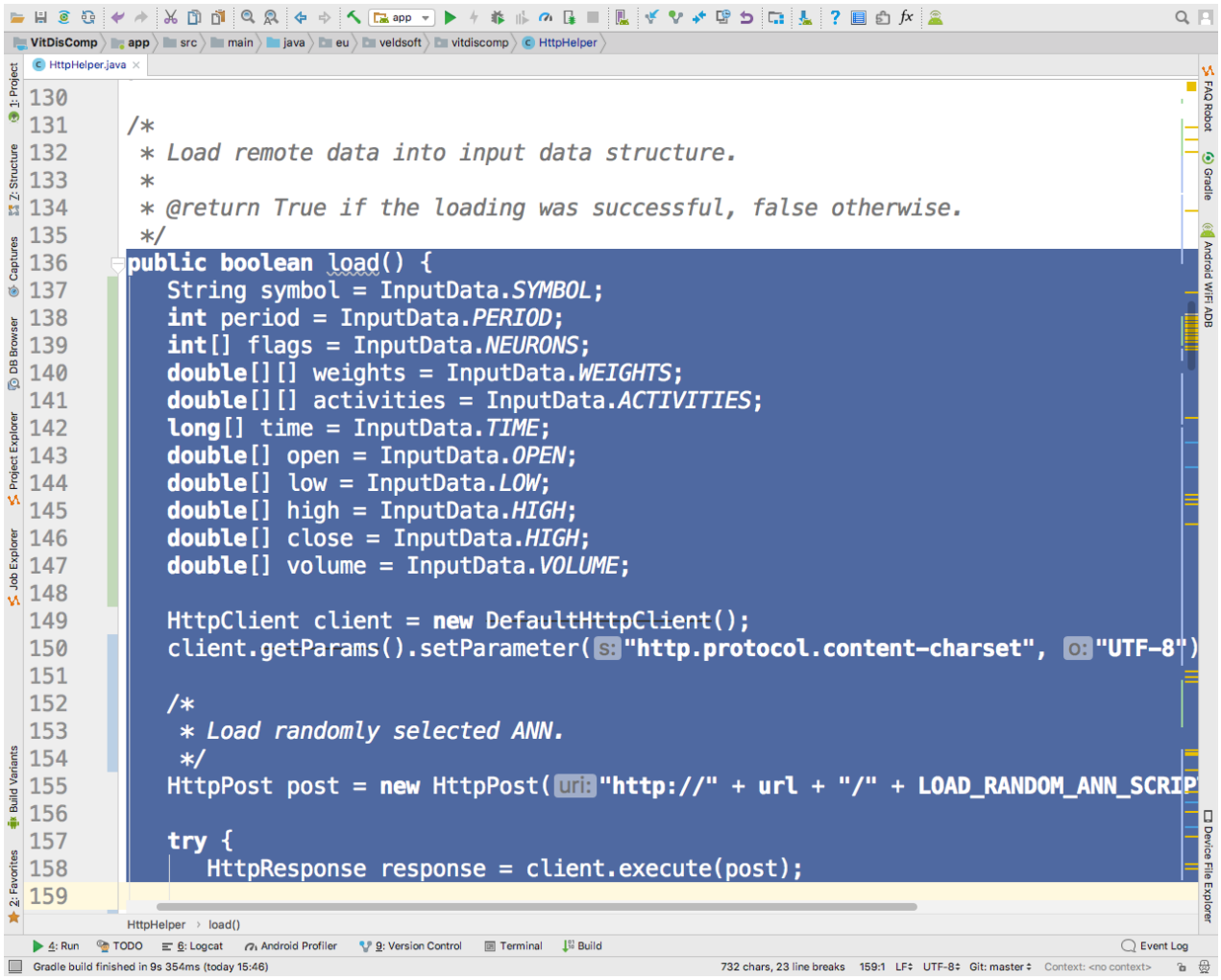
Фигура 6.3: HTTP комуникационен файл

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 именувани 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.



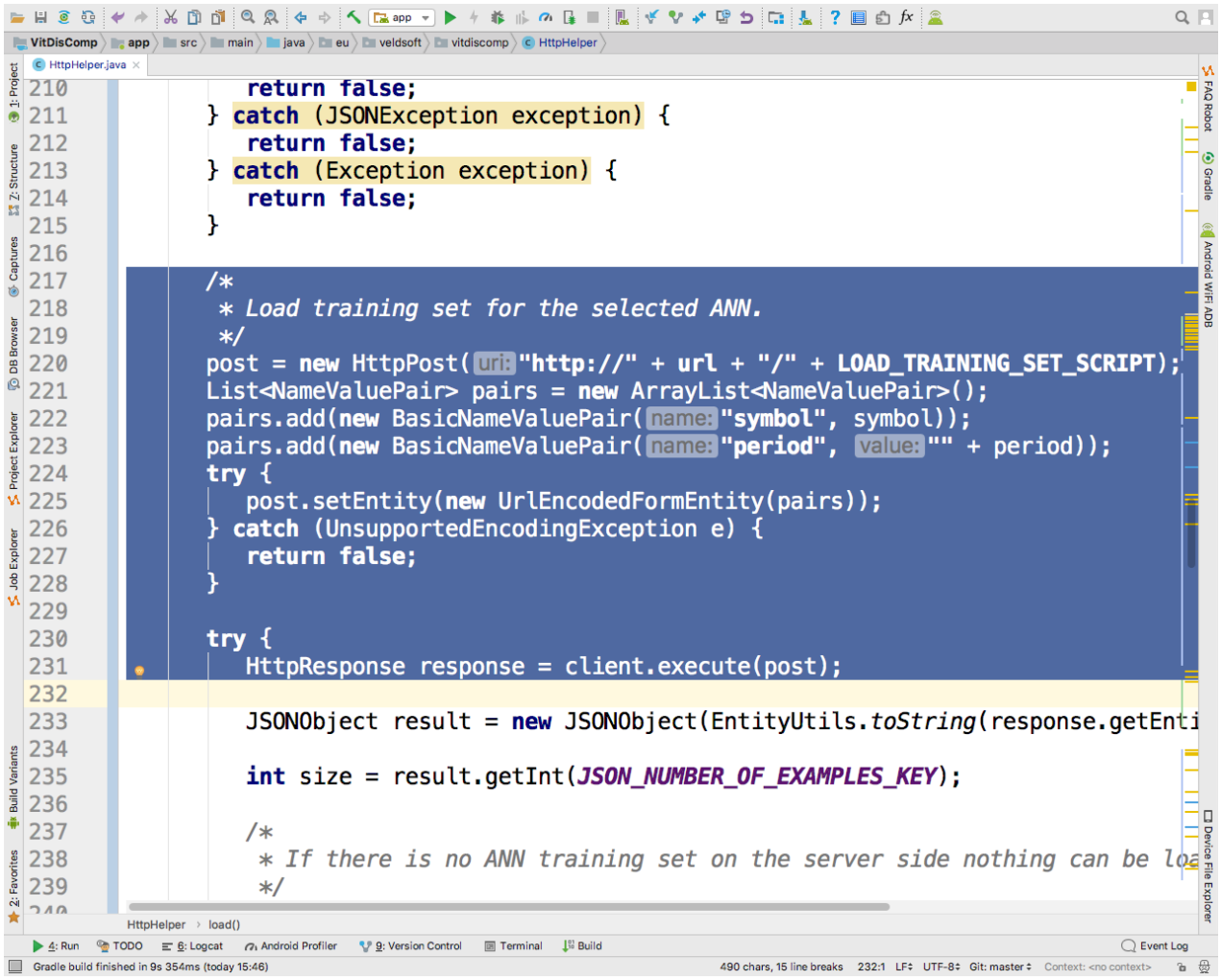
Фигура 6.4: Конструктор на спомагателния клас

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).



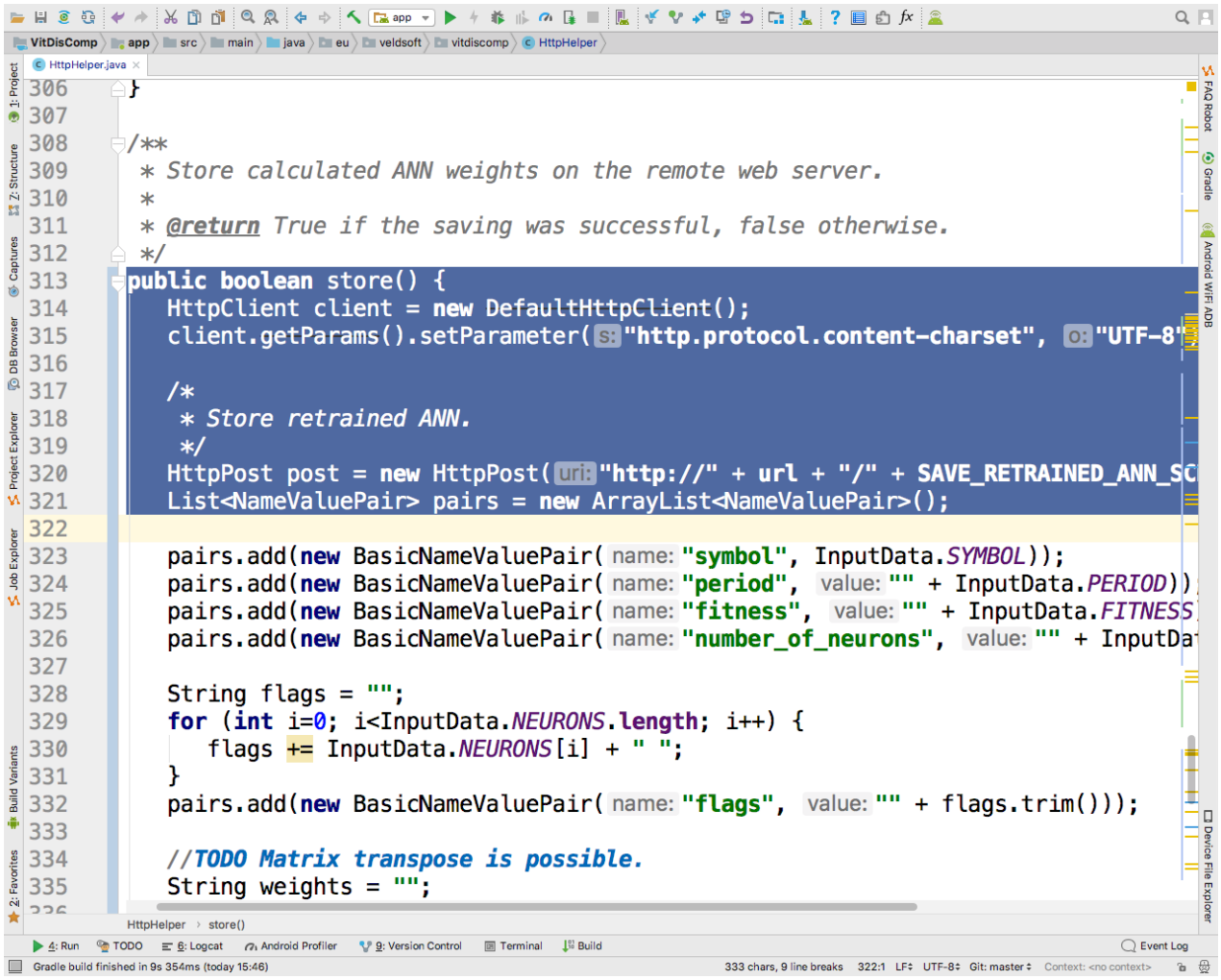
Фигура 6.5: Функция за зареждане на случайно избрана от сървъра мрежа

За зараждане 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).



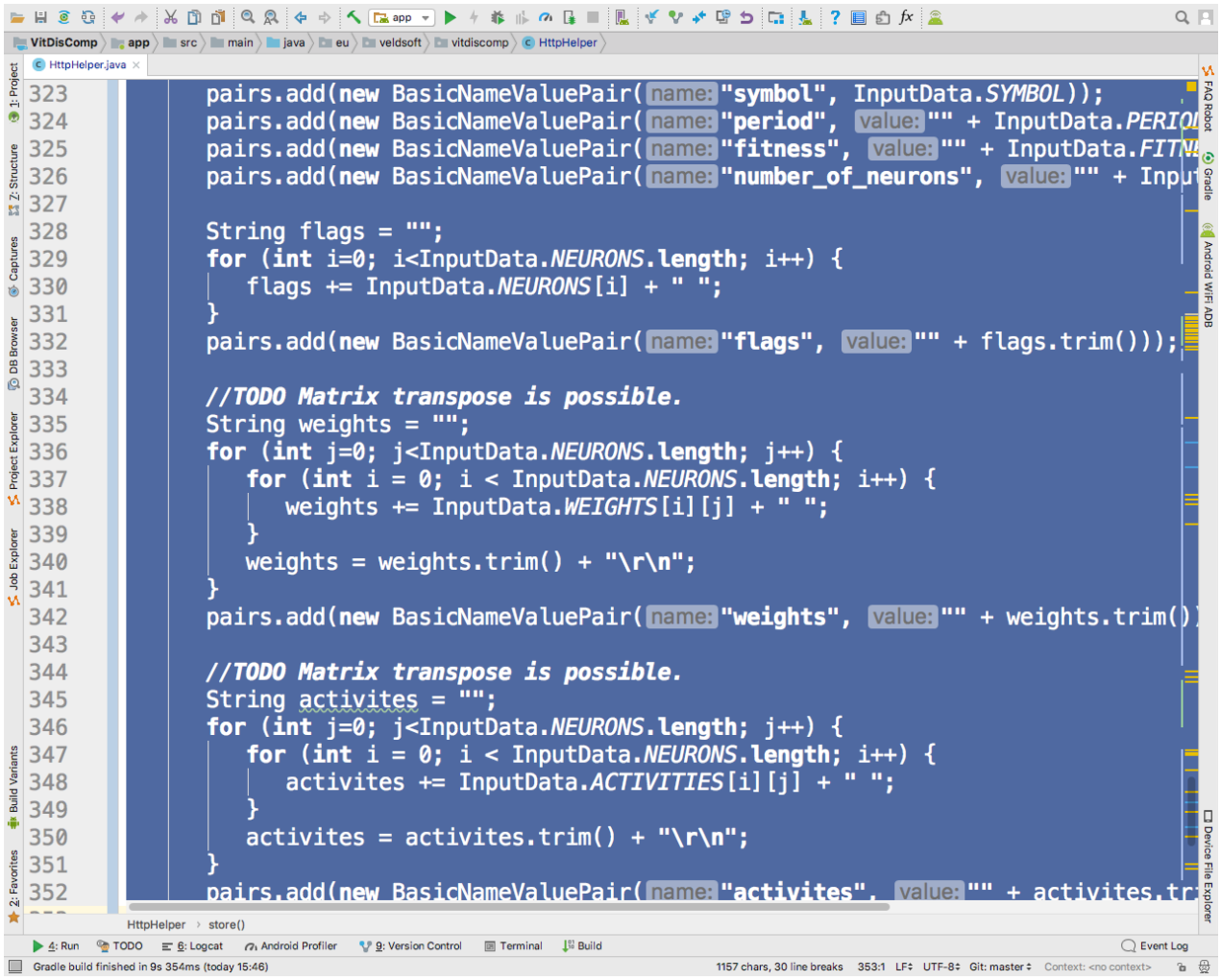
Фигура 6.6: Зареждане на тренировъчно множество

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.



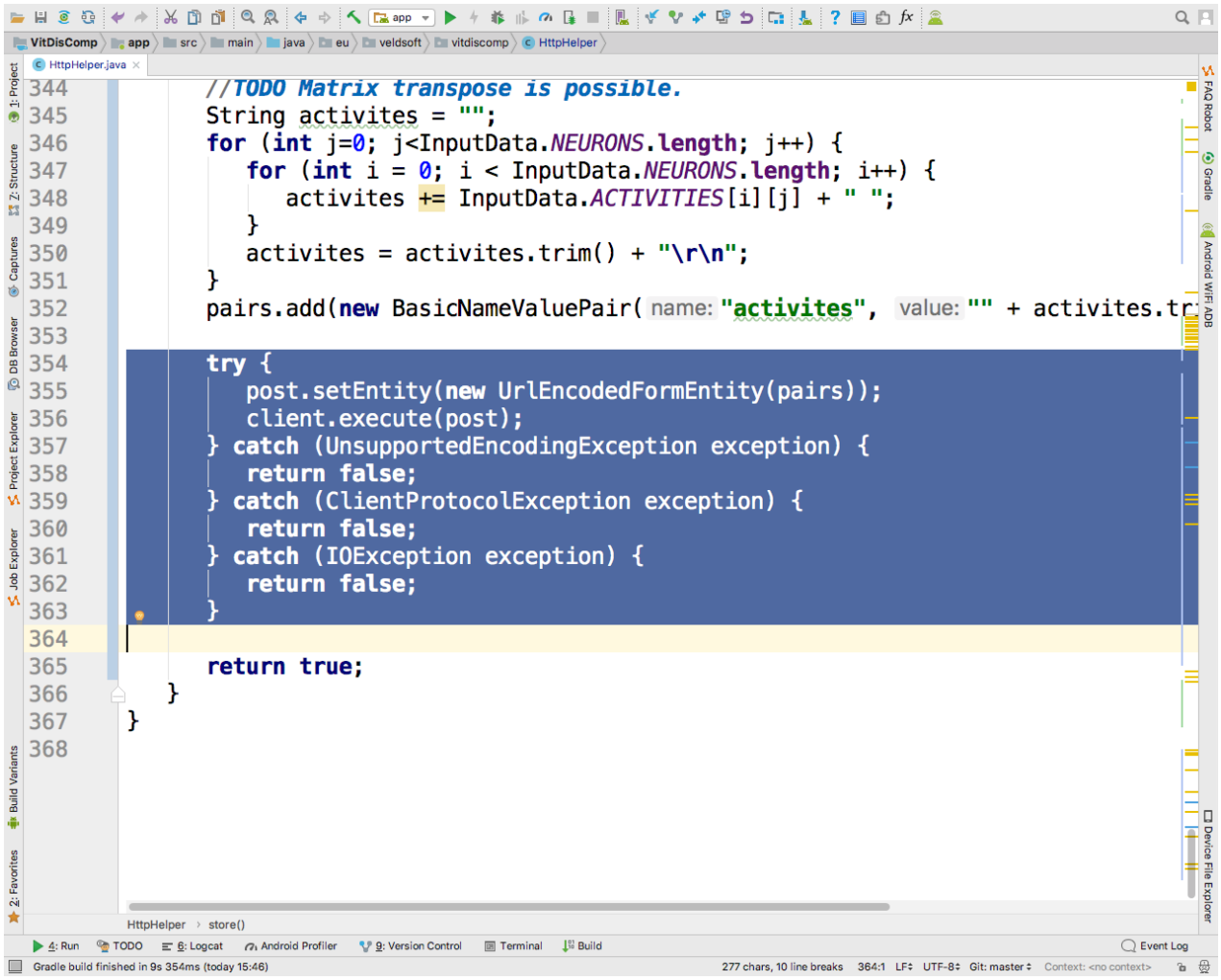
Фигура 6.7: Функция за запазване на допълнително обучавана мрежа

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).



Фигура 6.8: Подготвяне 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 текстови (Фиг. 6.8).

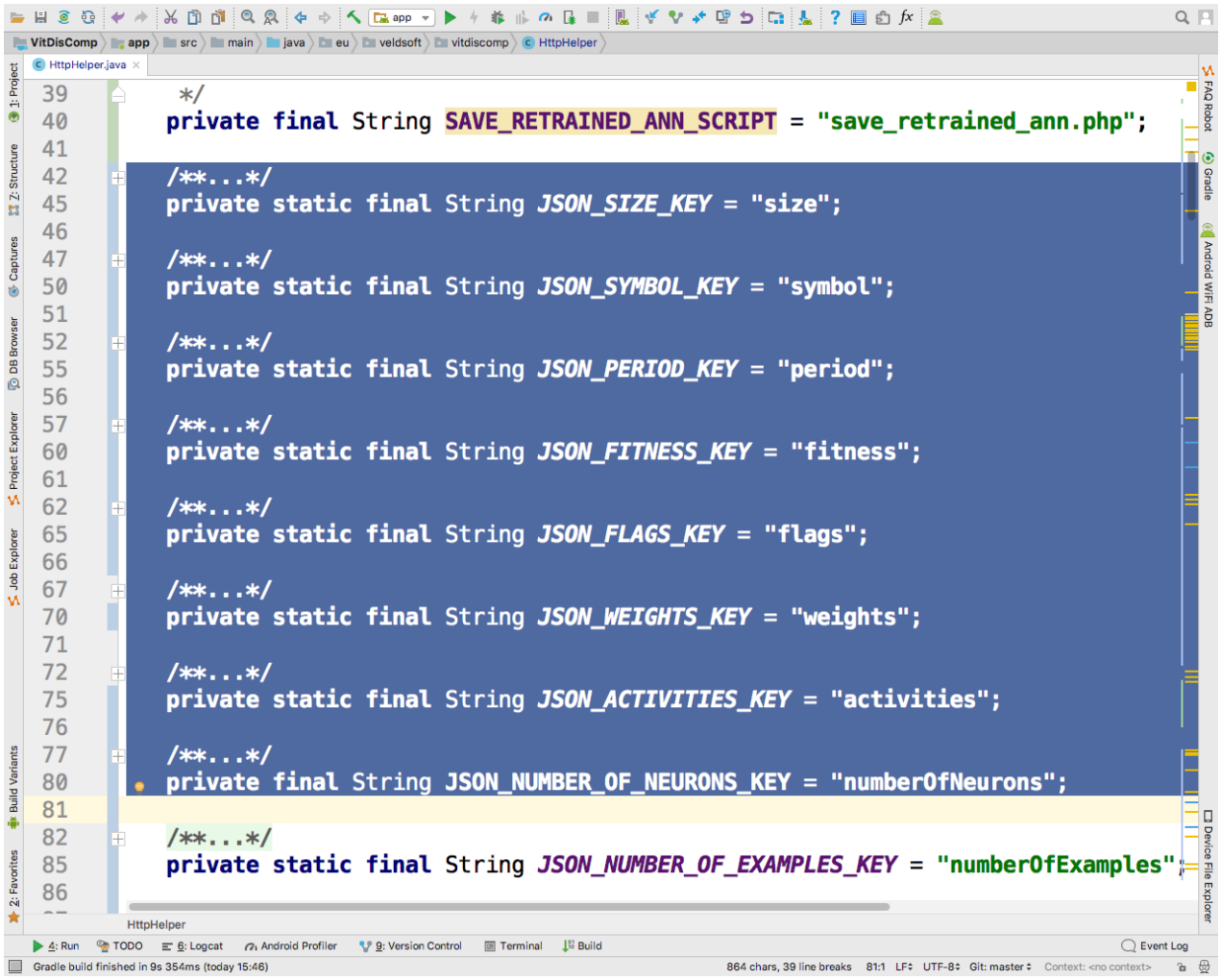


Фигура 6.9: Изпълнение на POST заявката

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

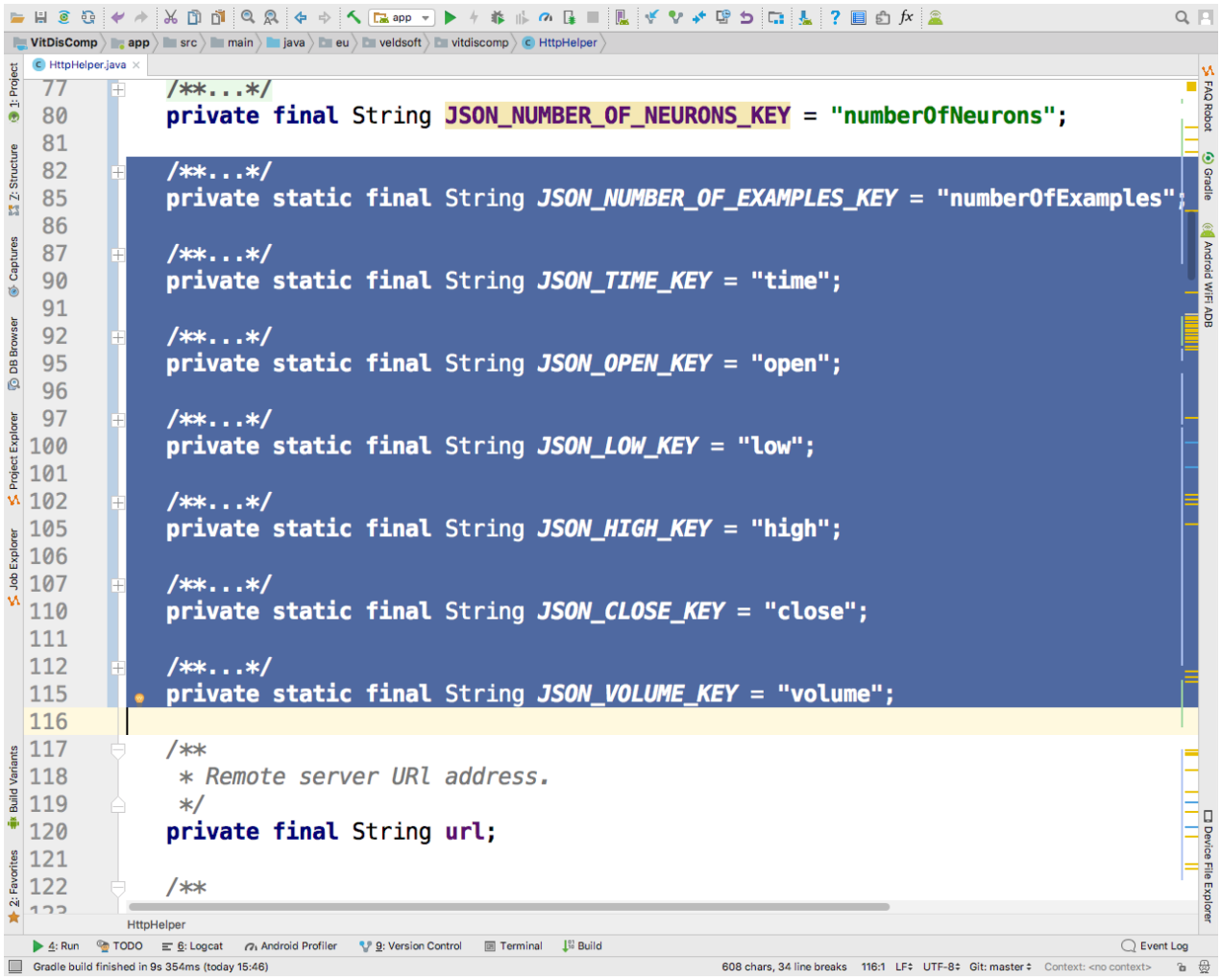
6.2 JavaScript Object Notation

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.



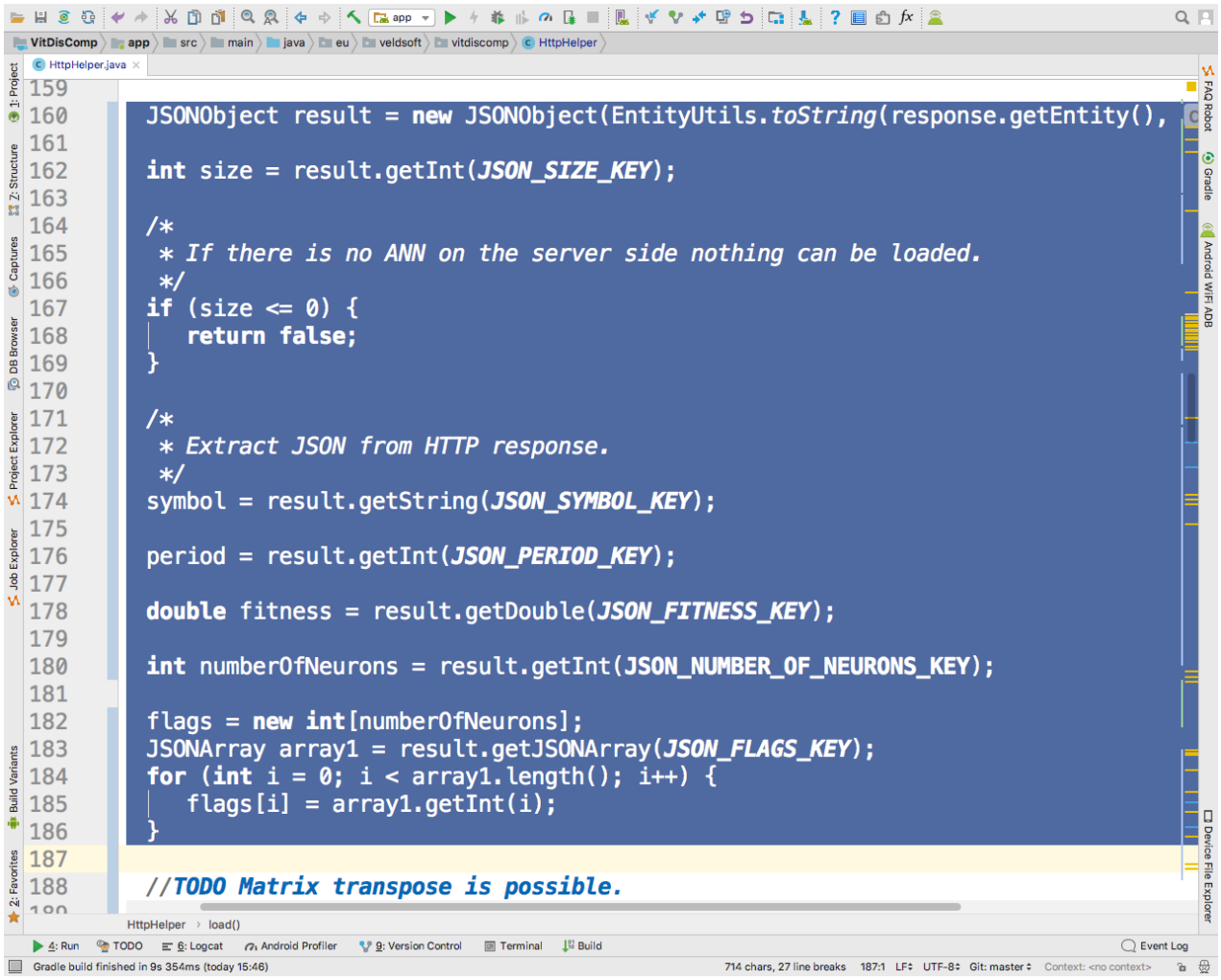
Фигура 6.10: Ключови стойности за описване на мрежата

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).



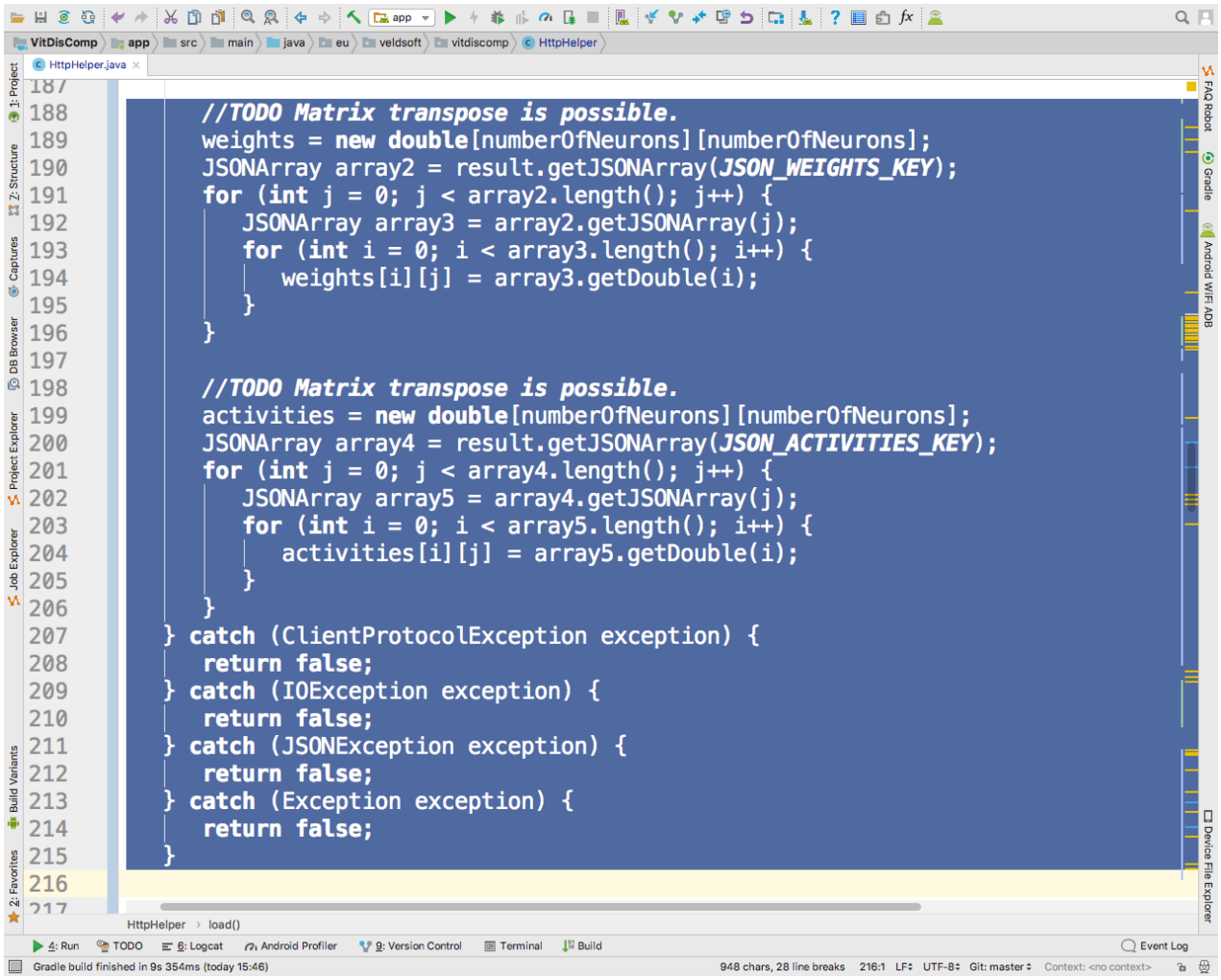
Фигура 6.11: Ключови стойности за описване на тренировъчното множество

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.



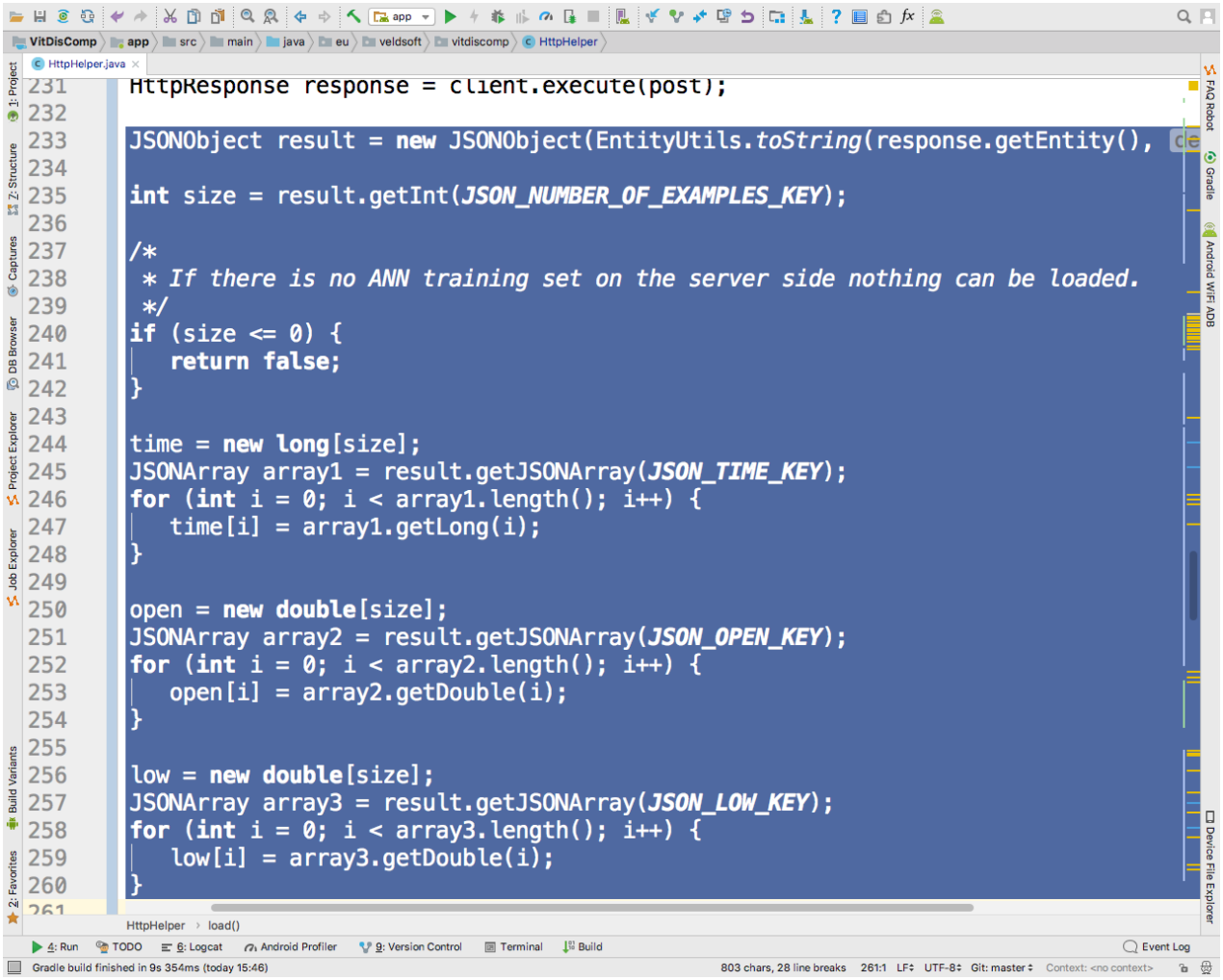
Фигура 6.12: Стойности за времевия ред и невроните

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).

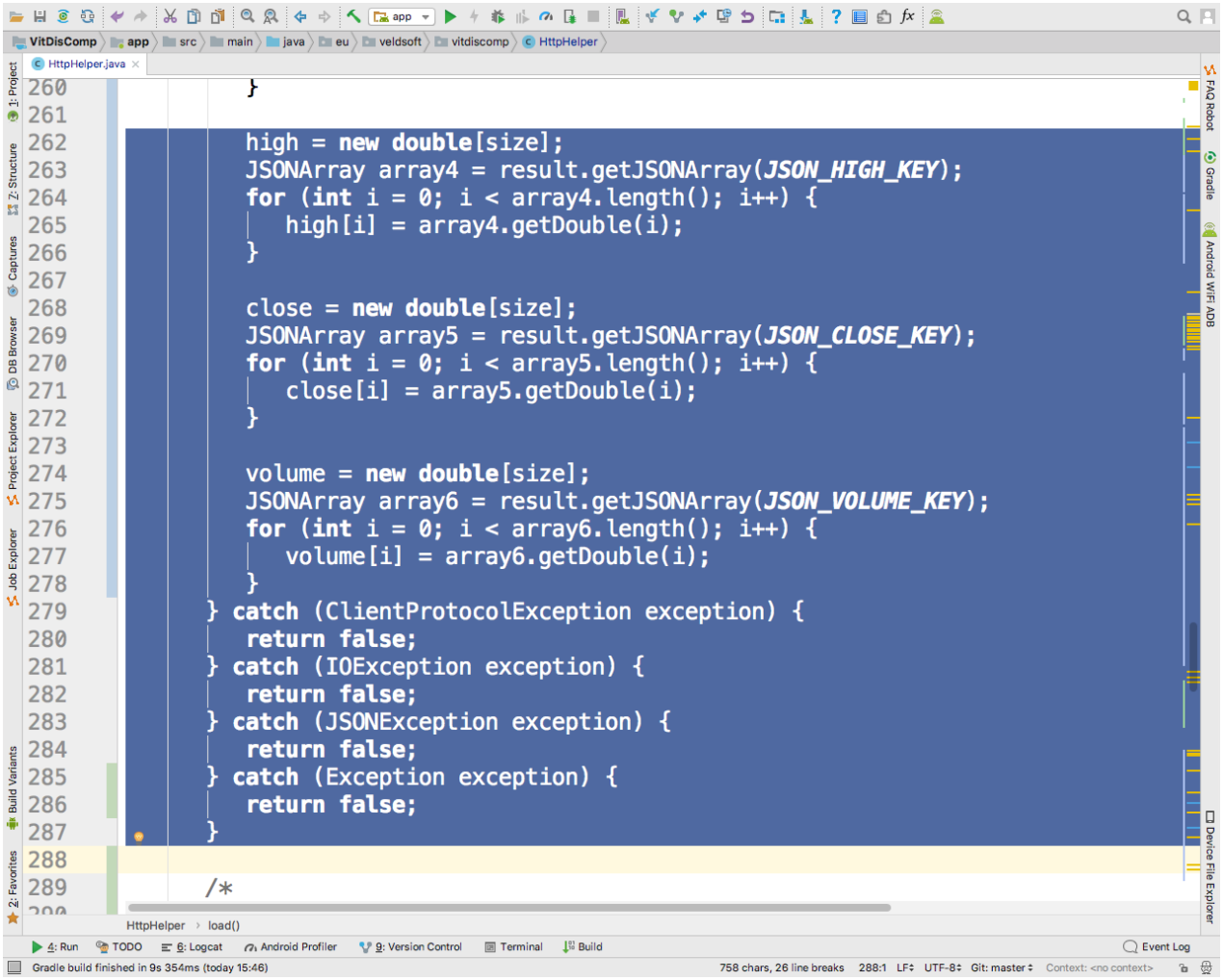


Фигура 6.13: Стойности за теглата и силата на връзките

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).

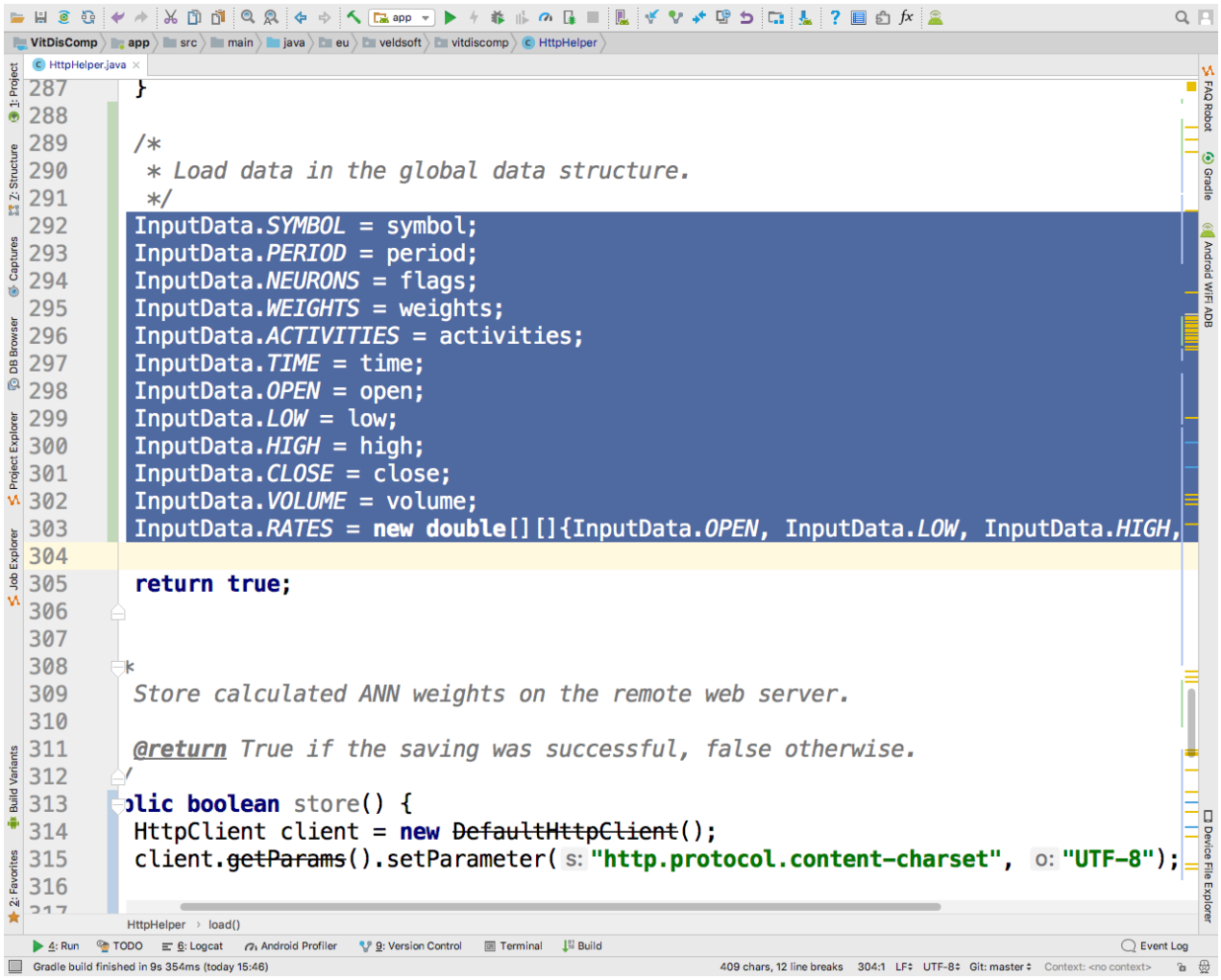


Фигура 6.14: Стойности за време, отваряне и най-ниска постигната стойност



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).



Фигура 6.16: Записване на данните в глобалната структура

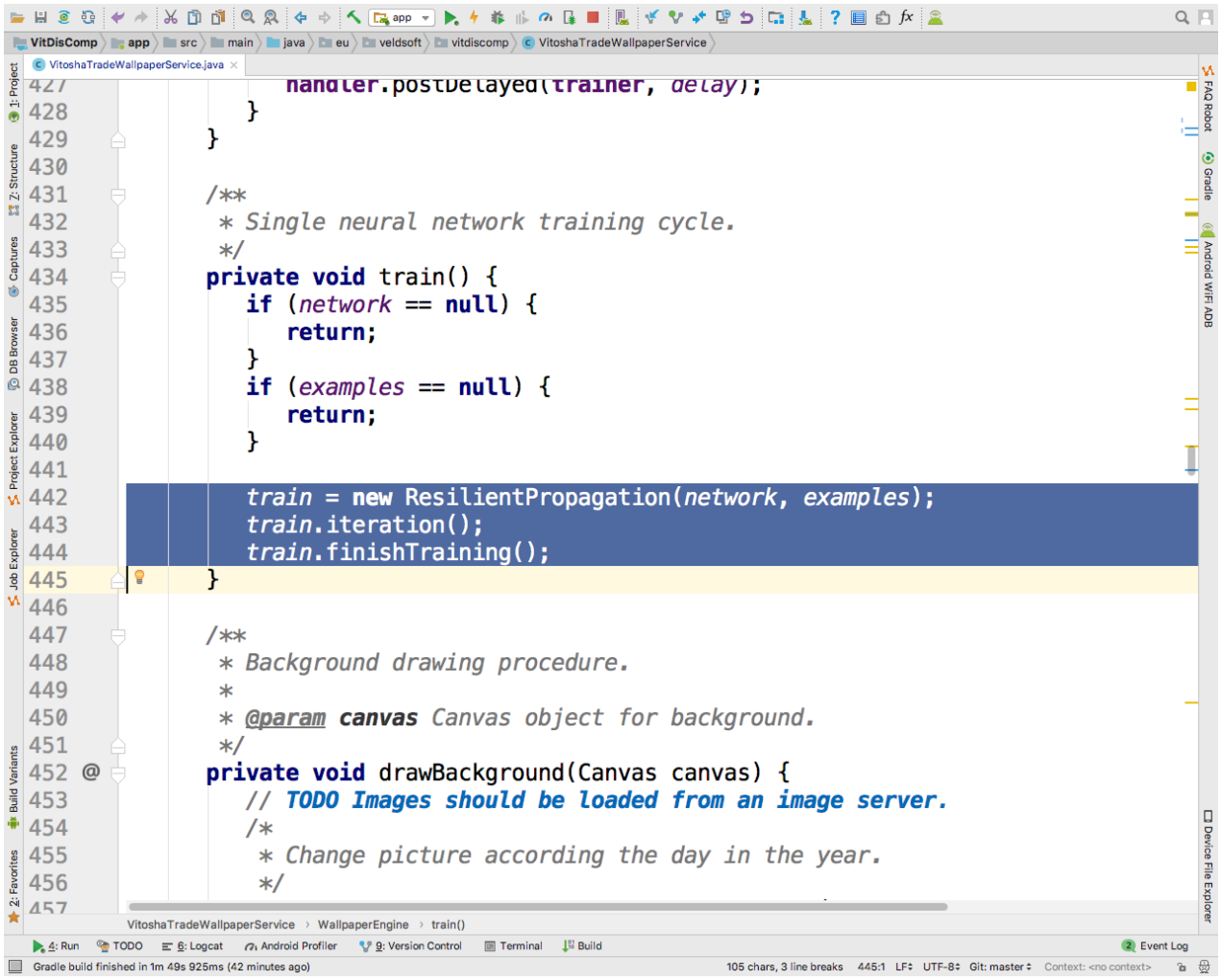
Глава 7

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 разликите. За реализацията на генетичния алгоритъм се използва софтуерната библиотека Genetic Algorithms - Apache Commons.

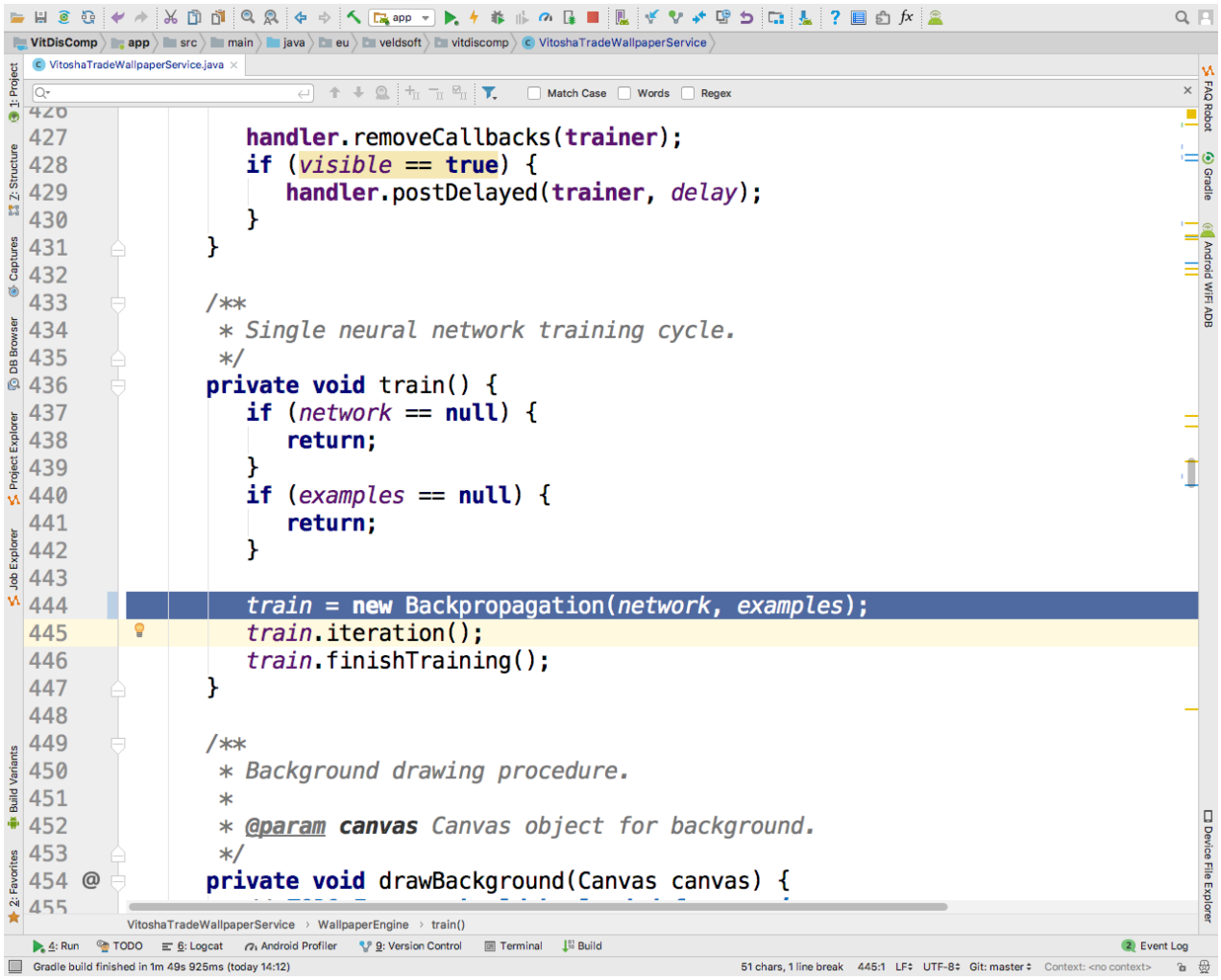
7.1 Обратно разпространение на грешката

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.



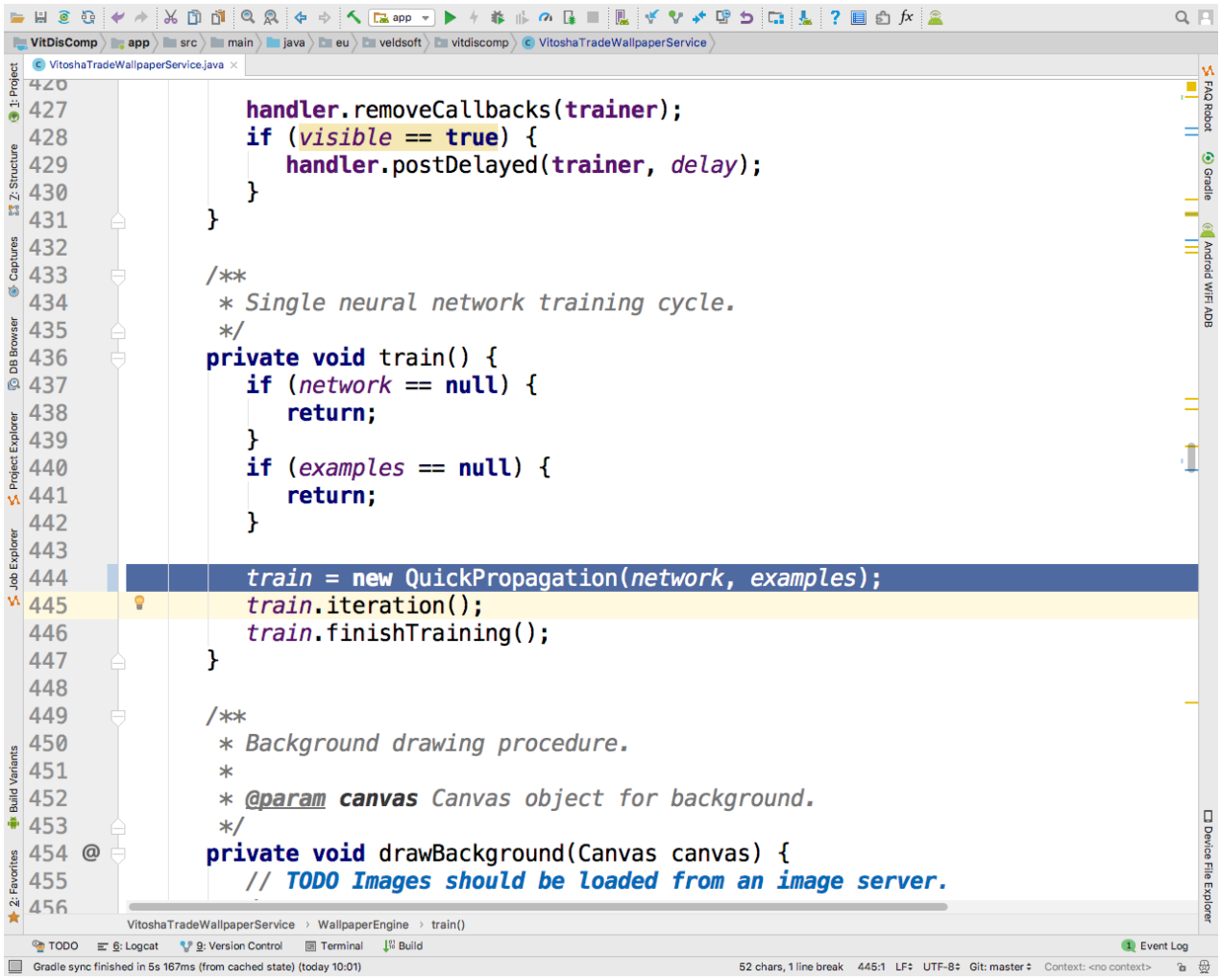
Фигура 7.1: Единична стъпка за обучение на мрежата

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 нагласяне на теглата в крайната фаза от обучението).



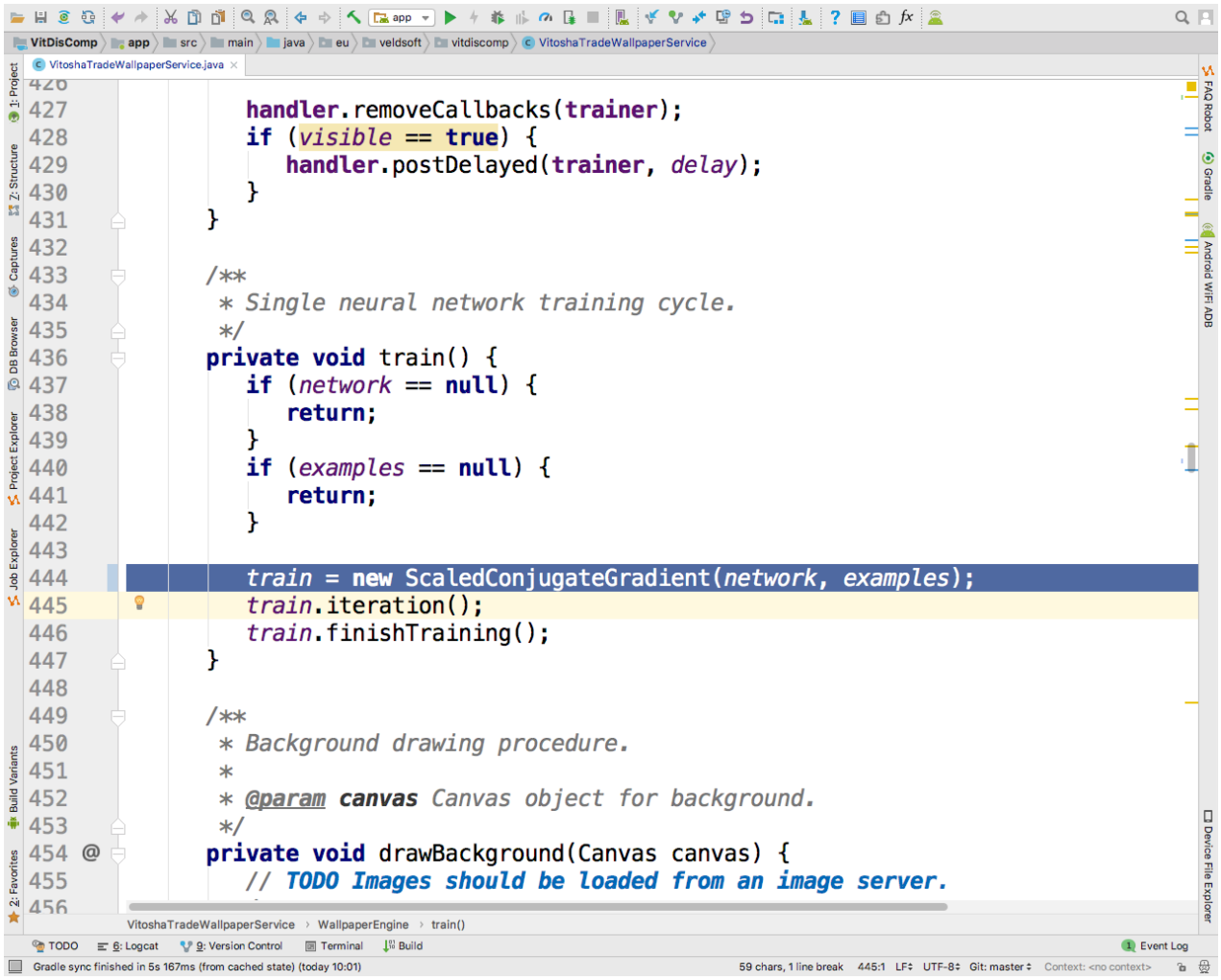
Фигура 7.2: Класическо обратно разпространение на грешката

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.



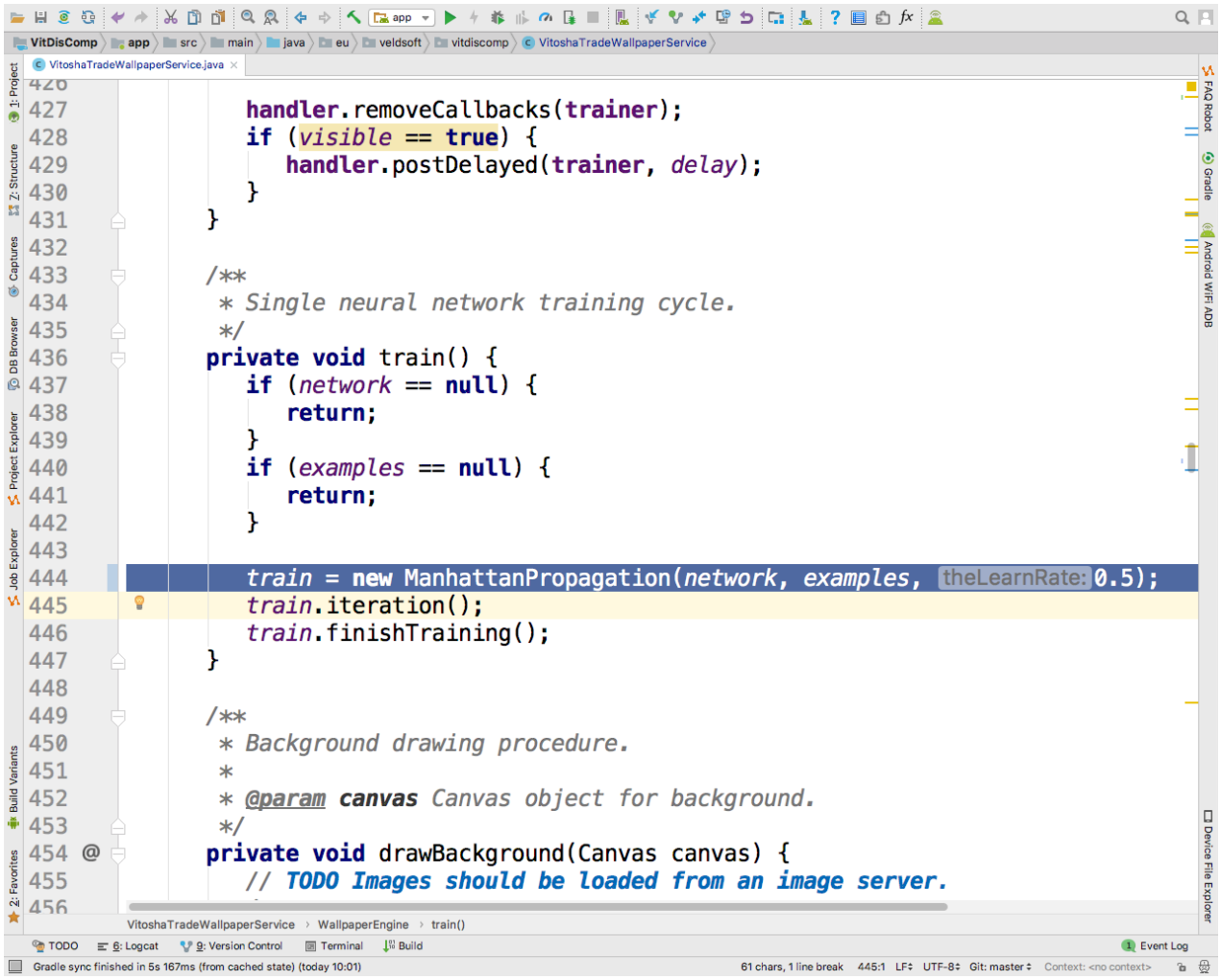
Фигура 7.3: Бързо разпространение на грешката

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.



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.

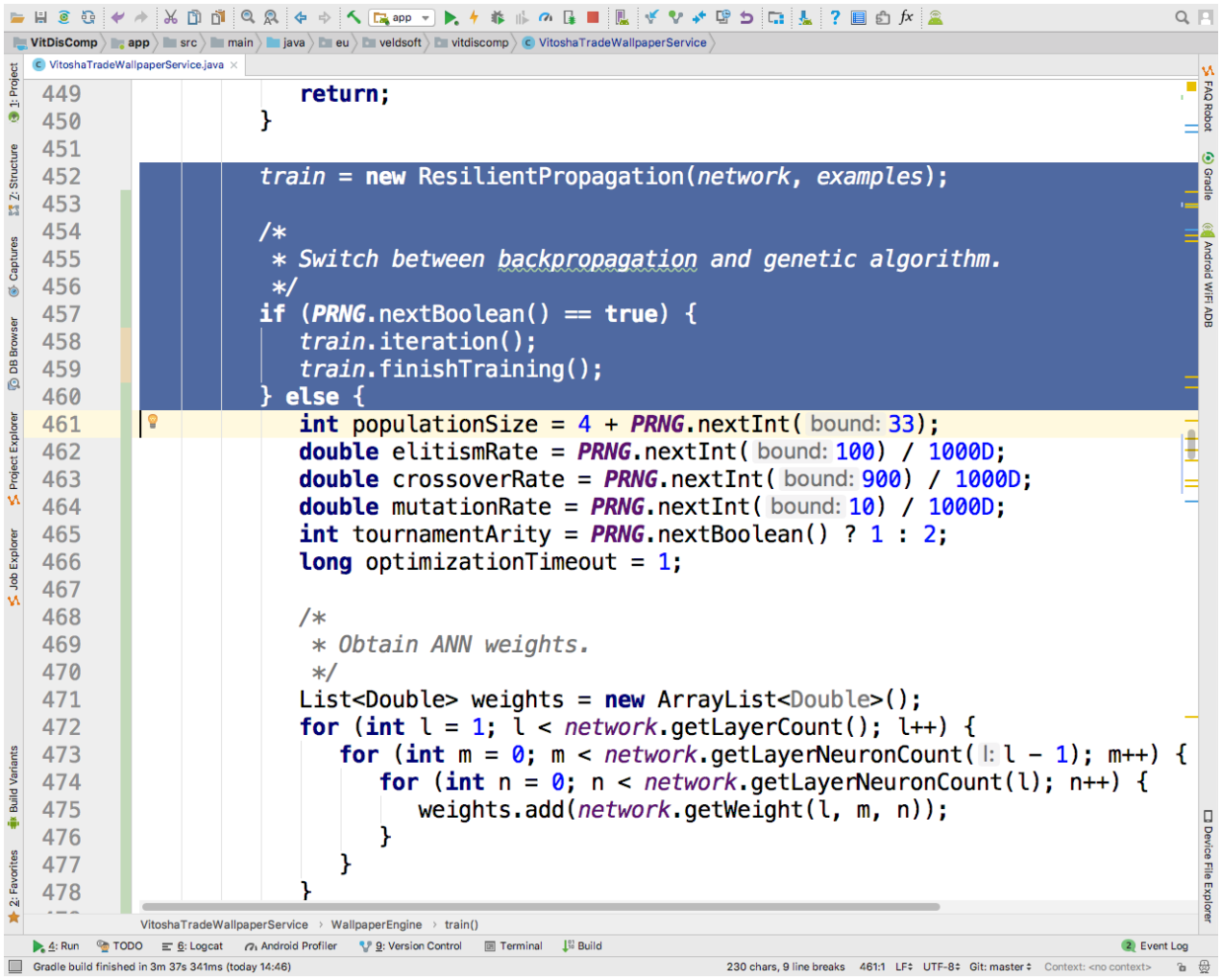


Фигура 7.5: Корекция на теглата по Манхатън правило

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.

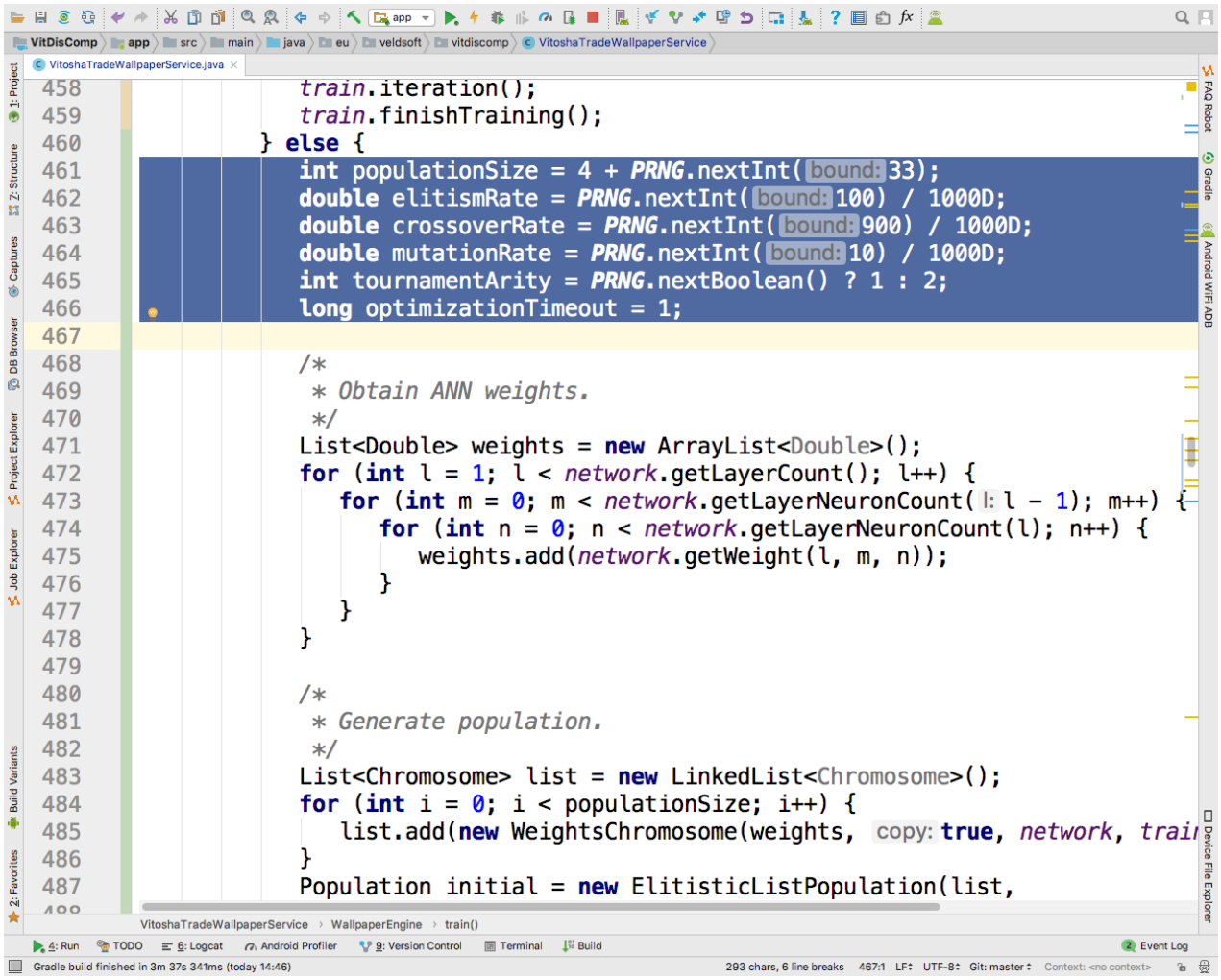
7.2 Генетични алгоритми

Въпреки че библиотеката Encog 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.



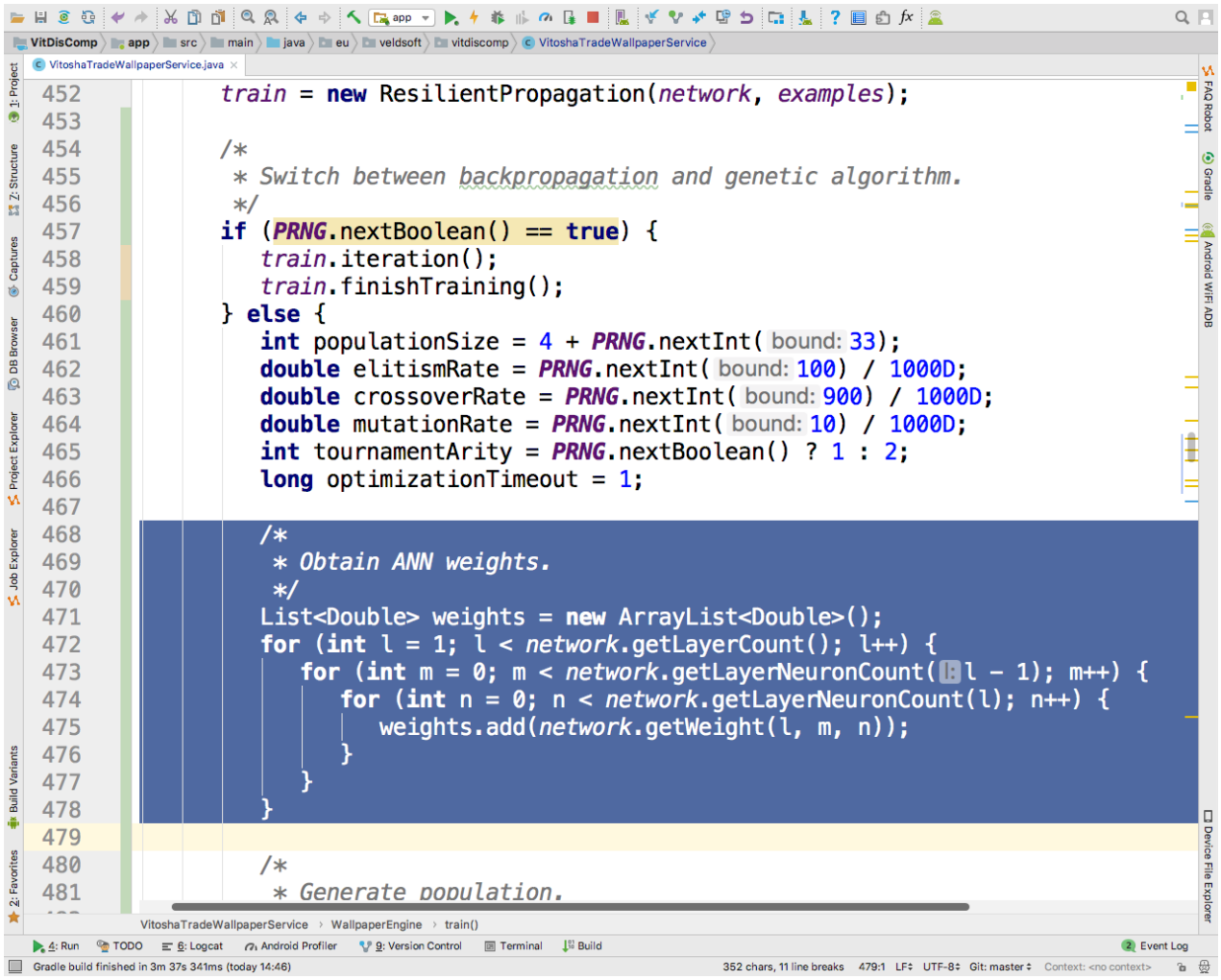
Фигура 7.6: Превключване между обратно разпространение на грешката и генетичен алгоритъм

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.



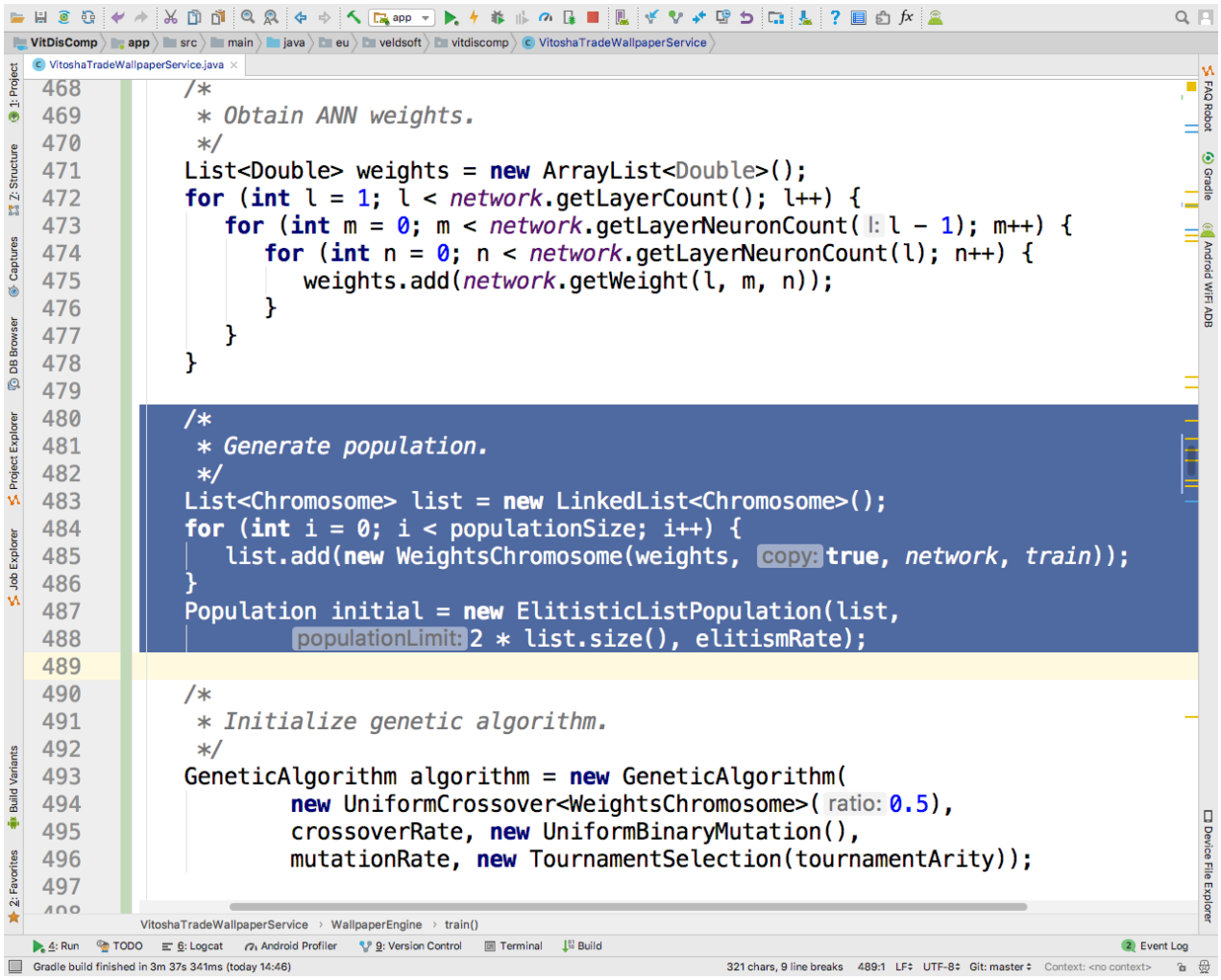
Фигура 7.7: Параметри на генетичния алгоритъм

Генетичният алгоритъм се настройва с група параметри (размер на популацията, процент на елита в популацията, честота за кръстосване, честота за мутация, численост за турнамента 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).



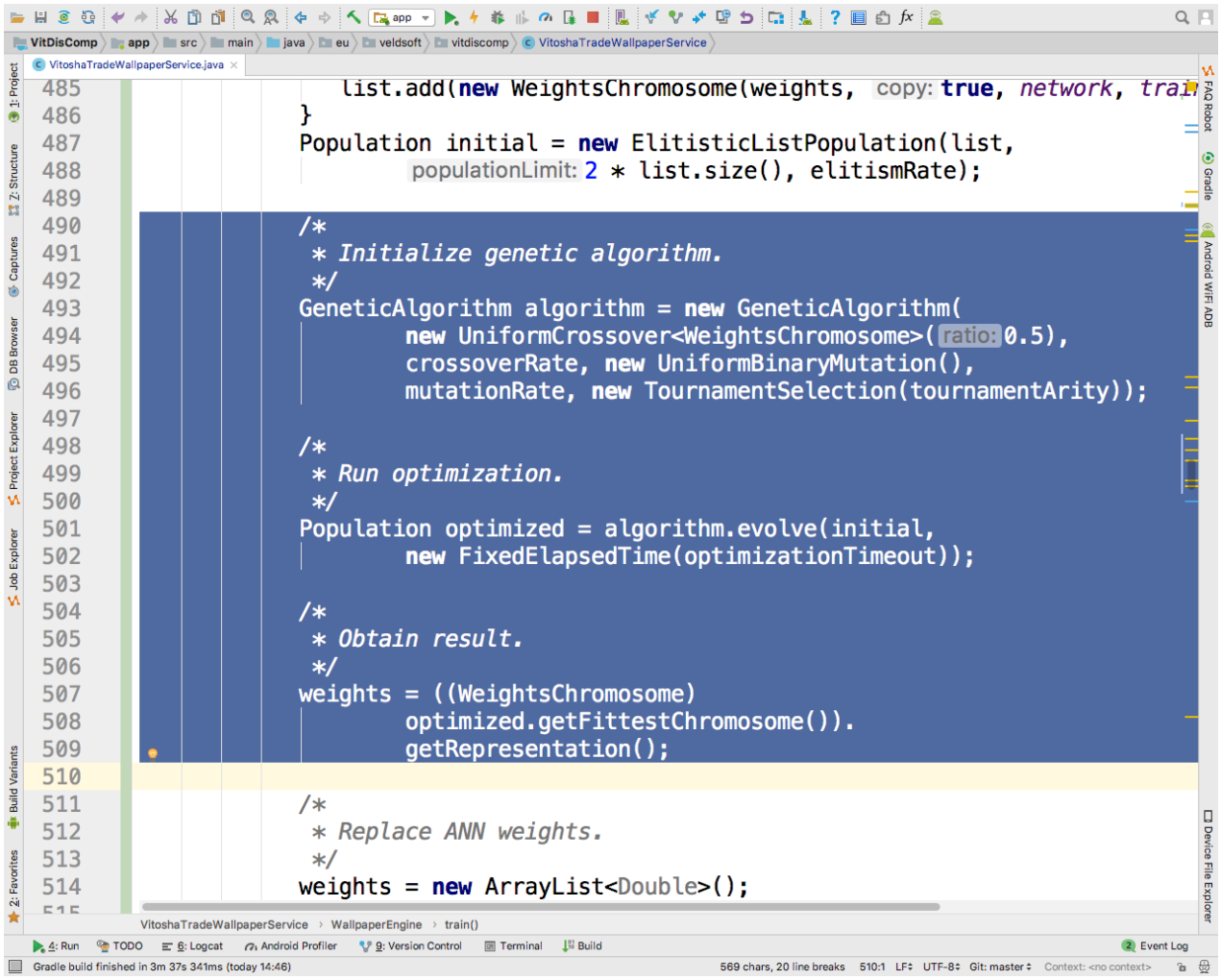
Фигура 7.8: Извличане на теглата от мрежата

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



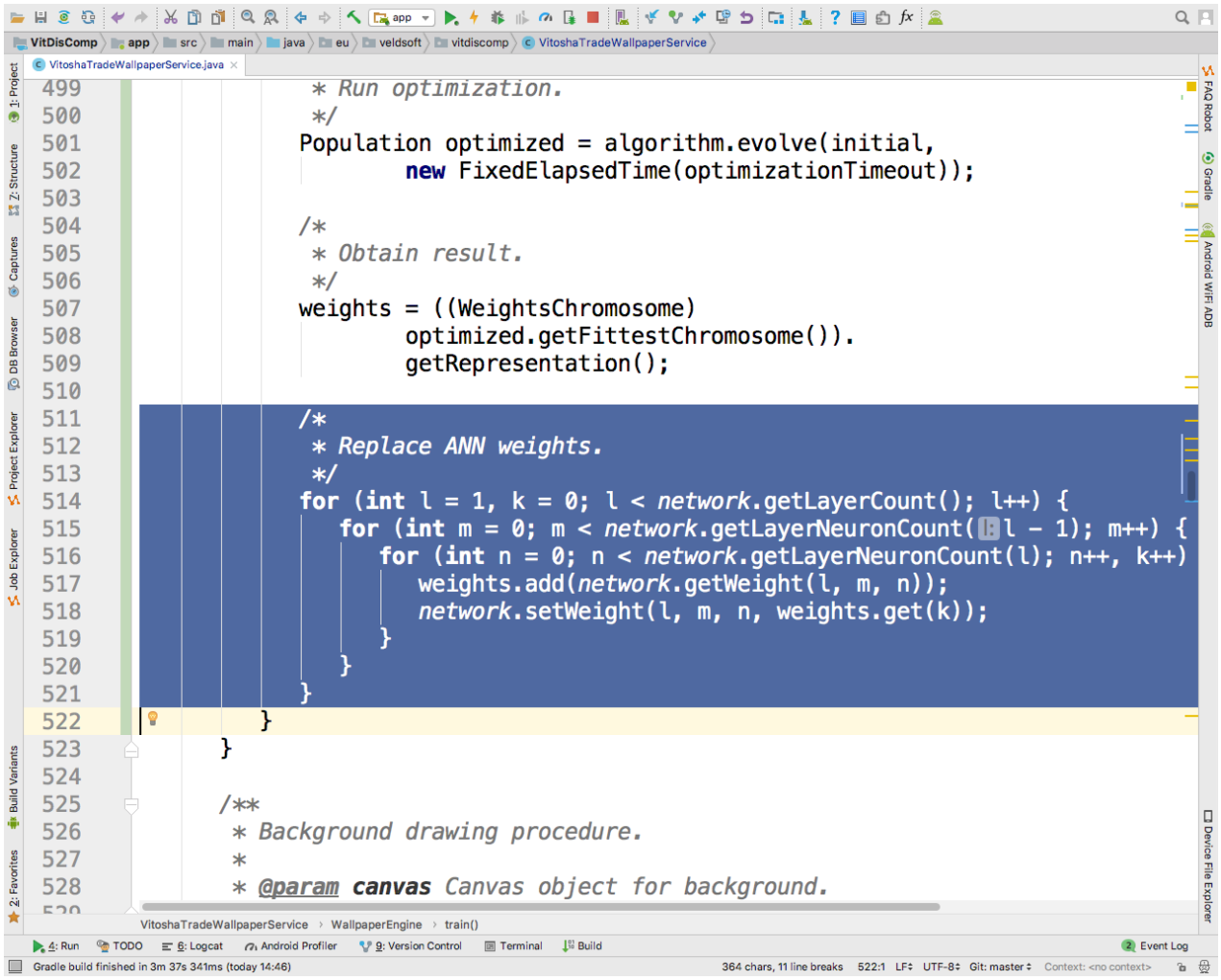
Фигура 7.9: Начална популация

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



Фигура 7.10: Изпълнение на генетичния алгоритъм

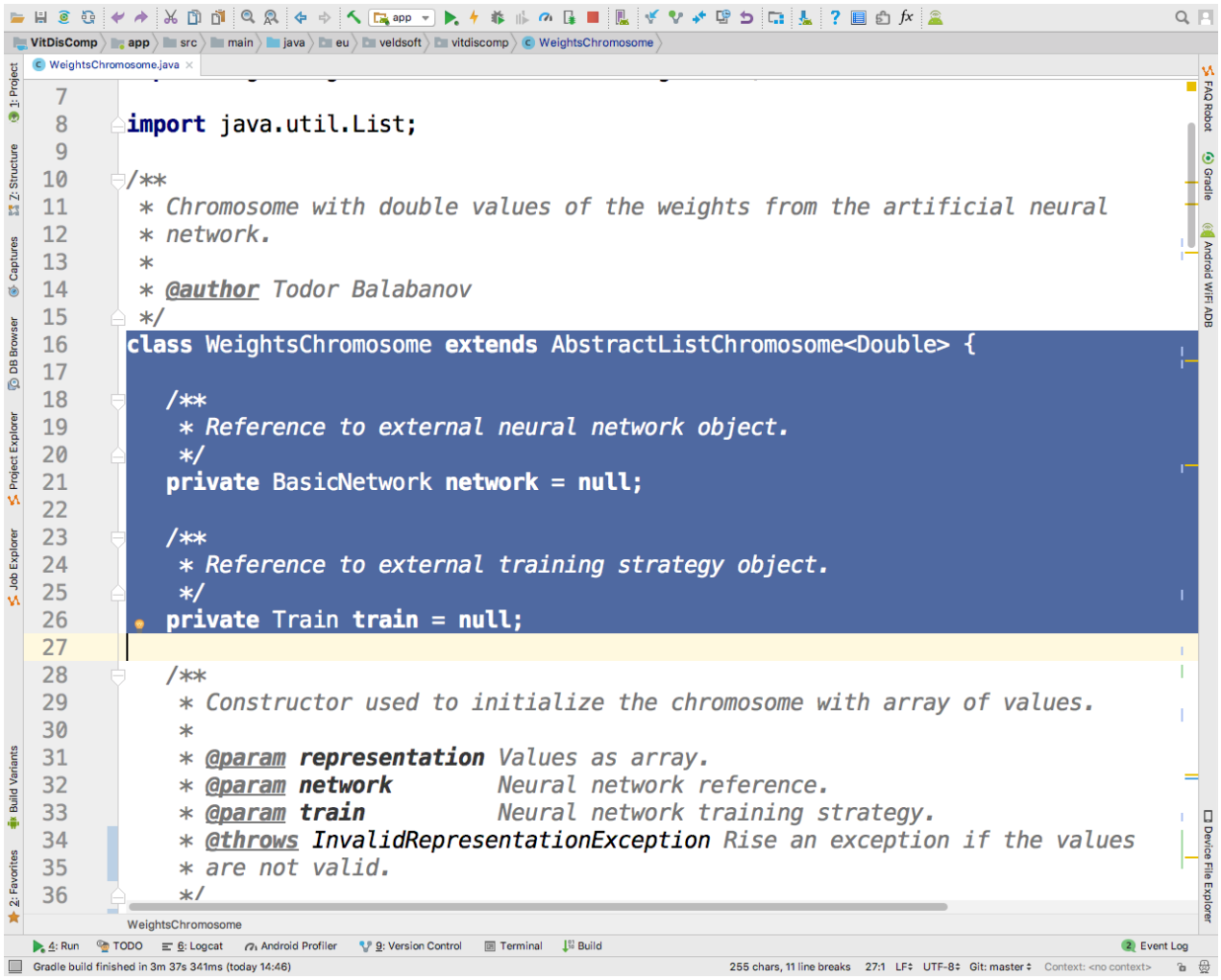
Следва изпълнение на генетичния алгоритъм (Фиг. 7.10). За кръстосване се прилага равномерно случайно кръстосване. За мутация се прилага равномерна бинарна мутация върху всички гени на хромозомата. Най-добрият индивид от оптимизираната популация се зарежда обратно в мрежата (Фиг. 7.11).



Фигура 7.11: Обновяване на теглата в мрежата

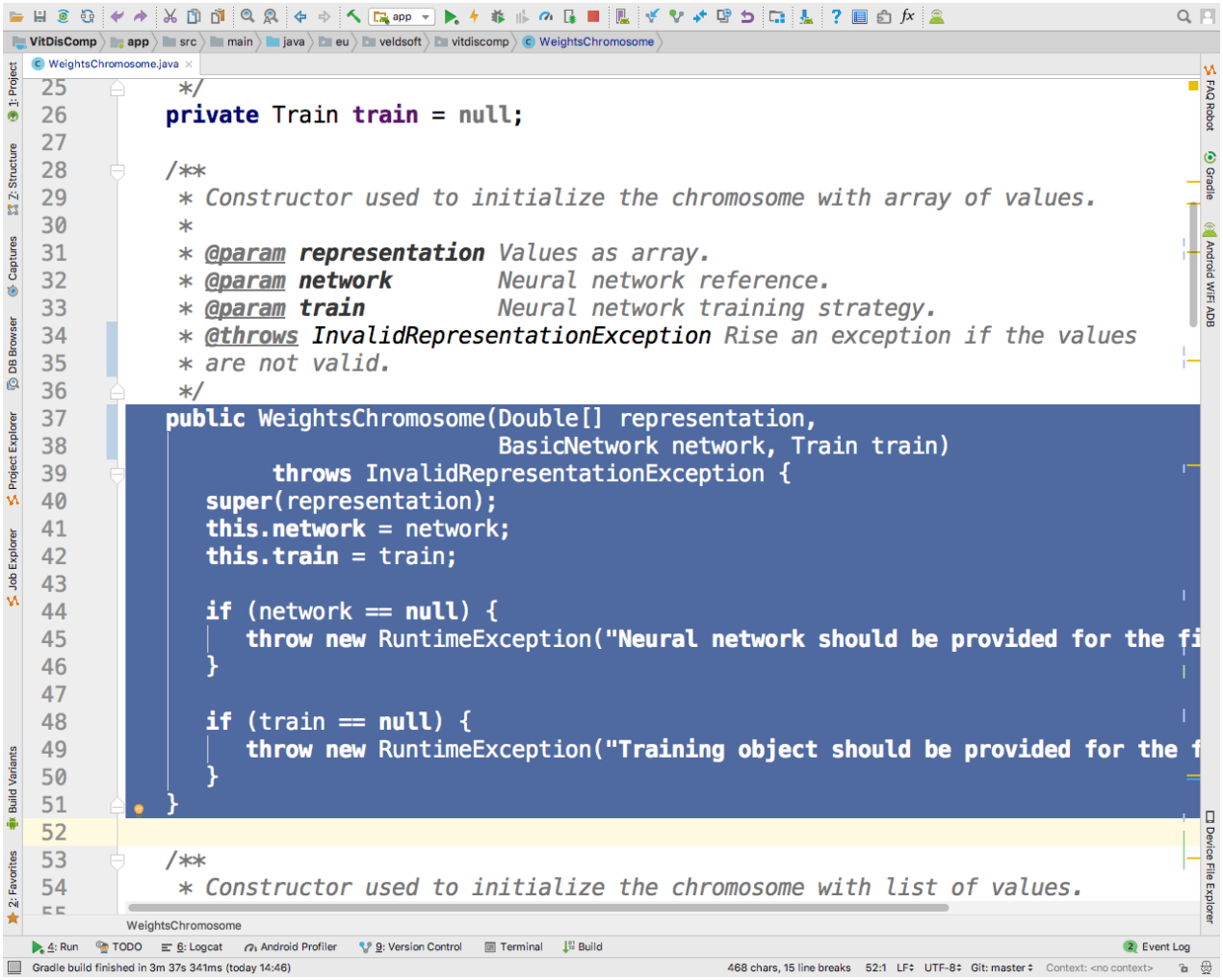
7.2.1 Кодиране на хромозомите

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.

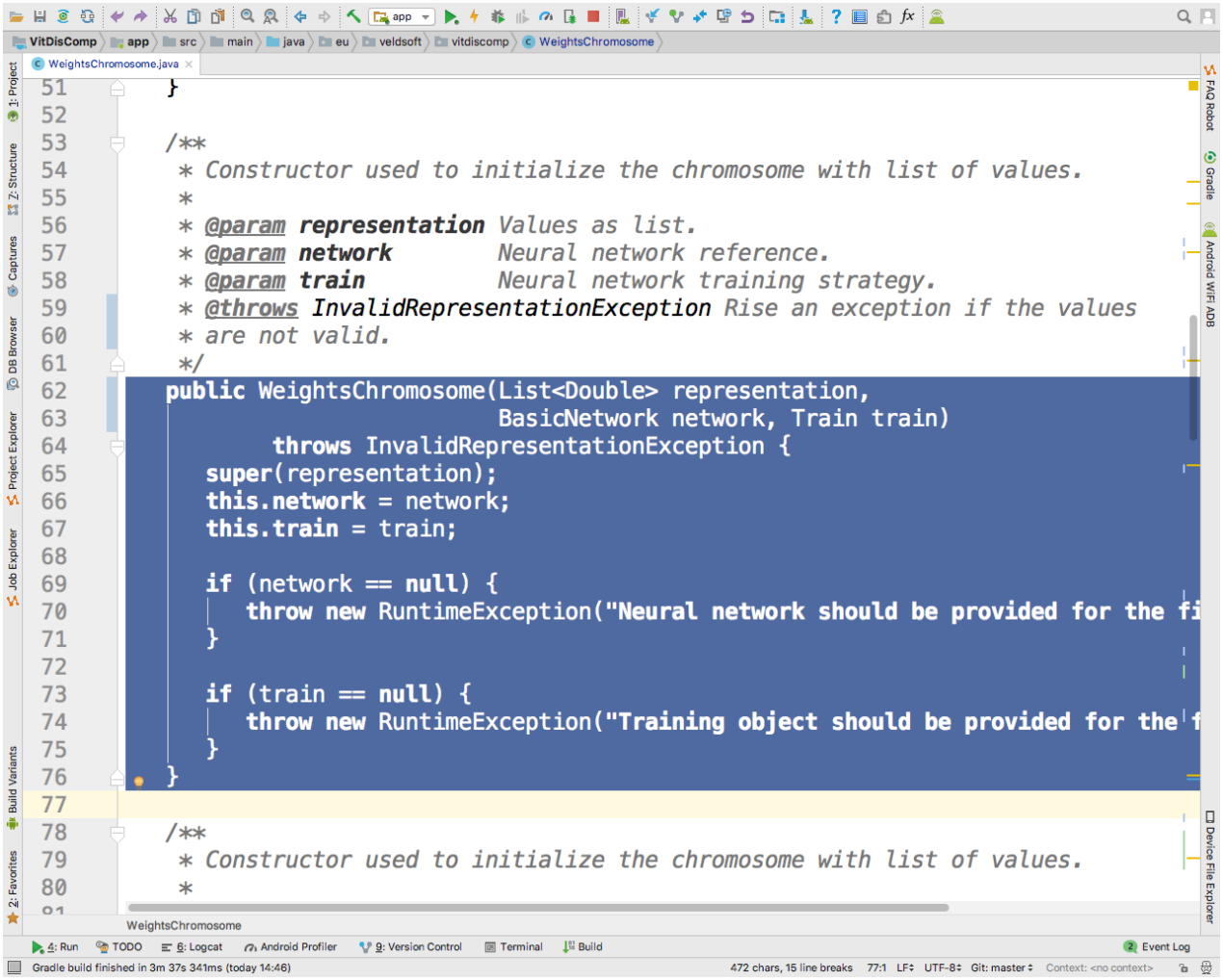


Фигура 7.12: Кодиране на теглата в хромозомите

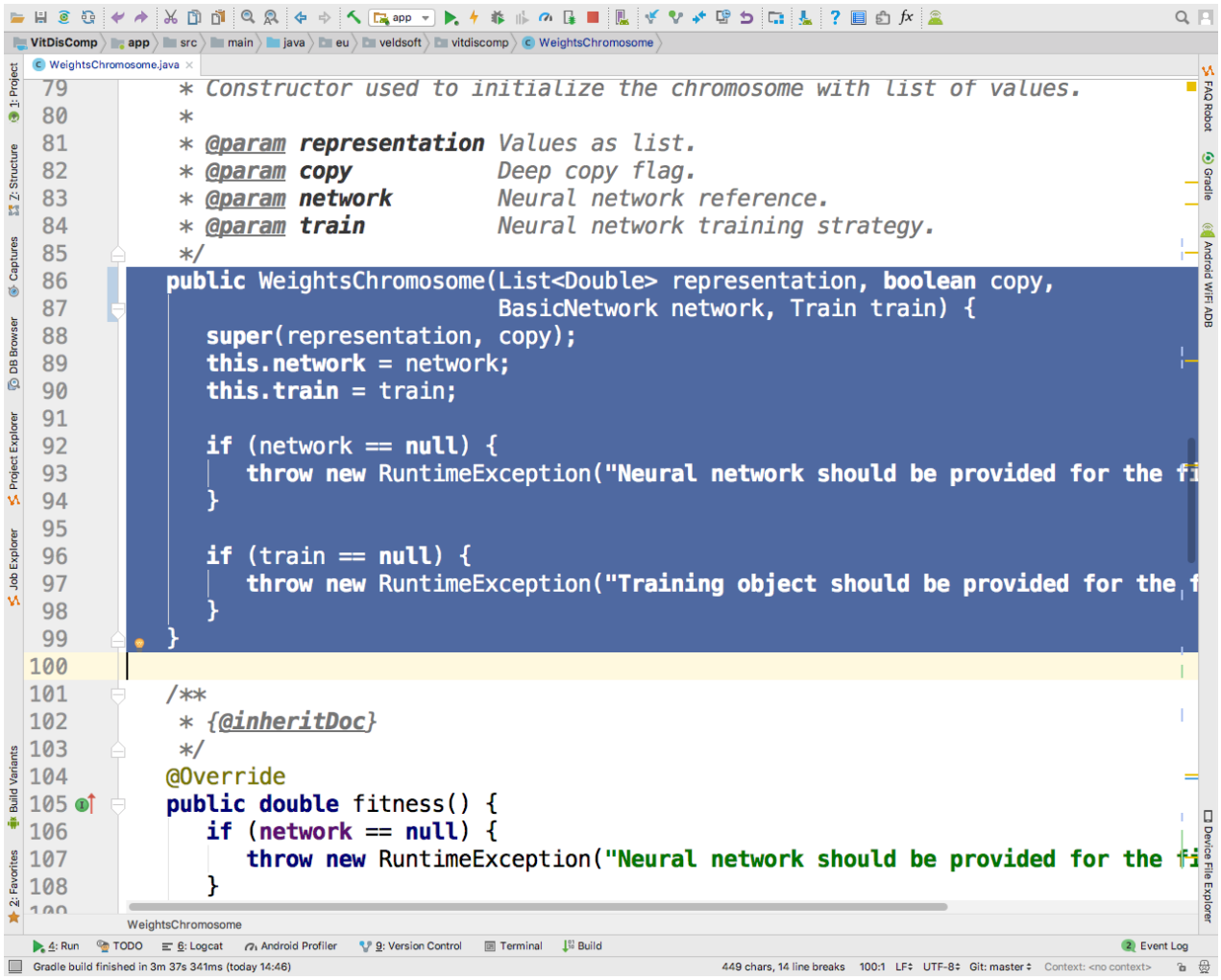
За гъвкавост при създаването на обектите хромозоми са предефинирани трите наследени конструктора (Фиг. 7.13, Фиг. 7.14, Фиг. 7.15).



Фигура 7.13: Конструиране по зададен масив

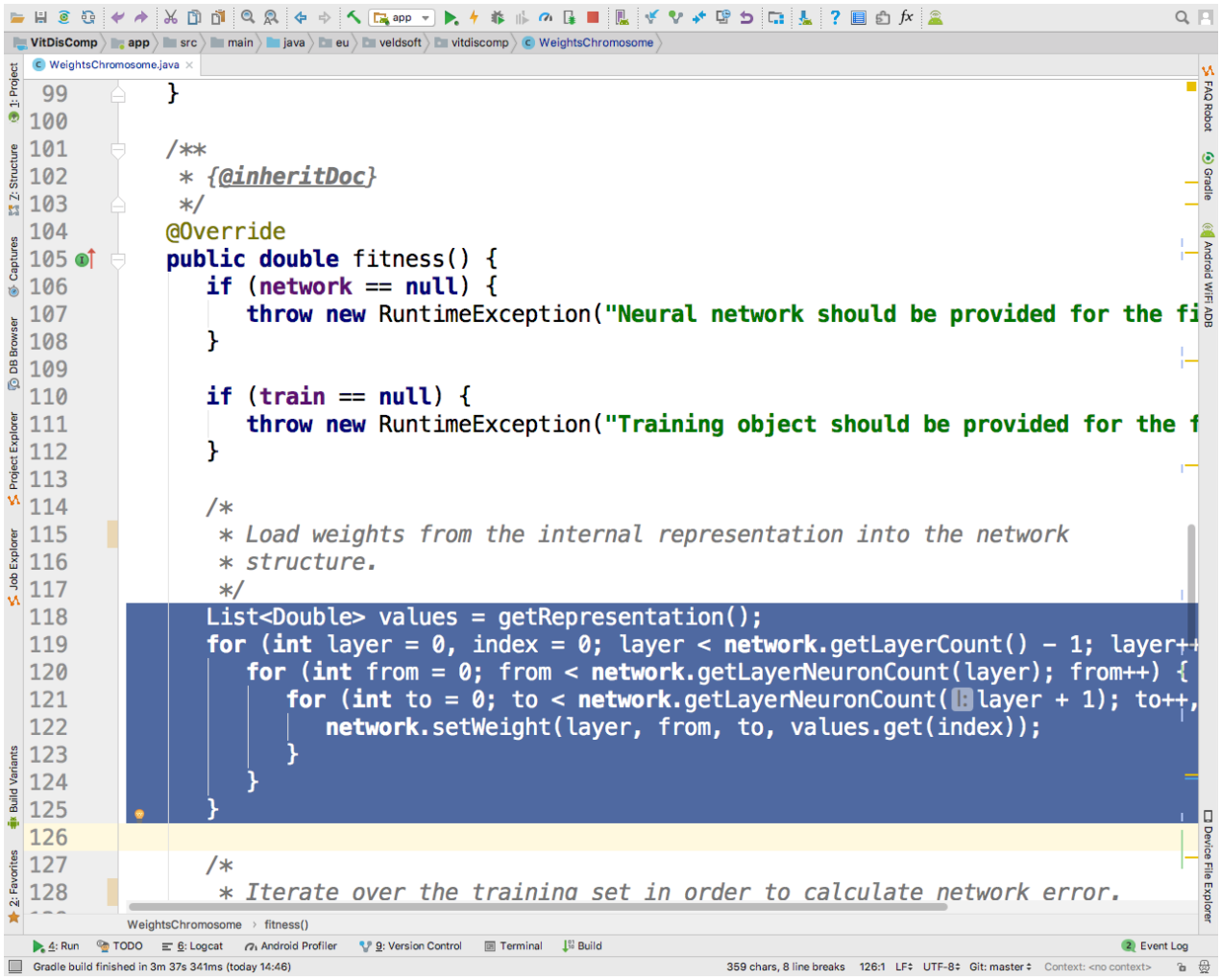


Фигура 7.14: Конструиране по зададен списък



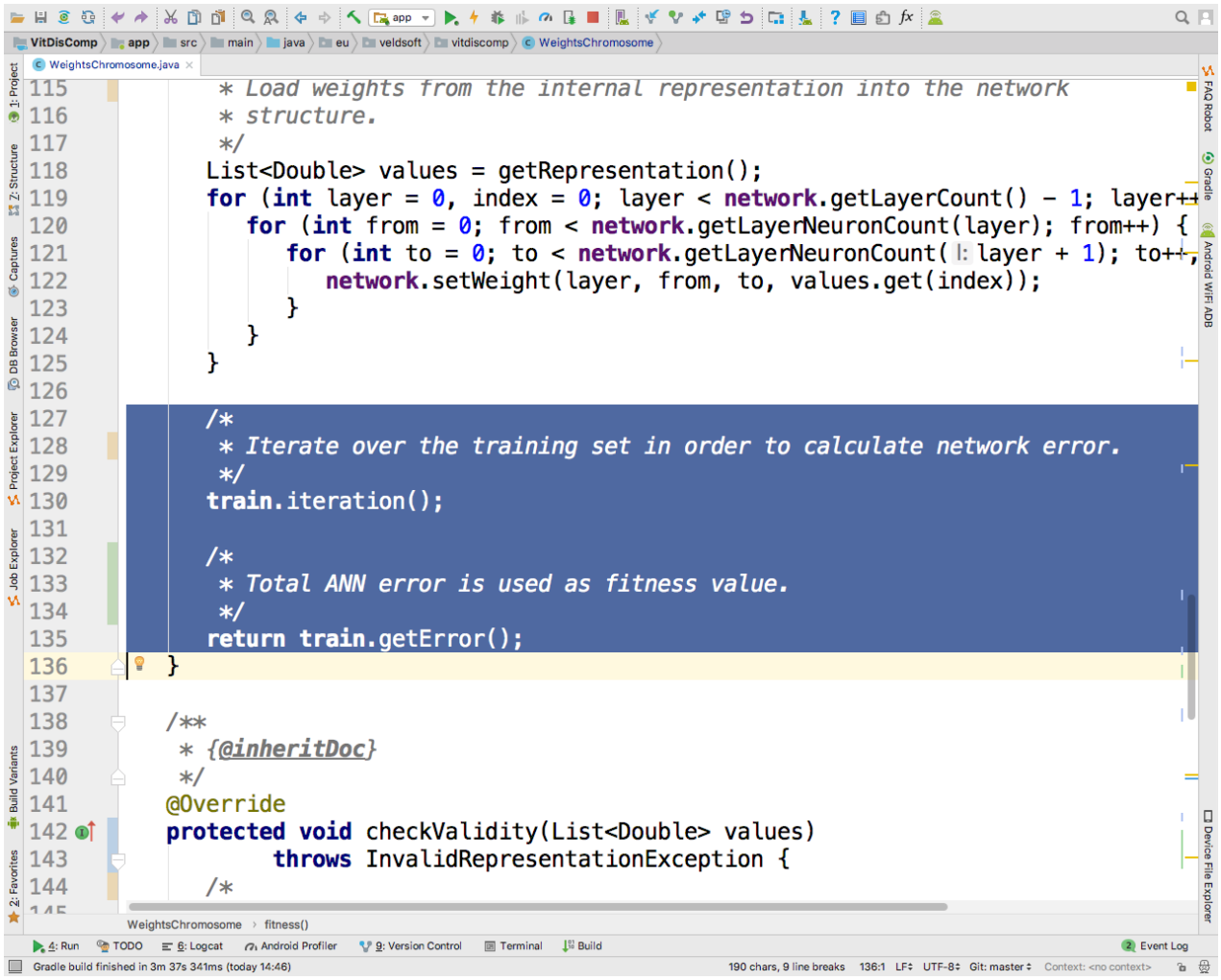
Фигура 7.15: Конструиране по зададен списък с възможност за дълбоко копиране

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



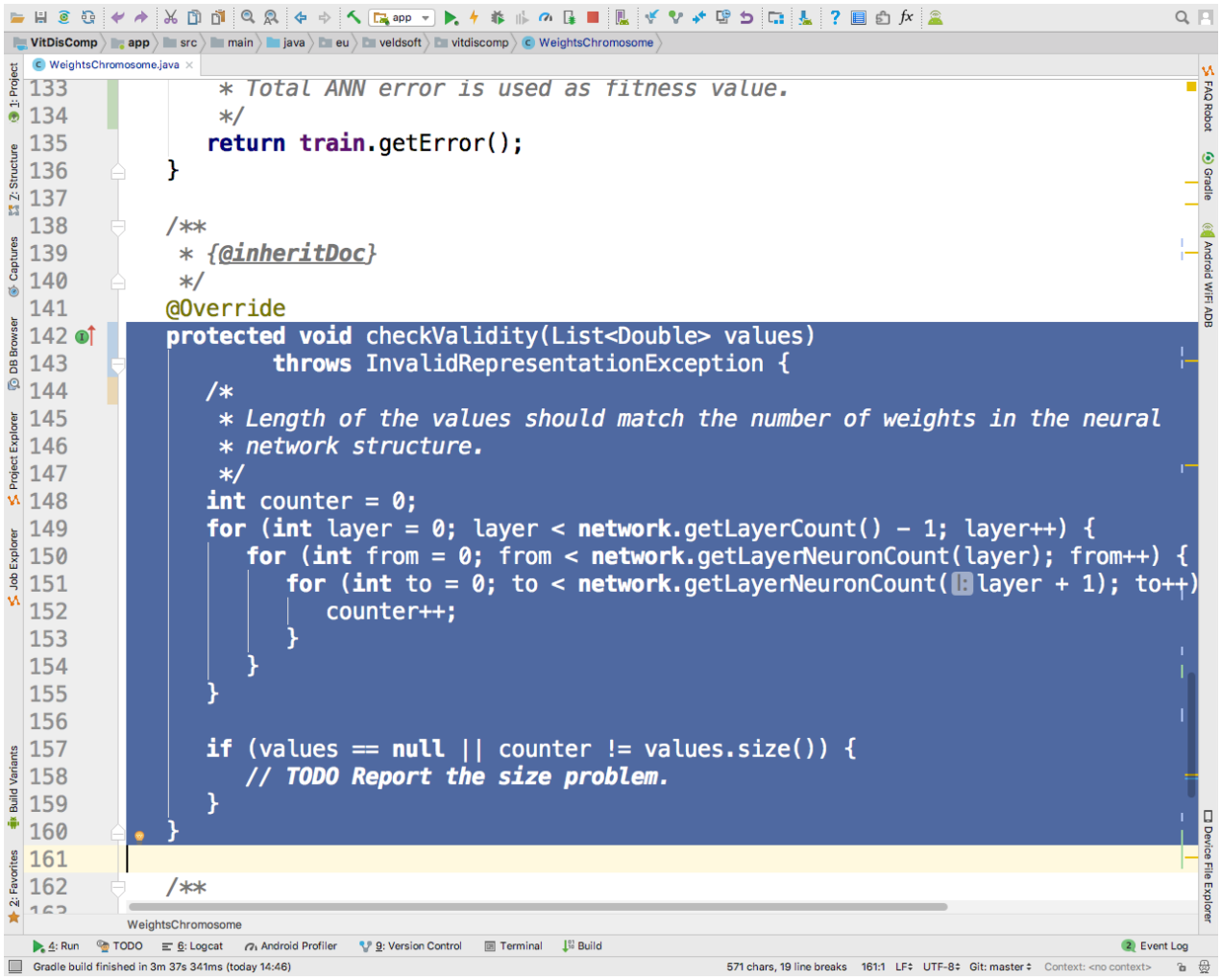
Фигура 7.16: Зареждане на стойностите като тегла в мрежата

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



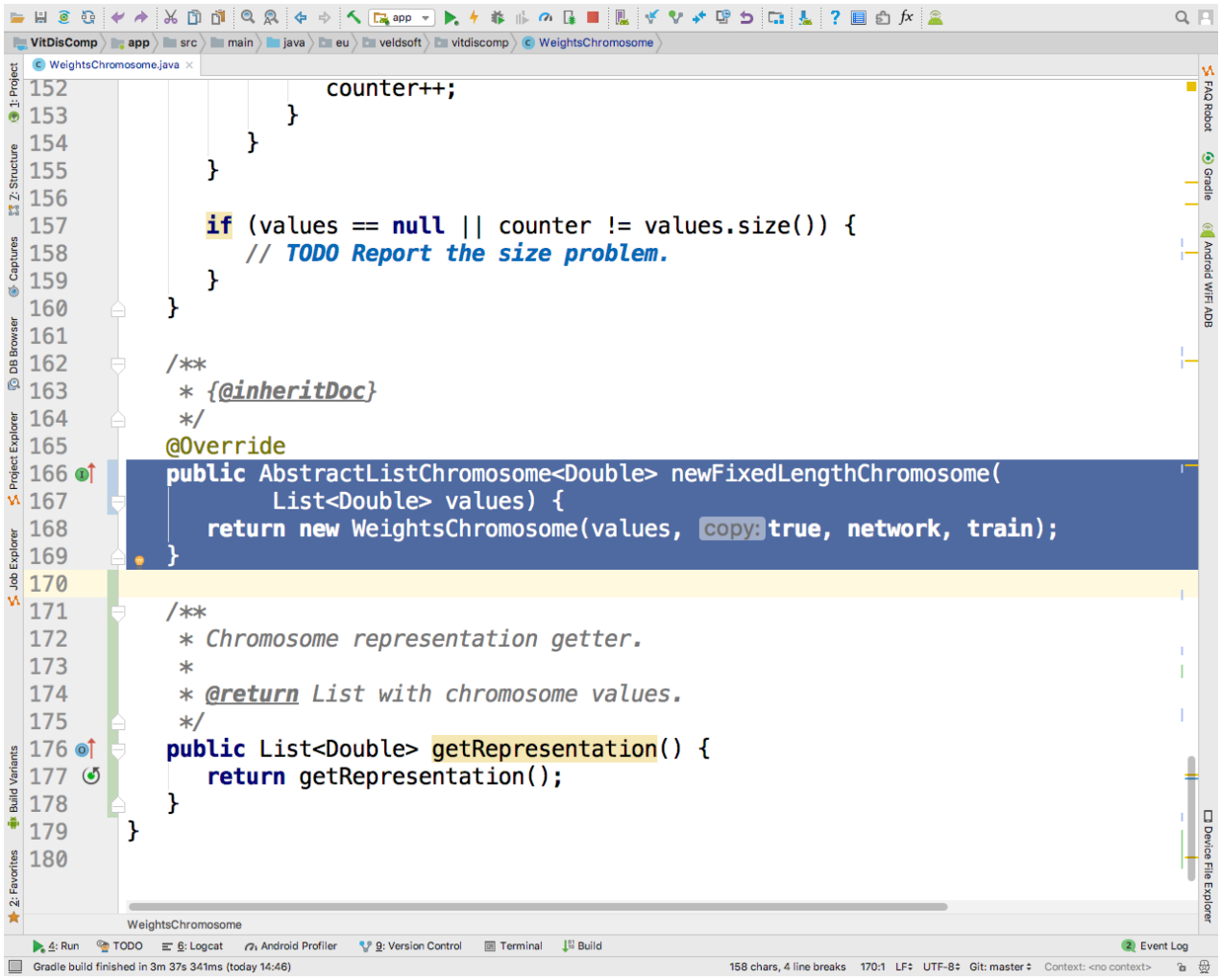
Фигура 7.17: Изпълнение на правия пас

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).



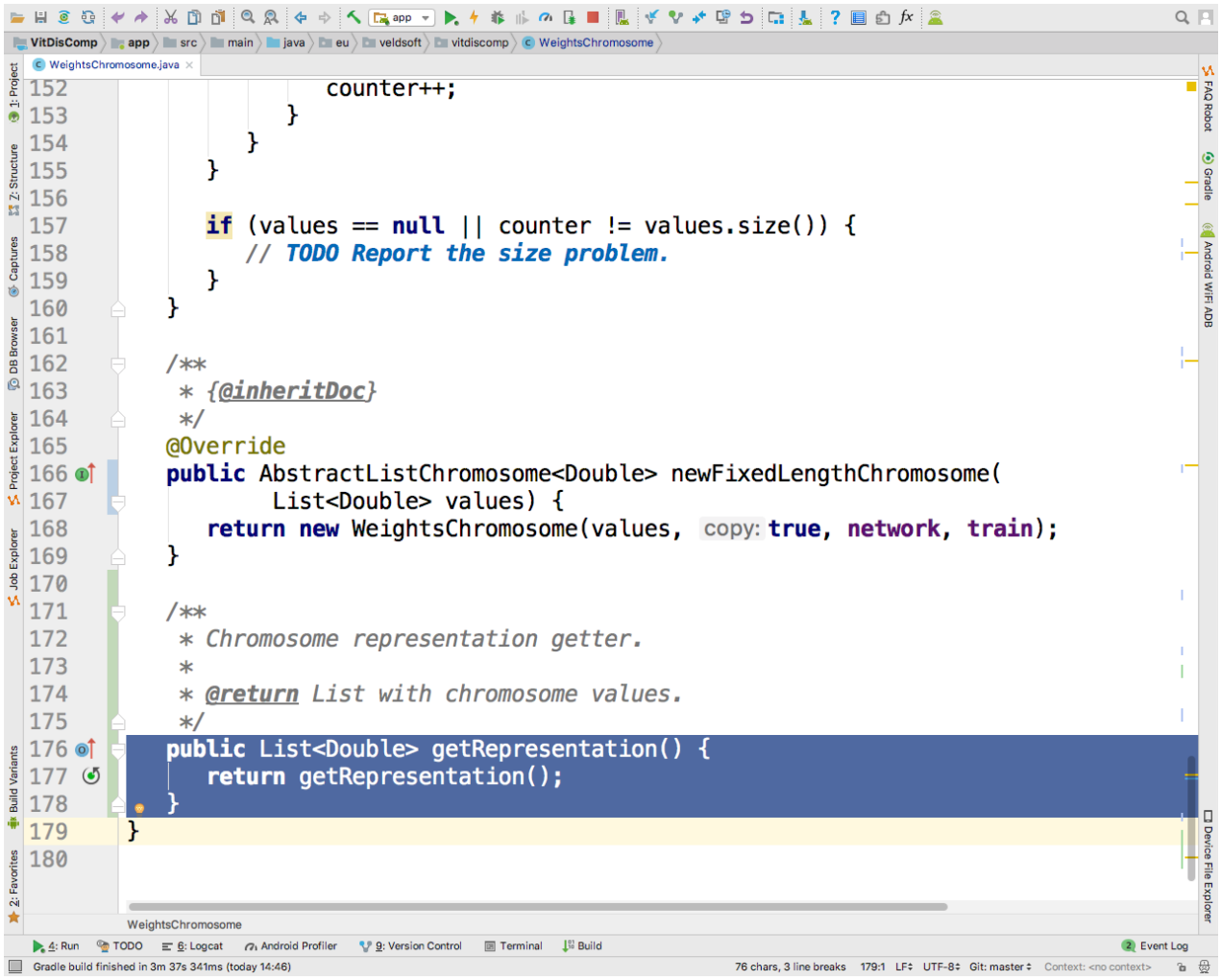
Фигура 7.18: Условие за валидност на хромозомата

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).



Фигура 7.19: Конструиране на нова хромозома с фиксирана дължина

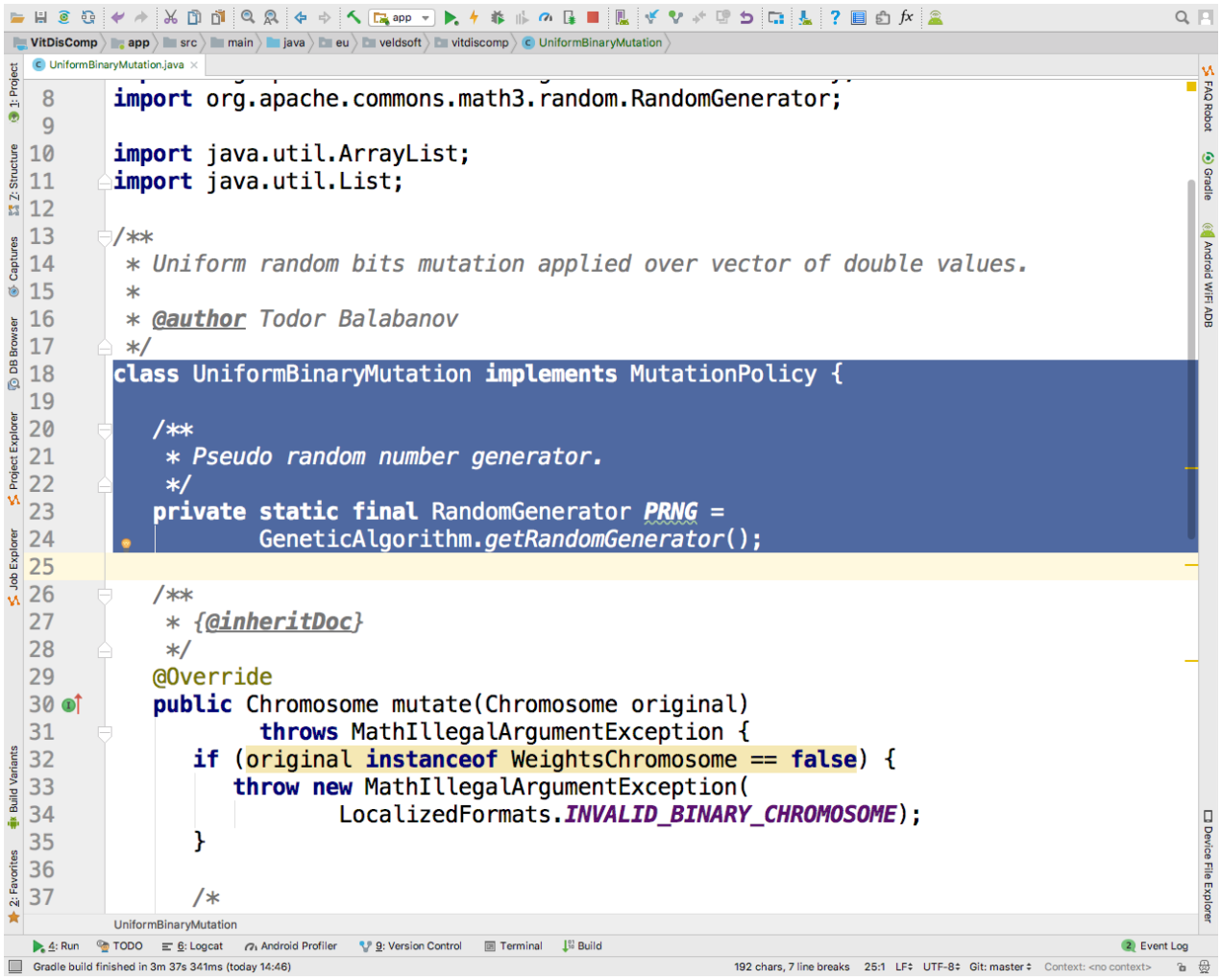
За нуждите на пресмятанията извън класа е създадена функция за извеждане на стойностите (Фиг. 7.20).



Фигура 7.20: Функция за извеждане на стойностите извън хромозомата

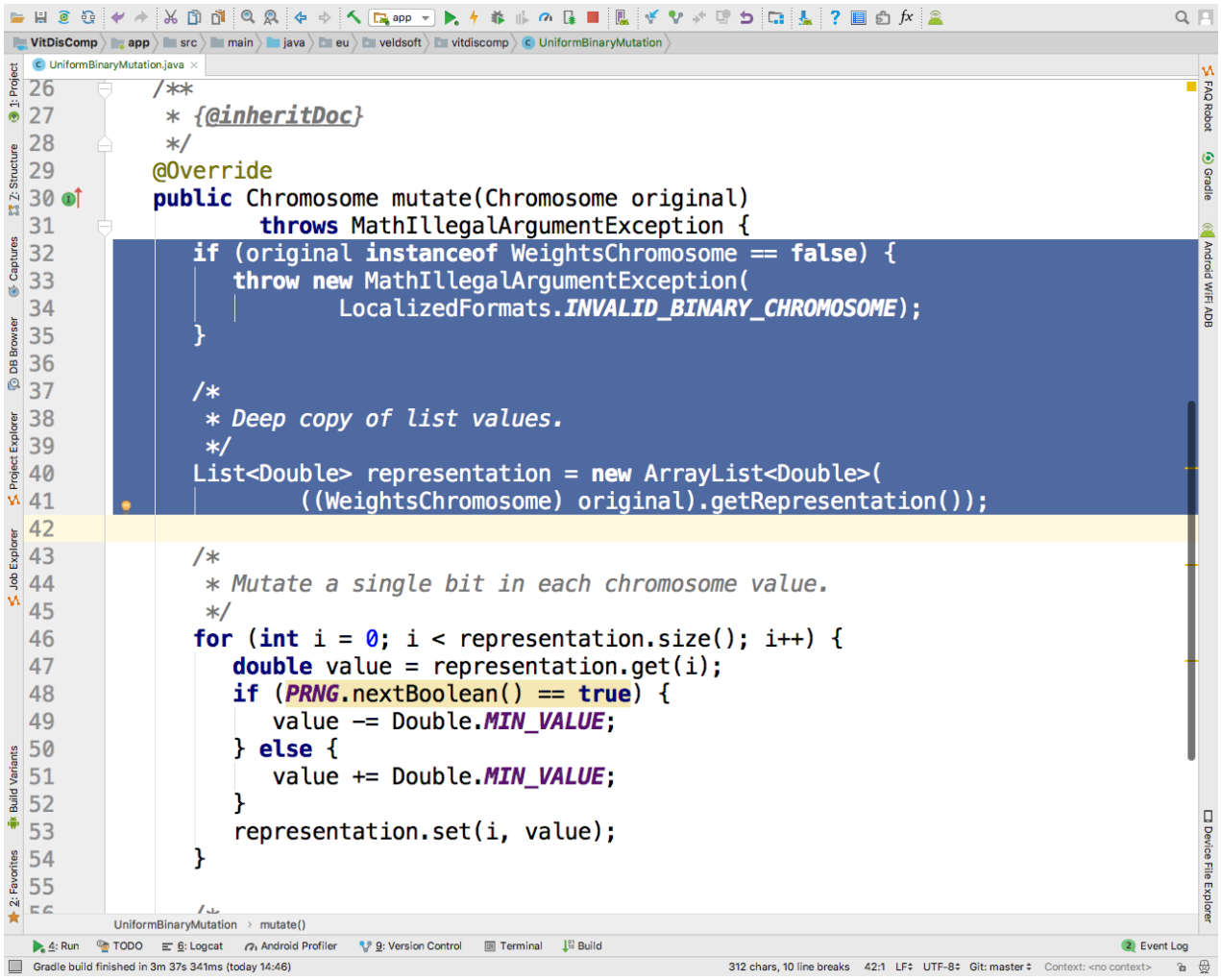
7.2.2 Случайна равномерна мутация

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).



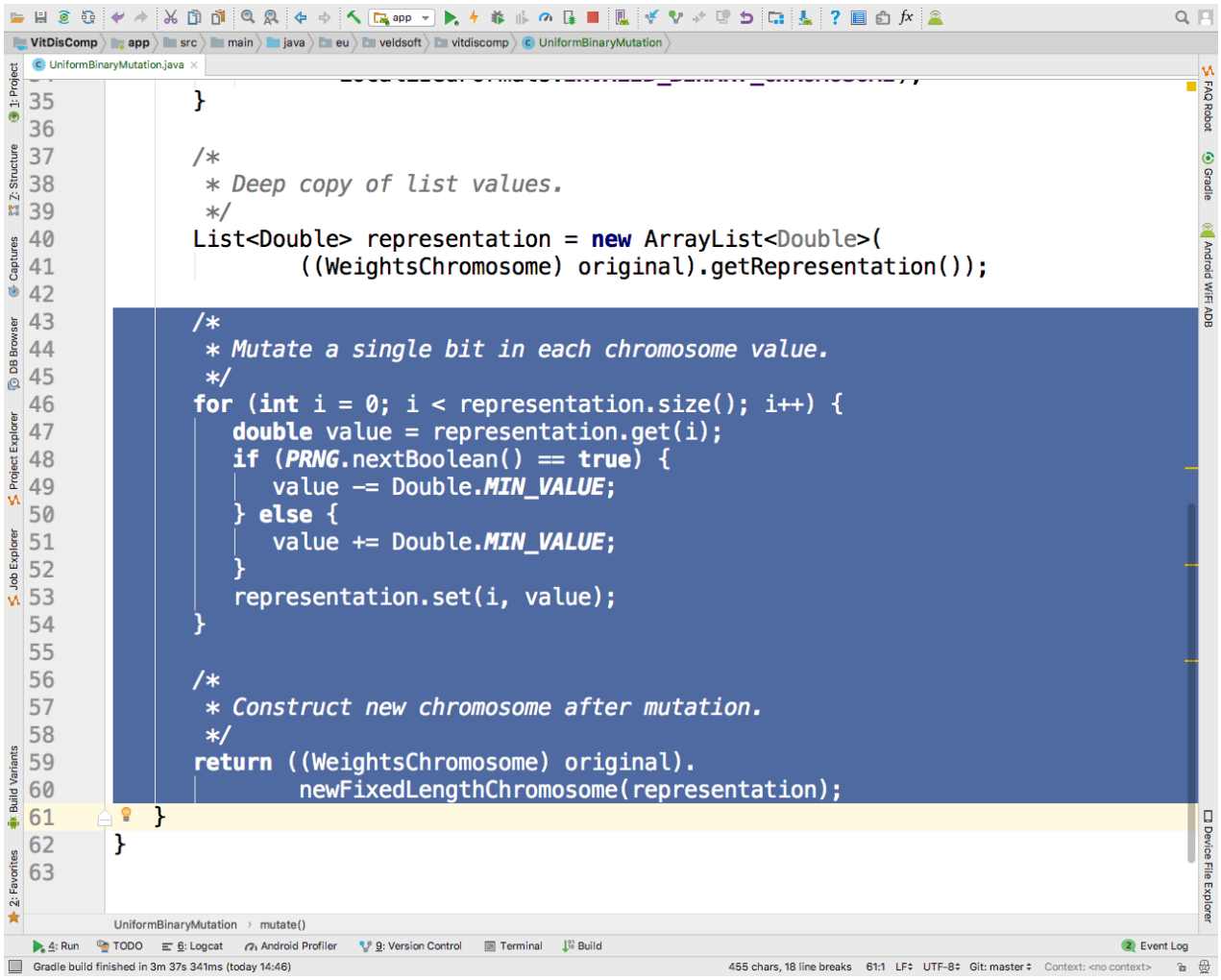
Фигура 7.21: Клас извършващ мутацията

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).



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).



Фигура 7.23: Случайна промяна на всички стойности в хромозомата

Мутацията приключва със създаване и връщане на мутирал екземпляр.

Заключение

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 capacitiy* 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

Списък на фигурите

Азбучен указател

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.

eISSN 2367-8666