Съдържание

**I. Функционално описание на системата3**

**II. Итеративен подход на разработка**4

1. Централизация4

1.1 Git4

1.2 Maven5

1.3 База Данни5

2. Децентрализация6

2.1 Maven6

2.2 База Данни 6

2.3 Сървъри7

3. **Реализация на итеративният подход за разработка** 7

3.1 Git8

3.2 Maven8

3.3 Tomcat9

3.4 Oracle база данни10

3.5 Liquibase11

**III. Системна архитектура и база данни**14

1. Приемане на заявки и Jersey14

2. Менажиране на обектите и Spring14

3. Менажиране комуникацията с базата данни и Hibernate16

4. Архитектура на базата данни19

5. Менажиране визуализацията на данните и Angular19

**IV. Особености на програмната реализация**23

1. Клас диаграми23

2. Използвани технологии, похвати и шаблони за дизайн25

2.1 Сървър и комуникация с база данни25

2.2 Визуален интерфейс42

**V. Ръководство за потребителя** 49

**VI. Заключение**53

**VII. Източници**54

**VIII. Приложение** 55

1. Сървър55

2. Визуален Интерфейс91

**I. Функционално описание на системата**

Представената система има за цел да предостави функционалност свързана с често-използваната от учебни заведения „Moodle“ система, където преподавателски екипи предоставят учебни материали, места за дискусия по даден предмет, тестове и изпити с фиксирана продължителност за оценяване на учащите и самите оценки. С цел улесняване работата на преподавателите, представената в това задание система предлага улесняване достъпа на файловете използвани от курса на предмет както и статистики за целият курс, от които един преподавателски екип има нужда и интерес да се възползва.

Предоставената системата използва вградена във всяка Мудъл система функционалност за извличане на информация от нея чрез „HTTP“ (Hypertext Transfer Protocol) заявки, записва в релационна база данни описателна за всеки файл информация, включително „URL“ адрес за изтеглянето му от самият Мудъл. Освен информация отнасяща се специфично до файловете качени в даден курс, по аналогичен начин системата извлиза и записва в базата данни статистически-значима информация като брой качени материали разделени по типове и брой дискусии отново обобщени за всеки курс. Системата обработва, сегментира и подрежда тези данни в подходящ за визуализация вид и в графичен интерфейс достъпен чрез браузър представя тези данни в таблица или стълбовидна диаграма. Освен това, системата предоставя липсващата в Мудъл функционалност да бъдат изтегледни наведнъж всички файлове за даден курс, значително спестявайки усилието да бъдат свалени ресурсите в курс с десетки или стотици файлове.

За защита на горепосочената функционалност е използвана автентикация и оторизация посредством система за вход изискваща предварително създаден в Мудъл системата не-студентски потребител, валидирайки подадените данни посредством интеграцията на двете системи.

Представената система е средство за улеснено боравене със свързана към нея работеща Мудъл система посредством събирането на данни за системата и предоставянето им в удобен за потребителя вид. Системата се явява решение на проблема, че наличната информация е неизползвана от крайните потребители не поради липса на достъп до нея, а поради непрактичноста от това действие заложена в трудностите свързани с достъпването, четенето и боравенето с информацията.

**II. Итеративен подход на разработка**

Представената система използва принципи за разработка на софтуер, които предразполагат и улесняват използването на модерни подходходи за гъвкавото разработването на софтуерни продукти. Изполвзани са Agile и Continuous Integration методологиите, които изискват постепенното надграждане и тестване на малки по обем на работа функционалности. По този начин се подобрява качеството на проекта, както и правилното оценяване за времетрайноста за изработването на различните функционалности, в допълнение към възможността за лесното добавяне на допълнителни функционалности и промяна в дизайна на приложението, докато то се разработва и след като планираните функционалности са завършени.

За постигане на изброените цели е необходимо полагането на стабилни основи за гъвкавост на работата позволяваща едновременната работа по множество на брой независими една от друга функционалности. Това съответно позволява и работа от скалируем брой хора. От това следва, че проекта трябва да се стреми да позволява работата и проверка на функционалността на неограничен брой независими функционалности и еквивалентен брой разработчици и докато свежда породените от това проблеми до минимум. Взимайки това в предвид е необходимо да бъде разгледана цялостната архитектура и интеграциите между компонентите на архитектурата преди да бъде разгледани в детайли индивидуалнте компоненти на архитектурата.

**1. Централизация**

За да бъде предоставена възможността неограничен брой разработчици да работят едновременно по един и същи проект е необходимо централизирането на основни за проекта данни, библиотеки и версии на използван софтуер. Същевременно, по този начин са сведени до минимум разликите не само между средите на всеки от разработчиците, но и между среда за разработка, за тестванене и за употреба от крайният потребител.

**1.1 Git**

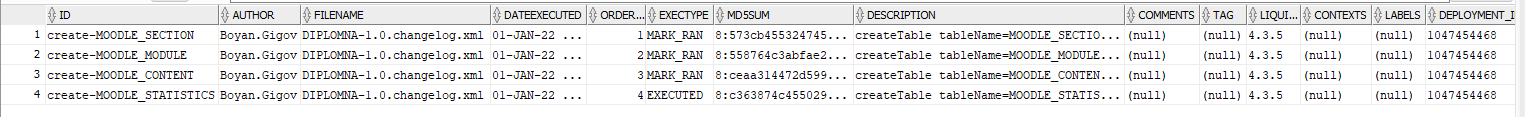
Един от най-популярните софтуери за разработка на софтуер в днешно време и необходим за практически всеки програмист, „Git“ предоставя отличителна възможност за превключване между различни версии на проекта и е предназначен за работа с един главен клон (branch) за централна версия и множество клонове, където да се извършват и тестват промени преди да се добавят към главният клон използван като кандидат версия за версията на проекта, която ще достигне крайният потребител.

**1.2 Maven**

Централизацията на библиотеки в този проект е постигнато чрез популярният „Maven“ софтуер, предназначен за менажиране на софтуер чрез автоматизиране на изграждането на проекта в приложение (build). Това автоматично изграждане се извършва с използването на зададени в специални пом файлове (pom.xml) библиотеки, от които зависи проекта, заедно с техните версии, в допълнение към написаният от разработчиците код. Приложението изградено от „Maven“ може да е във вид на специално пригодените за „Tomcat“ сървъри *.*war файлове.

**1.3 База данни**

Подходът използван за версиите на библиотеките е подходящ и при използването на бази данни. Изграждането на централизирана база данни позволява множество разработчици да използват използвани от всички стандартизирани настройки, тестови данни, схема и релации. В допълнение, промените по централизираната база данни се извършват не чрез трудно-проследими или непроследими промени извършени чрез заявки от всеки индивидуален разработчик, а чрез автоматизираният „Liquibase framework“, който изготвя заявките от записани във файл желани промени. Освен че фреймуърка предоставя нагледен архив от всички извършени заявки към базата данни (Фиг. 1), „Liquibase“ позволява проследяването на поредноста на изпълнените заявки, предоставя възможност за проверка дали наистина са били изпълнени и предотвратява проблеми предизвикани от едновременното изпълнение на заявки от множество източници. Друга важна функционалност предоставена от фреймуърка е възможността за прехвърляне на таблици, изгледи, индекси, курсори и т.н. с всичките им ограничения и релации от една база данни на друга. Тази функционалност е благодарение на логването на фреймуърка служещо като информация за подредбата на изпълнените промени и може да бъде използвано за генерирането на промените в точният им ред на изпълнение. Подредбата от изпълнените промени може да бъде разглеждана като подредба от всички състояния на базата данни. Освен пълно копиране текущото състояние на базата данни, състоянието на базата данни съответно може лесно да бъде възстановено до точно определена промяна.

Фиг. 1 Поредица от записаните изпълнени промени в базата данни

**2. Децентрализация**

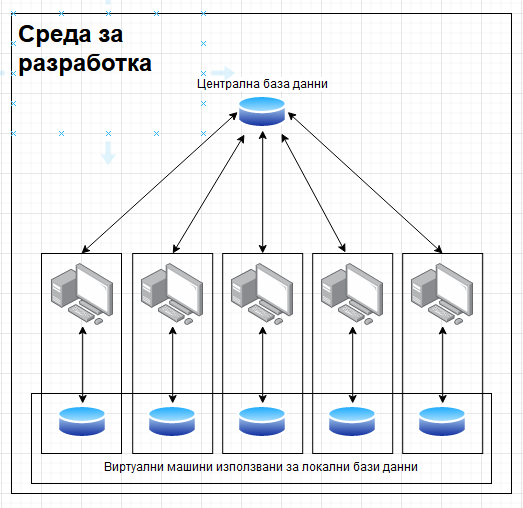
Централизацията е необходима за уеднаквяване средите на множество разработчици, но понеже една среда и кода на нейното приложение и многото софтуерни версии използвани в него не са статични се налага предоставянето на децентрализирана възможност за работа на индивидуален разработчик.

**2.1 Maven**

Децентрализацията на процеса - използването, променянето и тестването на кода от индивид с локалната му среда за разработка - се осъществява лесно чрез променянето на пом (pom.xml) файловете, което позволява и на всеки индивид да промени версиите на библиотеките и да провери и при нужда разреши проблемите предизвикани от това преди да промени централизираната версия на библиотеките и промените на индивида да засегнат други среди на проекта. Чрез изграждането на приложението в *.war* файлове се предоставя възможността на разработчика да провери не само локалната си версия, а и резултата от нея, който би се използвал от сървъра предназначен за приложението, преди разработчика да промени централизираната версия на коя да е библиотека.

**2.2 База данни**

За да бъде предоставена възможността всеки индивид да може да работи на изцяло локална среда, се предоставя и локална база данни. Това е постигнато чрез използването на виртуална машина на същата машина, която използва разработчика и използването на тази виртуална машина като база данни (Фиг. 2).

  
Фиг. 2 Диаграма на среда за разработка за множество разработчици  
използващи централна база данни и локални бази данни

**2.3 Сървъри**

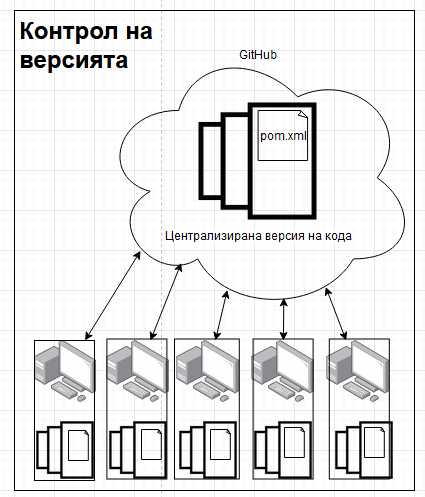
За пълноценна проверка на промените от един разработчик е необходимо да се провери, че приложението функционира по предназначеният начин. За тази цел, изграденото приложение се тества от личен за разработчика сървър използващ изградените от Мейвън .*war* файлове. Така разработчика може да изолира функционирането на собствените си промени чрез използването на вече функционираща версия на проекта и в случай на проблем да проследи ясно причините им в логовете на локалният сървър.

**3. Реализация на итеративният подход за разработка**

Използвайки технологиите и основавайки се на принципите и целите споменати по-горе позволява реализацията на итеративен процес на работа, който може да бъде лесно скалиран с броя разработчици и е подходящ за използването на модерните гъвкави модели за работа, като „Agile“ методологията.

**3.1. Git**

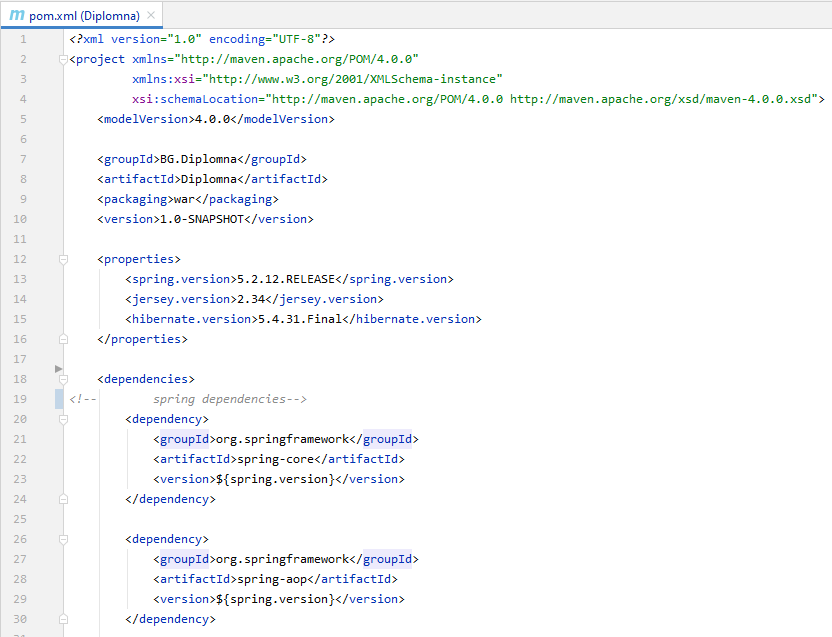
В проекта е използван „Git“ за съхранение единствено на функционални версии от проекта на централизирано място от което проекта се споделя с други лица докато междувременно разработчика локално разработва допълнителни или променя текущите функционалности, всички от които ще бъдат част от следващата версия на проекта. Сред тези промени се включват и промени по версиите на използвани библиотеки, включени в пом файлове. След завършването на нова версия от проекта, чрез „Git“ се изпраща новото състояние на проекта към централизираното хранилище (Фиг. 3).



Фиг. 3 Kонтрола на версиите на файловете в проекта,   
включително пом файлове, чрез „GitHub“

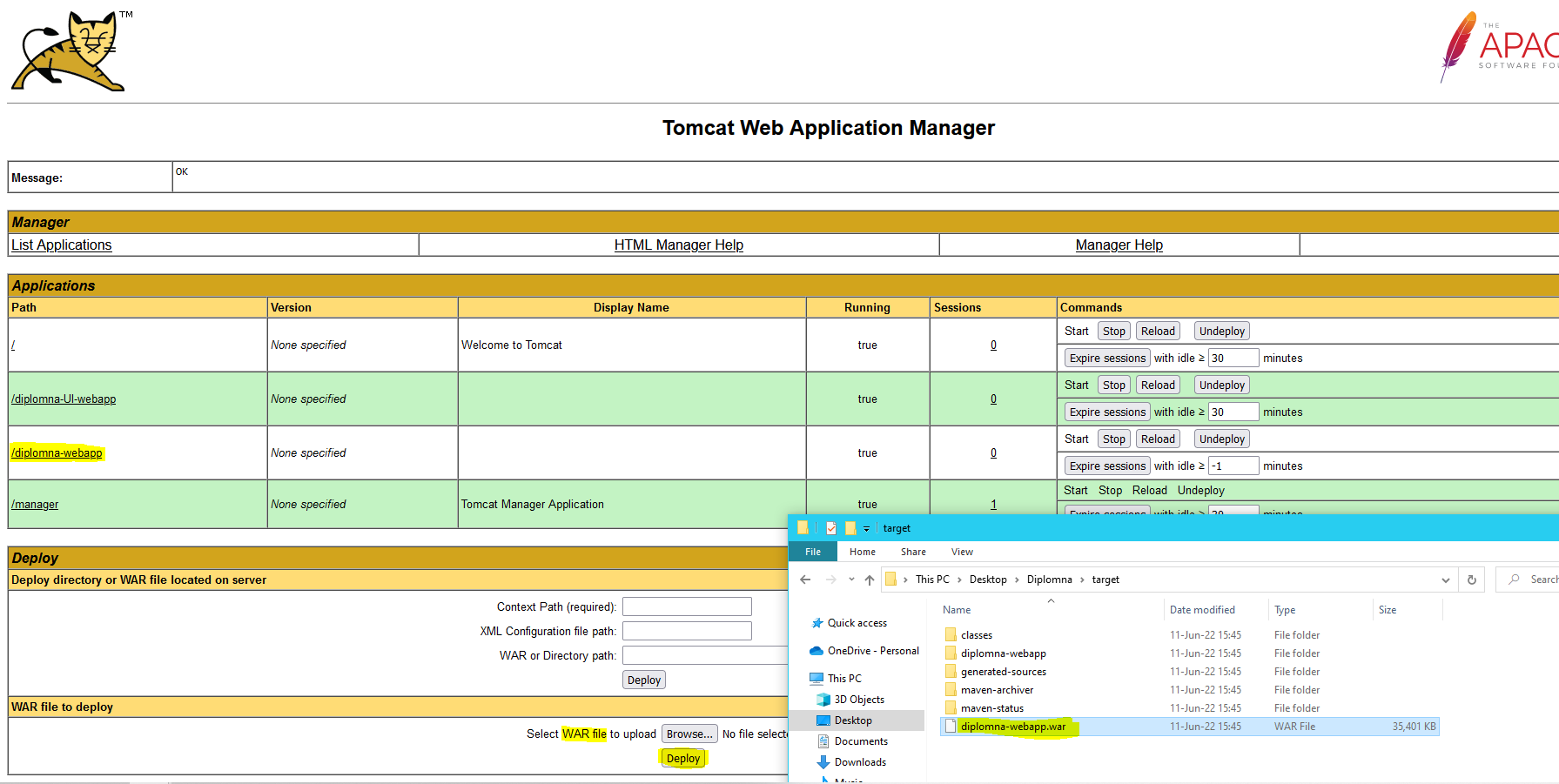
**3.2. Maven**

След изготвянето на версия на проекта кандидат за функционална версия или вече изготвена функционална версия, в този проект кодът се превръща в използваем от „tomcat“ сървър архив с .war разширение на файла. Това се постига чрез използване на предоставеният от операционната система терминал, отваряне на папката на проекта и използване на мейвън команди като “mvn clean install”, която проверява че има само 1 версия за всяка библиотека в проекта, сваля указаните в пом файла (Фиг. 4) библиотеки със съответните версии през интернет, съхранява ги на машината за в бъдеще, компилира кода от проекта и след това библиотеките, кода и допълнителните към кода ресурси се пакетират в .war файл.

Фиг. 4 – .pom файл с две от библиотеките,  
 от които зависи функционирането на приложението

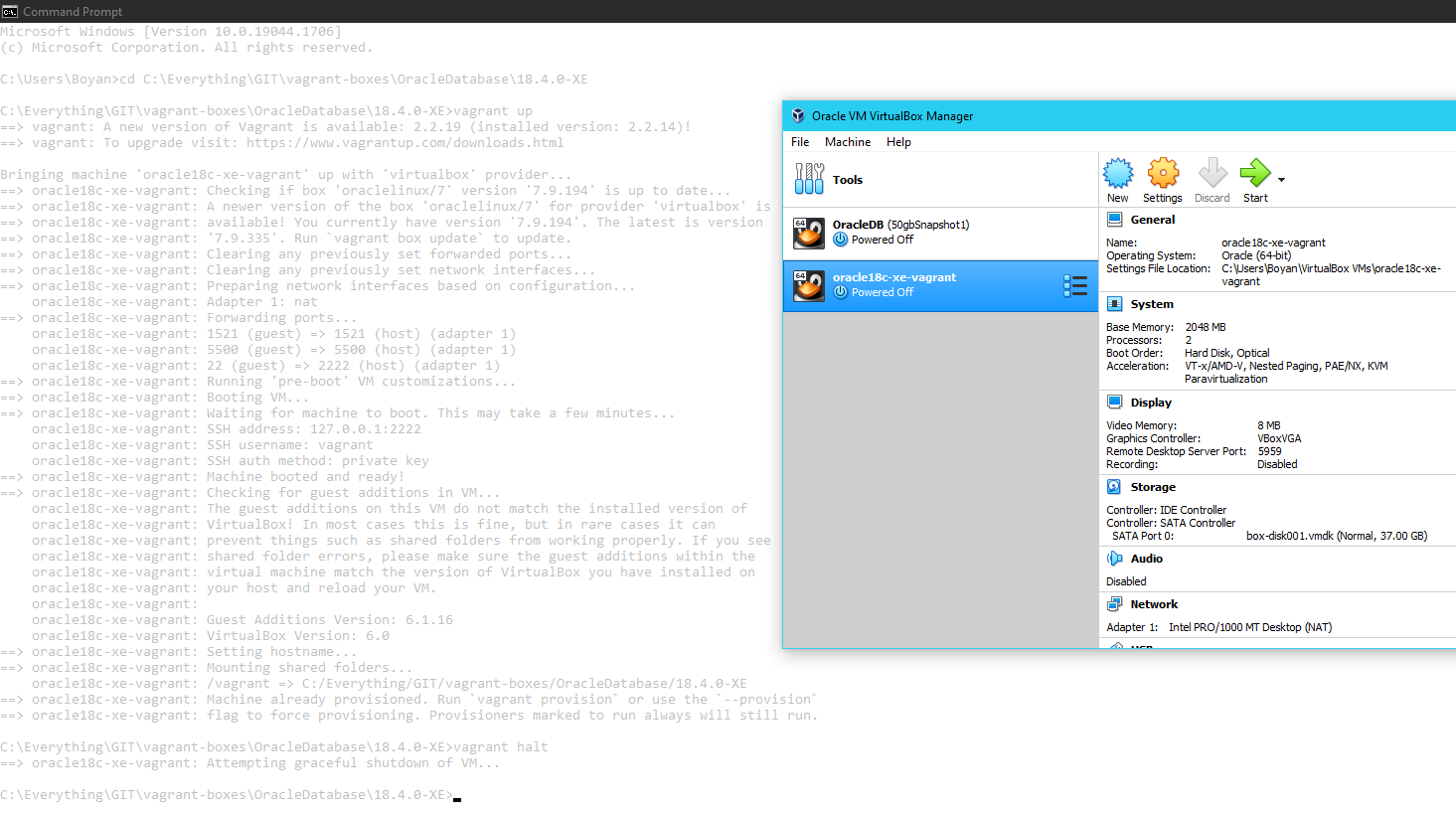
**3.3. Tomcat**

Използвайки мениджъра на „Apache“ за „Tomcat“ сървъри, при проверка на кандидат за функционална версия разработчика посочва предварително изготвеният .war архив и приложението което този файл представлява се зарежда от Томкат и може да бъде достъпено през браузъра. По този начин разработчика борави с локален сървър и може да провери поведението на приложението, без това да влияе на други среди ако използва локална база данни. В случай, че вече е заредена друга версия на приложението е необходимо първо тя да бъде премахната посредством мениджъра (Фиг. 5).

  
Фиг. 5 – .war файлове, локален „Tomcat“ сървър

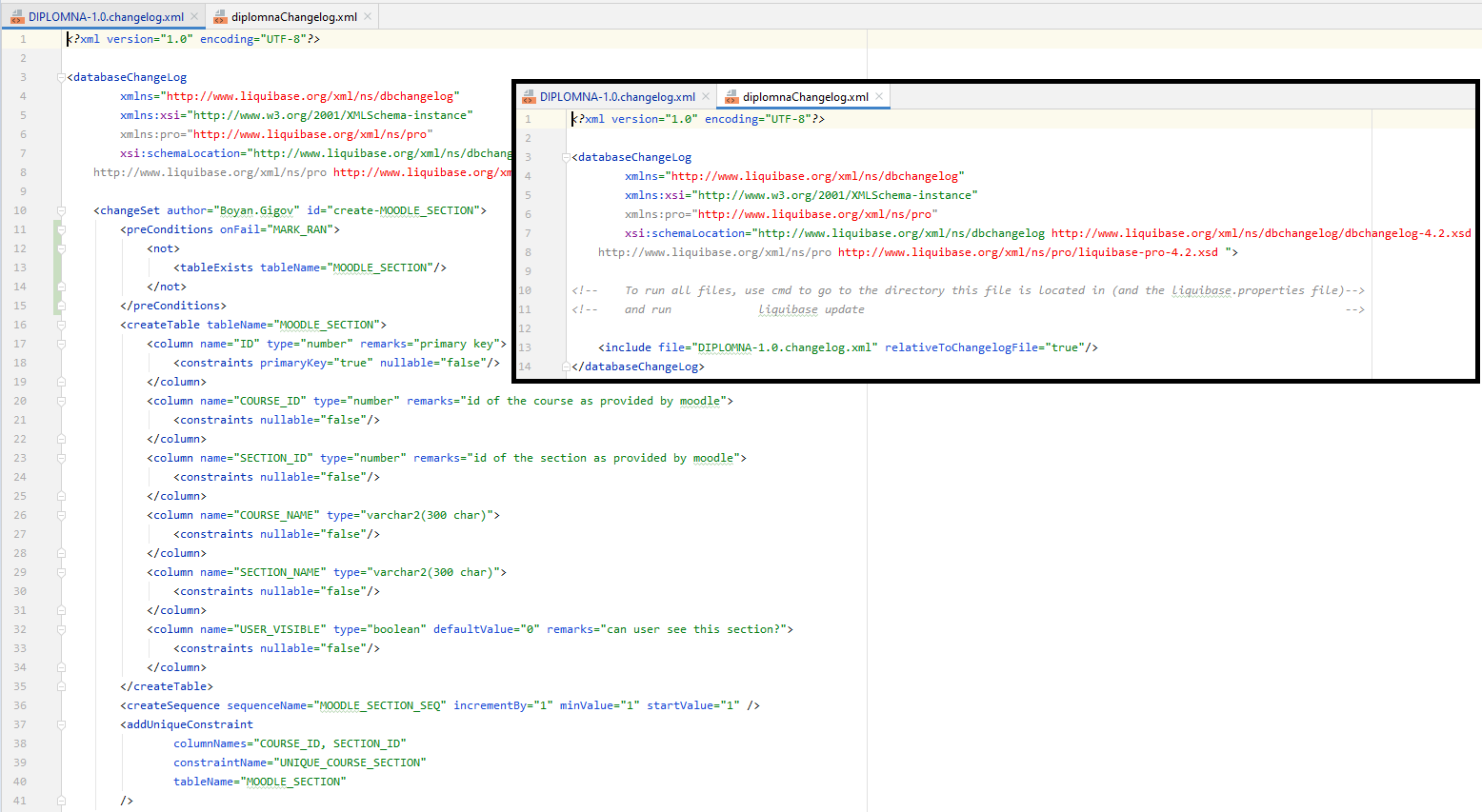
**3.4. Oracle база данни**

В днешно време базите данни често са на отделни машини от сървърите, върху които работи приложението. Съответно, за да се приближи максимално до работни условия на крайният проект разработчика има нужда да работи с база данни, с която се комуникира по същият начин като с външна база данни. За тази цел може в действителност да се използва външна за машината база данни, но в случай на локална база данни това се осъществява чрез използването на виртуална машина на използваната от разработчика машина. По същество, виртуалната машина се осъществява чрез заделяне част от дългосрочната памет на машината и използването на софтуер за едновременното й опериране от друга операционна система. За комуникация се използва същият метод като за външна система и по този начин повечето разлики между комуникацията с локална и външна бази данни се премахват. За базата данни е използван „Vagrant Box“, който съхранява състоянието на базата данни и осигурява лесното й използване и поддръжка. За виртуалната машина се използва „VirtualBox“, с който „Vagrant box“ e създаден да бъде съвместим. При нужда от достъп до локалната база данни се отваря конзолата в директорията на вече подготвената база данни и се стартира с „vagrant up” и когато вече не е необходима се изключва с „vagrant halt” (Фиг. 6).

  
Фиг. 6 „Vagrant box“ използващ „VirtualBox“

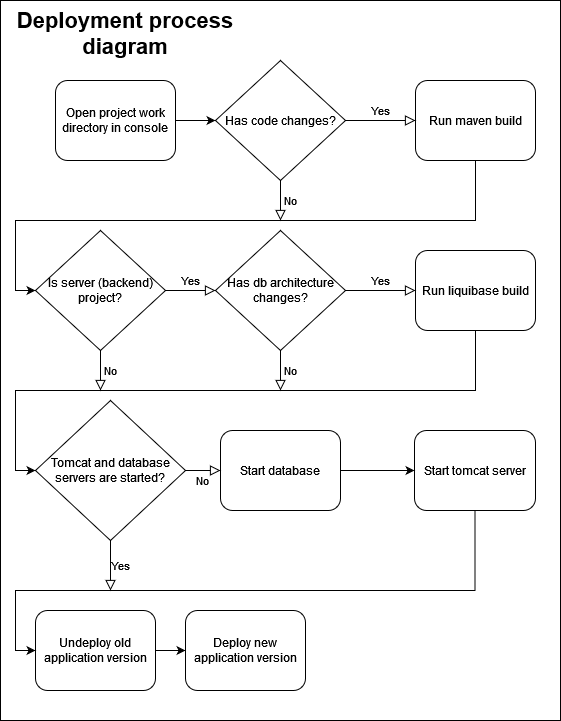
**3.5. Liquibase**

Използвайки „Liquibase“ се гарантира еднаквостта на изпълнените промени по базата данни, както и реда на тяхното изпълнение. За предотвратяване на грешки, във всеки „changeSet“ таг с промени се използва и „precondition“ тага, който позволява програмно и автоматизирано да изпълним промените само при желани условия или да прекратим изпълнението на обновленията при несъответствие с желаните условия. За изпълнение на обновленията чрез конзолата се достъпва директорията на „Liquibase“ съдържаща .xml файла с промените и се изпълнява „liquibase update“. За подобрен контрол над обновленията и версиите, както и за улеснено четене и логическо разделение на промени, за всеки набор от промени на проекта се използва отделен .xml файл и командата се изпълняват върху централен .xml файл, към който чрез „include“ таг се добавят всички други .xml файлове, чиито обновления е желано да се изпълнят (Фиг. 7).



Фиг. 7 Изготвянето на 1.0 версията на базата данни и добавянето на файла съдържащ всички от тях в четеният от фреймуърка diplomnaChangelog.xml файл

Указаните стъпки, с изключение на добавянето на промените в „Git“ представлява процеса за зареждането на промените и стартиране на приложението както и на локална версия на приложението, така и на функционална версия на приложението. Взимайки в предвид логическото разделение на сървъра и визуалният интерфейс на два отделни проекта, процеса на зареждане представен в диаграма се вижда на Фиг. 8.



Фиг. 8 Диаграма на процеса за зареждане на нова версия на проекта

**III. Системна архитектура и база данни**

**1. Приемане на заявки и Jersey**

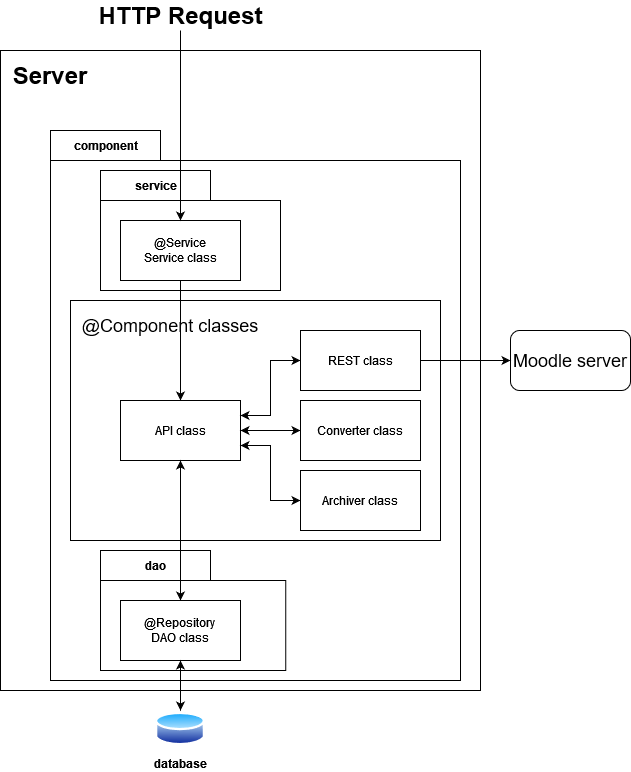
За комуникация с външни и вътрешни за приложението системи, приложението използва „Hypertext Transfer Protocol“ (HTTP) заявки. Сървърната страна на приложението, която е написана на Java 11, използва „Jersey“ 2.x фреймуърк за получаване на „HTTP“ заявки и трансформацията им в „Java“ обекти. Фреймуърка позволява кратък и интуитивен запис за всеки отделен адрес приемащ заявки благодарение на множество анотации, сред които анотации указващи адреса, метода, типа на входните данни и параметрите на заявката. Както приемането на заявката, така и отговора генериран от фреймуърка използваш широко-приетите стандарти (Фиг. 9).



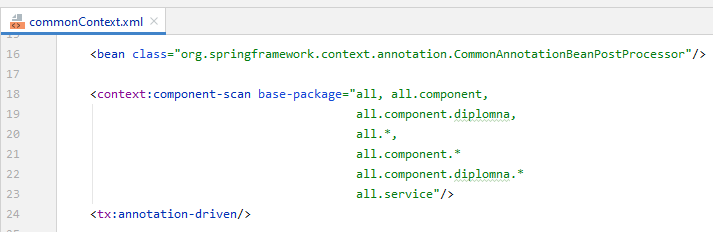
Фиг. 9 Вход на заявка „/getCourseData?courseId=X“, където X е параметър

**2. Менажиране на обектите и Spring**

За менажирането на обектите в процеса на работа на сървъра се използва „Spring framework“ известен със своят „Inversion Of Control“ (IoC) принцип, позволяващ на „Spring“ да зарежда инстанциите на обектите и да проверява за коректни взаимовръзка между тях. За целта, класовете трябва да бъдат анотирани като компоненти на „Spring“ чрез „@Component“ анотация или по-специализираните типове компоненти като „@Service“ и „@Repository“. Всеки компонент може да бъде достъпен и използван и в други компоненти чрез „@Autowired“ анотацията, която чрез интерфейса генерира Singleton обекти, имащи във всеки момент една единствена инстанция. Това се използва за осъществяване комуникацията между различните слоеве на приложението – презентационният слой с всички service-и, бизнес слоят съдържащ бизнес логиката и информационният слой, в който се извършва трансформацията на данните и връзката с базата данни (Фиг. 10).

  
  
Фиг. 10 Диаграма на слоевете, през които преминава информация предадена на системата

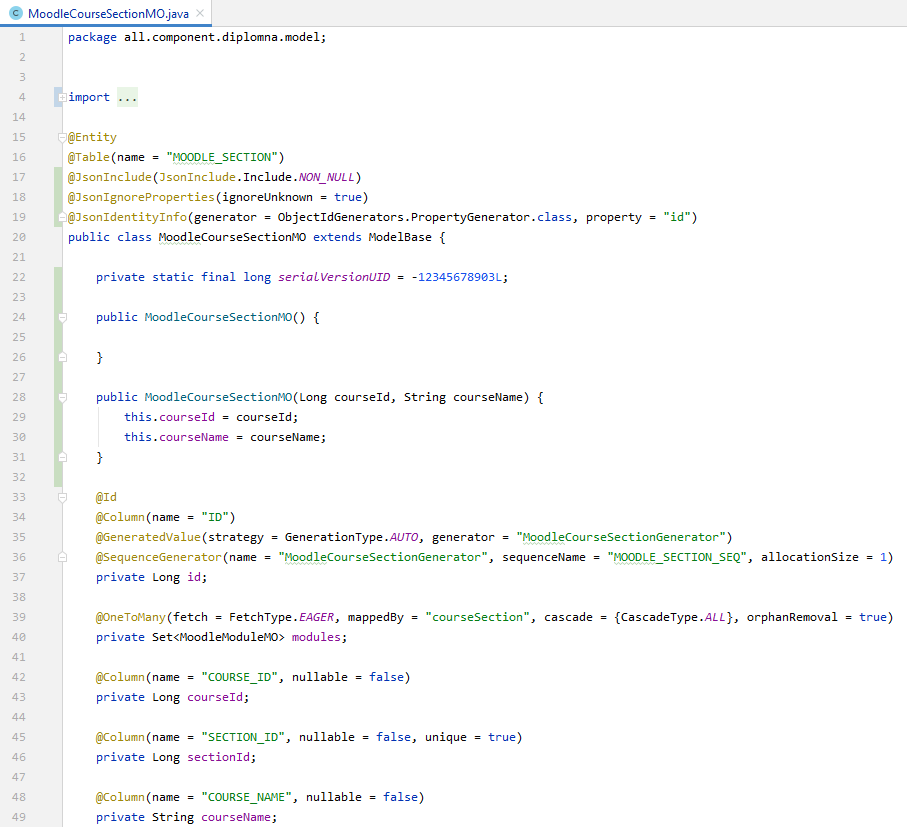
За работата на „Spring“ е необходим конфигурационен файл, указващ пътищата в проекта, където да бъдат търсени компоненти (Фиг. 11)



Фиг. 11 Конфигурация на пътища, в които „Spring“ търси компоненти в проекта и съответно класовете, в които е възможно използването на „IoC“

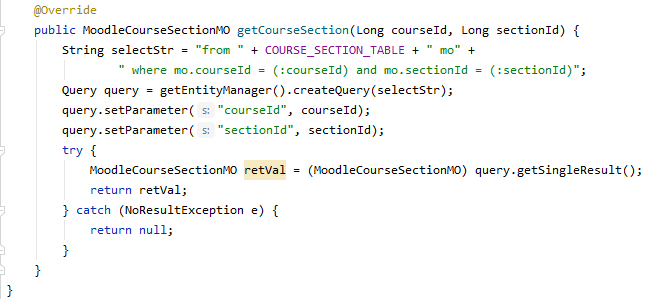
**3. Менажиране комуникацията с базата данни и Hibernate**

Заявки за четене и писане в базата данни се извършва чрез „Hibernate“ фреймуърка, за целта на което се създават специални класове съответстващи на различните таблици и инстанциите им представляват моделни обекти (Фиг. 12, „MO“ е съкратено за „Model Object“) на редове от базата данни. В класа на тези обекти, чрез анотации се означават името на таблицата, основният ключ (primary key) и всяка една от колоните. Нещо повече, могат да бъдат добавени и релациите на таблицата с други таблици и по този начин през обект, съответстващ на един ред от таблица А, можем директно да прочетем, запишем и променим ред от таблица Б, която има релация с таблица А. „Hibernate“ е интегрирана с „Java Persistence API“ (JPA) за изпълнението на заявки към базата данни.



Фиг. 12 Модел-обект репрезентиращ таблица от базата данни чрез Hibernate анотации. Също така се виждат 3 Jackson анотации за сериализиране на обекта към JSON формат

Вместо да се пишат „SQL“ заявки за определена база, „Hibernate“ използва близък по синтаксис език позволяващ не само заявки чрез имената на таблици и колони, а и заявки чрез моделните обекти и техните променливи. Това позволява код написан за една заявка да може да бъде изпълнен от различни видове бази данни, понеже Hibernate отговаря за трансформацията на заявката към подходящият за вида база език (Фиг. 13).

  
  
Фиг. 13 – Използване на Hibernate за генериране и изпълнение на заявка   
към базата данни върху таблицата от предишната фигура

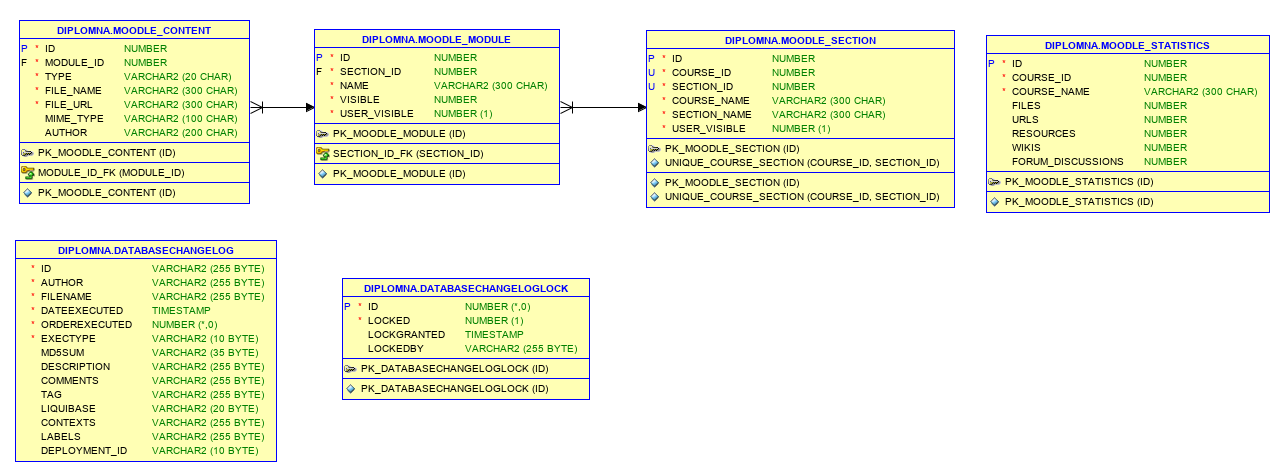
За да бъдат разпознати от „Hibernate“ модел-обектите е необходимо те да бъдат анотирани с „Entity“ анотацията (Фиг. 12) на „Hibernate“ и да бъде дефиниран „Entity“ мениджър (използван на Фиг.13, дефиниран на Фиг. 14). В проекта са интегрирани „Hibernate“ и „Spring“ фреймуърковете, и двата от които имат интеграция с „JPA“. Всеки един от дефинираните модел-обекти, както и пътят към него се дефинира и в конфигурационен файл. Менажирането, транзакциите и комуникация с базата данни са обяснени в точка „IV. Особености на програмната реализация“.



Фиг. 14 Конфигурационен файл на „Hibernate“ указващ всички модел-обекти и комуникация на фреймуърка с базата данни

**4. Архитектура на базата данни**

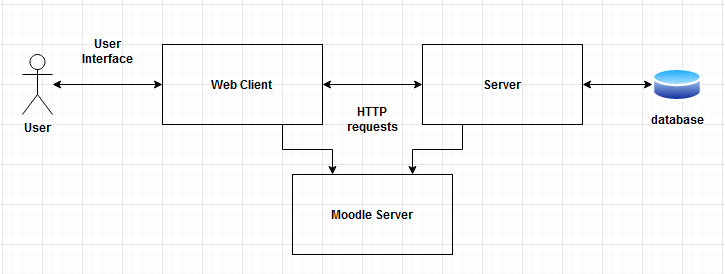
Архитектурата на базата (Фиг. 15) следва логическата йерархия установена от Мудъл системата, от където са извлечени данните. На най-горно ниво в йерархията има курс, а в него се съдържат различни модули – логически разграничени сегменти от курса – всеки от които има съдържание съставено от образователни ресурси като файлове, линкове, тестове и други. Извън тази йерархия, в базата данни съдържа статистически-значими за курс данни, но без да е използван ключ (foreign key). Причината за липса на връзка е, че статистически-значимите данни имат смисъл независимо от състоянието или изобщо текущото съществуване на курса. Релациите в базата данни не трябва да затрудняват осъществяването на исторически статистики и независимостта на статистическите данни от текущото състояние на един курс предполага наличието на всички статистически данни, следователно правейки релацията с други таблици ненужна. Освен значимите за Мудъл данни, в базата данни съществуват и две таблици използвани от „Liquibase“ за запазване състоянието на базата данни и за предотвратяване едновременното изпълнение на промени по базата данни от повече от една инстанция на „Liquibase“.



Фиг. 15 Диаграма на релациите в базата данни

**5. Менажиране визуализацията на данните и Angular**

Съхранената в базата данни информация и функционалността на приложението се достъпват чрез „HTTP“ заявки като за улеснение на потребителя достъпа и визуализацията на данни се осъществяват в браузър на логически-отделената клиентска страна (Фиг. 16) на приложението написано на „Angular“, framework разработен от екип на „Google“ за използване на обектно-ориентираният език „Typescript“ за изграждане на уеб приложения. „Angular“ улеснява менажирането на файловете, позволявайки декларирането на използваните компоненти, както и предоставяйки обширна библиотека с полезни функционалности, сред които е и достъпването на страниците единствено чрез боравене с интерфейса, което в приложението е използвано за ограничаване на достъпа без оторизиране, но без да се налага използването на бисквитки. „Angular“ помага значително не само в представянето на популярен и приятен за визия визуален интерфейс, но и помага значително за прегледноста и поддържането на кода и дори подобрява бързодействието на повечето приложения.



Фиг. 16 Диаграма на архитектурата на комуникациите в приложението

Въпрос на дебат е дали е правилно „Angular“ проектите да се казва, че използват „Model-View-Controller“, „Model-View-ViewModel“ или изобщо е сигурно какъв тип архитектура/шаблон на дизайн имат приложенията имплементирани посредством фреймуърка, та с цел придържане към фактите ще бъде посочен принципа на работа само на по-абстрактно ниво. Логическите единици в приложение се наричат модули и имат централизиран файл, където се декларират използваните в модула компоненти и библиотеки. В един модул могат да бъдат вложени други модули, практически могат да се възприемат като библиотеки. Визуализацията на уеб страница се намира в .html файл, стиловете асоциирани с нея се намират в .css или .scss, а логиката на страницата се намира в .ts файл. Съвкупността от логика и визуализация наричаме компонент. Зад всяка страница има типично 3 отделни файла, но е възможно да не бъде използван файл за стиловете ако няма такива или бъдат дефинирани директно в .html файла. Фреймуърка е създаден с интерактивност в предвид, тоест боравенето с визуалният интерфейс представен в .html страницата изпраща събития (events), които достигат до компонента и така промените по визуалният интерфейс имат отражение върху кода на клиента. По аналогичен начин, промените в кода по използвани в интерфейса компоненти се отразяват визуално.

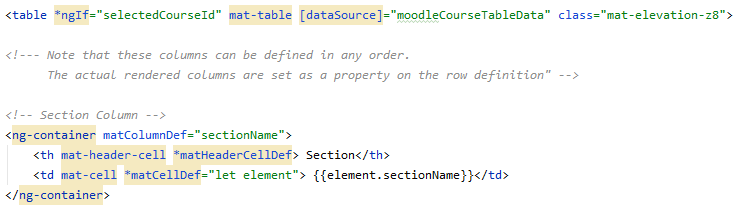


Фиг. 17 Използването на променливи от компонент във визуализацията на компонент представящ таблица с Мудъл курсове в course-table.component.html

На фиг. 17 се вижда имплементацията на падащо меню, което дефинирани две събития при натискането на мишката „click“, едно от които не само извиква метод от компонента, но дори препраща към параметър самото събитие и съдържаната в него информация. Различните курсове, репрезентирани с променливата „course“ се визуализират по стойноста им „courseName“ и избирането сменя курса (Фиг. 18 и Фиг. 19).



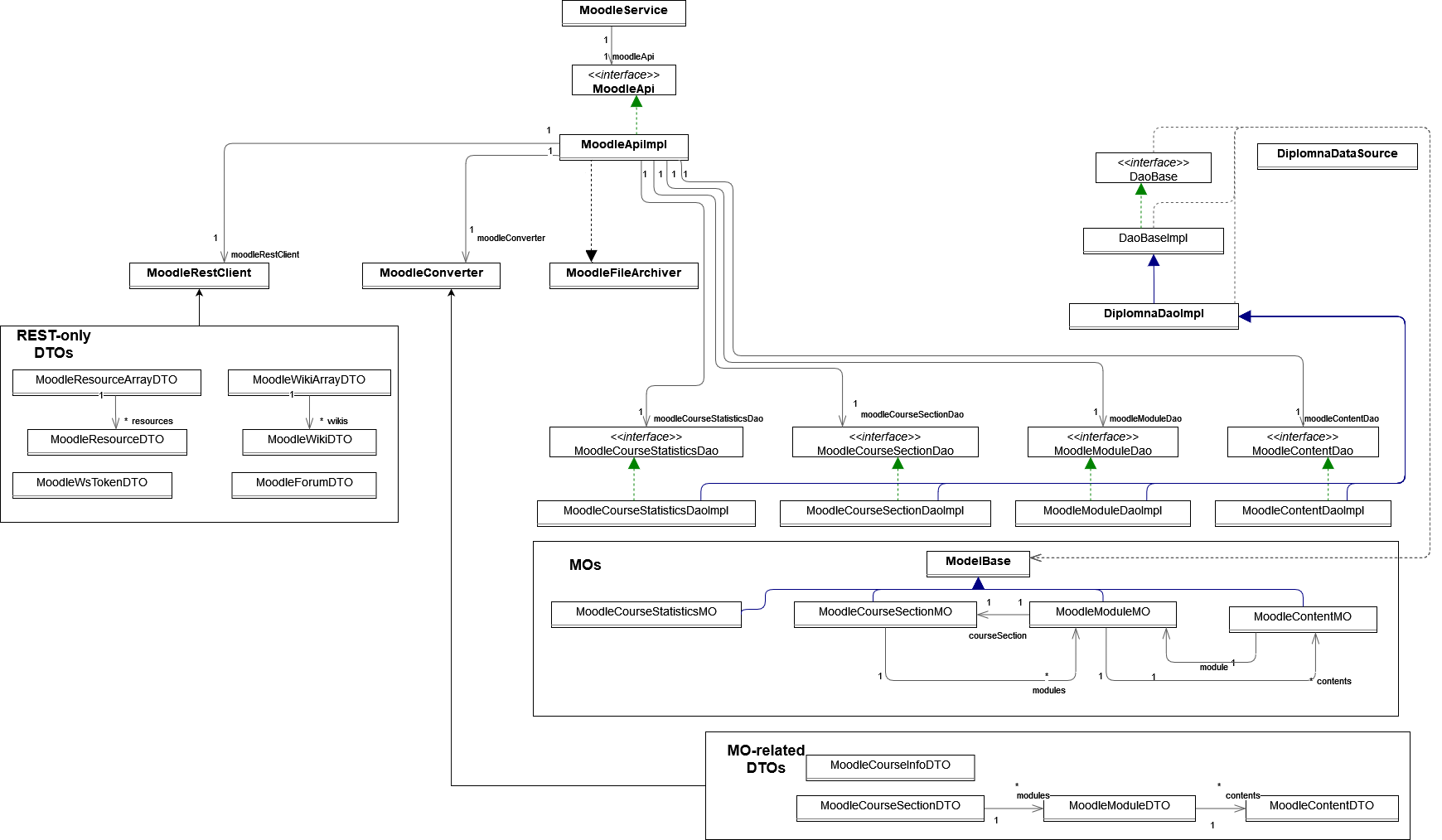
Фиг. 18 Дефинирането на компонента и функцията отговаряща за смяна на курса

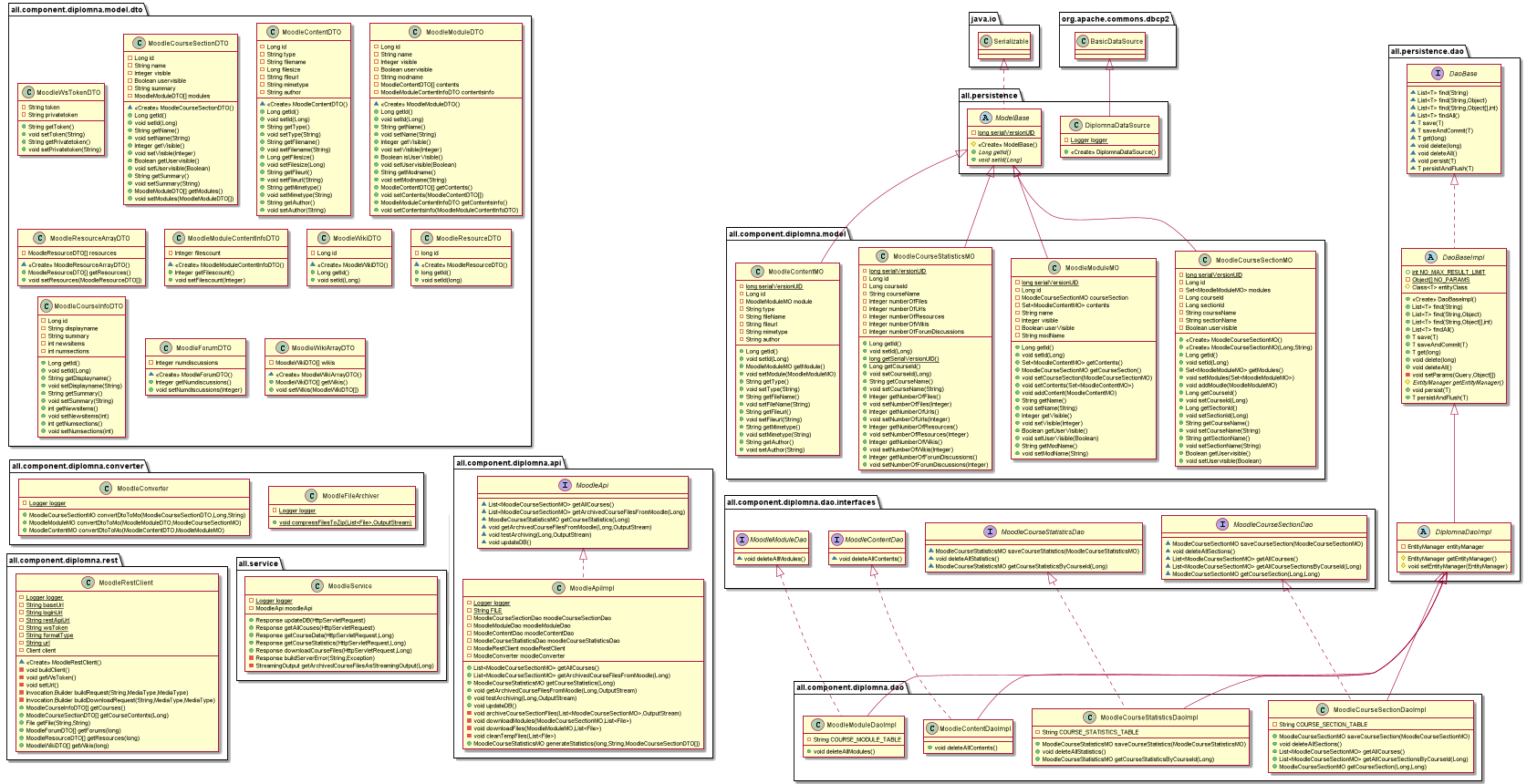


Фиг. 19 Визуализация на таблица в същата страница, като таблицата се определя от стойността избрана от потребителя, репрезентирана в логиката с променливата „moodleCourseTableData“

**IV. Особености на програмната реализация**

1. **Клас диаграми**

Фиг. 20 Клас диаграма на сървъра, показваща използването на класовете

****Фиг. 21 Клас диаграма на сървъра показваща методи, полета, пакети и наследствени връзки между класове и интерфейси

**2. Използвани технологии, похвати и шаблони за дизайн**

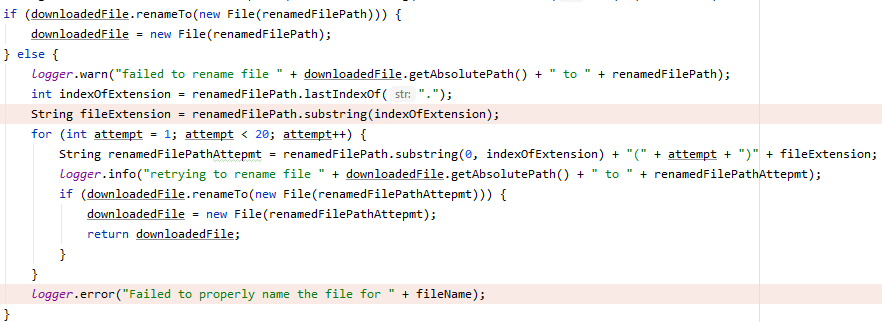
Системата използва многобройни и широко използвани технологии и шаблони за дизайн с цел написването на код, който се придържа към множество стандарти и по този начин е функционалността е лесна за разбиране, поддържане и разширяване.

**2.1 Сървър и комуникация с база данни**

За написването на кода в сървъра е използвана „Java 11“ поради широката популярност на „Java“ езика. Силните страни на езика, като множество потребители и съответно лесно намиране на отговор при възникване на проблеми, стабилни и стандартизирани библиотеки и фреймуъркове, обектно-ориентирано програмиране, строга типизация на обектите и др. са подходящи за търсеното решение. Слабите страни на езика, като недоразвитост на библиотеки и фреймуъркове, при които бързодействието е от ключово значение, не представляват пречка за даденият проблем.

а) Използват се четирите принципа на обектно-ориентираното програмиране - капсулация на данните за подобрена сигурност и контрол над задаването и достъпването на данни, наследяване за изнасянето на обща за множество обекти функционалност в едно единствено място, абстракция за скриване на ненужна функционалност и за дефинирането на стандарт за комуникация чрез интерфейси и полиморфизъм за изпълнението на сходна по логика но все пак различна функционалност върху различни класове през един общ метод.

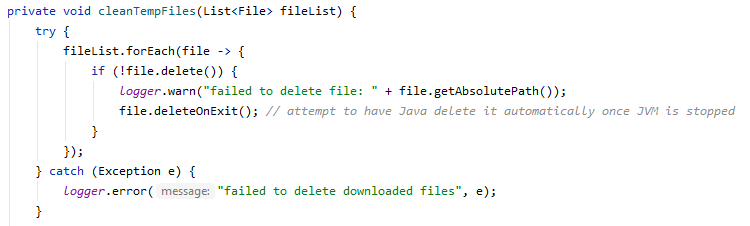
б) Анонимните класове са шаблон за дизайн използван за създаването на вложени класове при писането на по-сбит код, когато е необходимо един обект да не бъде рефериран множество пъти и съответно не е необходимо да бъде известно неговото име.1 Особено използване в проекта е използването на анонимните класове при създаването на файлове, показано на изображението по-долу. След извличането на файл чрез „HTTP“ заявка, „Java“ складира файла като временен файл с автоматично-генерирано име чрез хеширане. За промяна на името и местоположението на файла се използва „renameTo“ метода, но в случай на неуспех в кода остава инстанция репрезентираща файл, който всъщност не съществува и няма да съществува с невалиден път. За да се подобри четимоста и да се избегне това се използва анонимен клас при преименуване на файла и само в случай на успех се създава не вложен в метода анонимен клас. Видимо на Фиг. 22 по-долу е и опит за създаване на класа с подобно име, в случай на неуспех понеже най-често проблемът в разработка на проекта се дължеше на дуплициране на имена на файлове.

****

Фиг. 22 Използването на анонимни класове при преименуването на файлове в проекта

в) „Try-catch-finally“ блоковете са похват в „Java“ за прихващане и справяне със събития, които са изключения спрямо нормалното функциониране на програмата. Множество методи предупреждават за типовете изключения, които могат да се случат по време на изпълнението им. Например при работа с файл ако операционната система не позволи достъпа до файла или файлът бъде изтрит по време на използването му. Дефинирайки възможността за появата на изключителни събития в една функция предава отговорността на използващият ги разработчик да дефинира как програмата да функционира при използването на въпросната функция или да предаде отговорноста на още по-нататъшна функция (Фиг. 24).2

Този похват може да включва и „finally“ блок, който да бъде винаги изпълнен, независимо от това дали има събитие-изключение или не. В проекта има множество „try-catch“ блокове и нови слабо-дефинирани изключителни събития. В общият случай, подходът към тях е позволяването на системата да продължи да функционира нормално като запише появилата се ситуация в текстови лог файлове (Фиг. 23) и игнорира непредвидените ситуации, вместо стандартното поведение -прекратяване функционирането на сървърната система. Друг използван подход е дефинирането на стандартизиран „500 internal server error“ отговор на „HTTP“ заявките при срещането на недефинирано поведение (Фиг. 24).

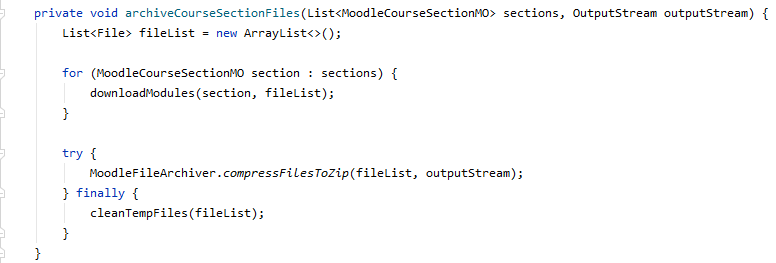


Фиг. 23 показваща опит за изтриване на временни файлове в блок за прихващане на изключителни събития



Фиг. 24 Код от програмата показващ опит за извличане на архивирани файлове за даден Мудъл курс посредством „HTTP“ заявка. При успех, се връща отговор с „HTTP“ код 200 „OK“ и списък от курсовете в „JSON“ формат. В случай на неочаквани събития се връща отговор с „HTTP“ код 500 „internal server error“

г) „Try-finally“ е особена разновидност на „try-catch-finally“ блока, който не се справя с възникнали при опита за изпълнение на метода изключителни събития или се справя с всички от тях по един и същ начин. Този блок дефинира код, който винаги трябва да се изпълни. В проекта е използван такъв блок за подсигуряване, че ще бъде направен опит за изчистването на файловете, свалени върху сървърната машина и преименувани така че да съответстват на имената им в Мудъл, след като те бъдат предоставени за сваляне на потребителя.



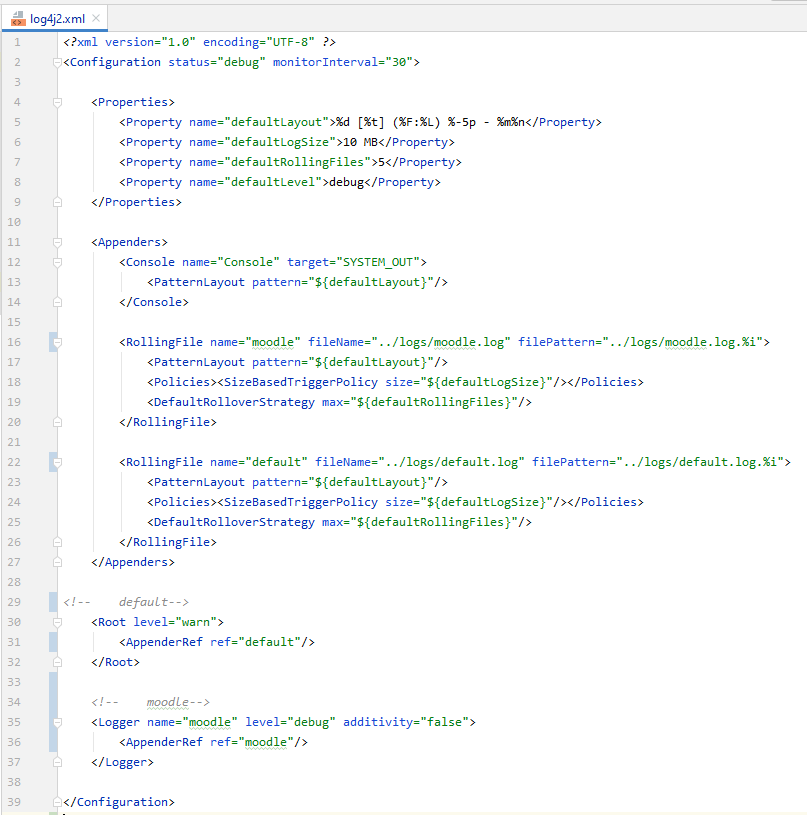
Фиг. 25 Изчистване на временно-съхранени файлове

д) „Try-with-resources“ е въведен с „Java 8“ опростен блок, подобен на „try-catch-finally“ блоковете, който позволява значително по-лесен за четене и поддръжка начин за справяне с изключителни събития при използването на ресурси. Ресурси, в контекста на „Java“ са обекти, които след приключване на използването им е необходимо да се затворят. Този блок за опростен запис на прихващане на изключителни събития позволява автоматичното затваряне на използваният ресурс и заместването на два „try-catch“ блока, които типично са необходими поради възможността за изключителни събития при затварянето на ресурс, с един „try-with-resources“ блок. За да е възможно това е необходимо ресурса, кокто трябва да се затвори да имплементира интерфейса „AutoCloseable“. Често работа с ресурси се среща при работа с файлове или други потоци данни и в проекта се имплементира „try-with-resources“ при архивирането на файлове. Въпреки работата с поток от данни при предоставянето за теглене на тези архиви, отговорен за затварянето на потока данни е „Jersey“ фреймуърка (Фиг. 26).



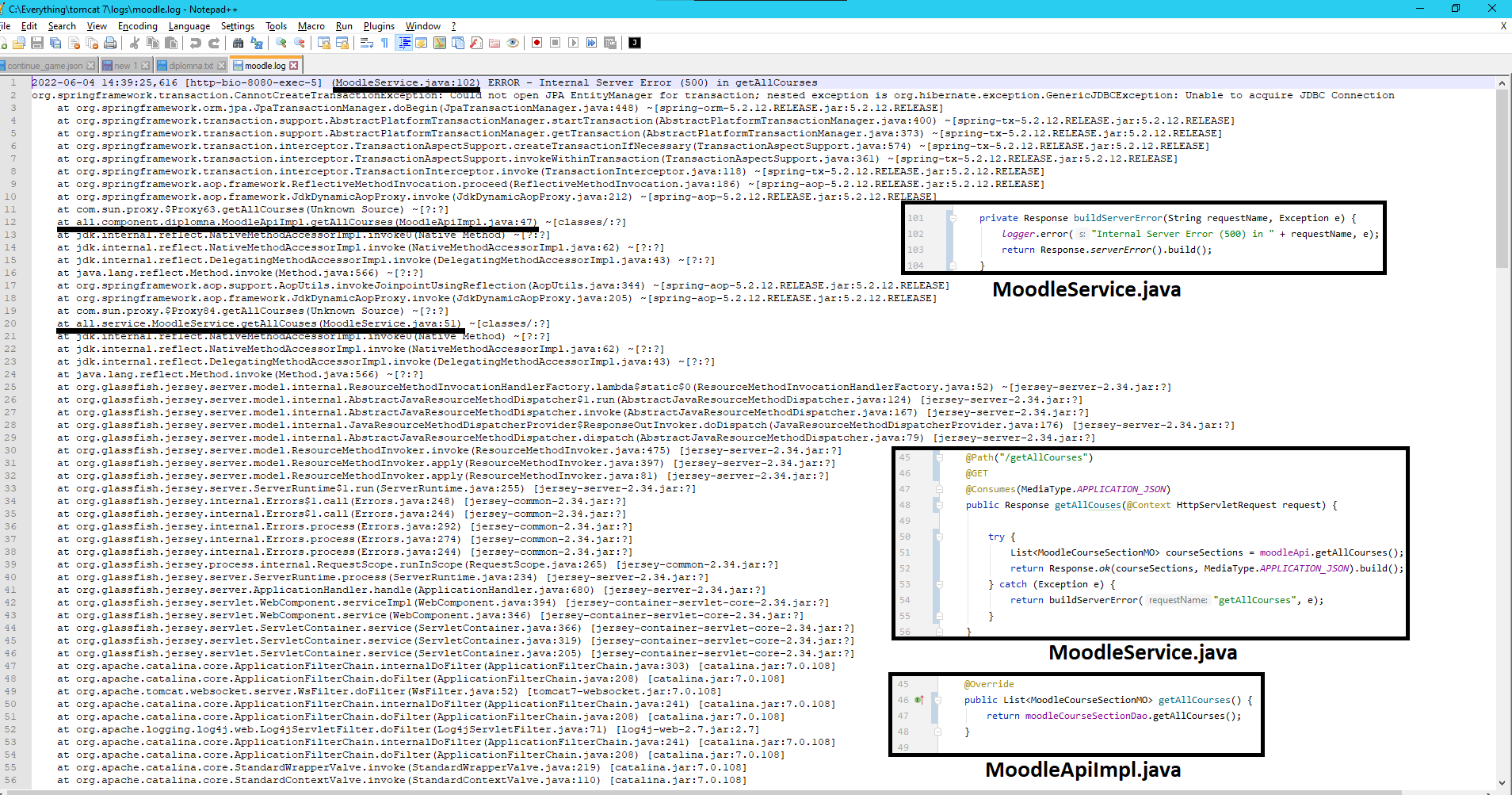
Фиг. 26 Имплементацията на архивирането на файлове и предоставянето им на поток от данни използван за тегленето им от потребителя. В случай на изключително събитие, грешката се записва и тегленето се прекратява

е) „Logger“ – както може да се види на множество фигури, в проекта се записват множество съобщения, особено при всяко непредвидено събитие. Тези съобщения се записват в специално-създадени за целта лог файлове, отделни от стандартните предвидени от сървъра файлове за лесно-скалируема и повишена гранулярност при разследване причината за изключителни събития и произхода им. Настройките на лог файловете показани (Фиг. 27) дефинират, че те не надвишават 10 мегабайта, записват всяко съобщение с ниво „debug“ и нагоре и се създават до 5 .log файла в отделна „logs“ директория. След достигането на 5 файла с размер 10 мегабайта, най-старият файл автоматично се изтрива и се създава нов на негово място. По този начин се подсигурява фиксиран размер количество логове съдържащи най-съвременната информация. На Фиг. 26 е видимо дефинирането на обект за изграждането на лог файлове специфичен за класа, в който се използва и две лог съобщения. Всяко от съобщенията е уникално за лесното разпознаване на мястото на грешката.

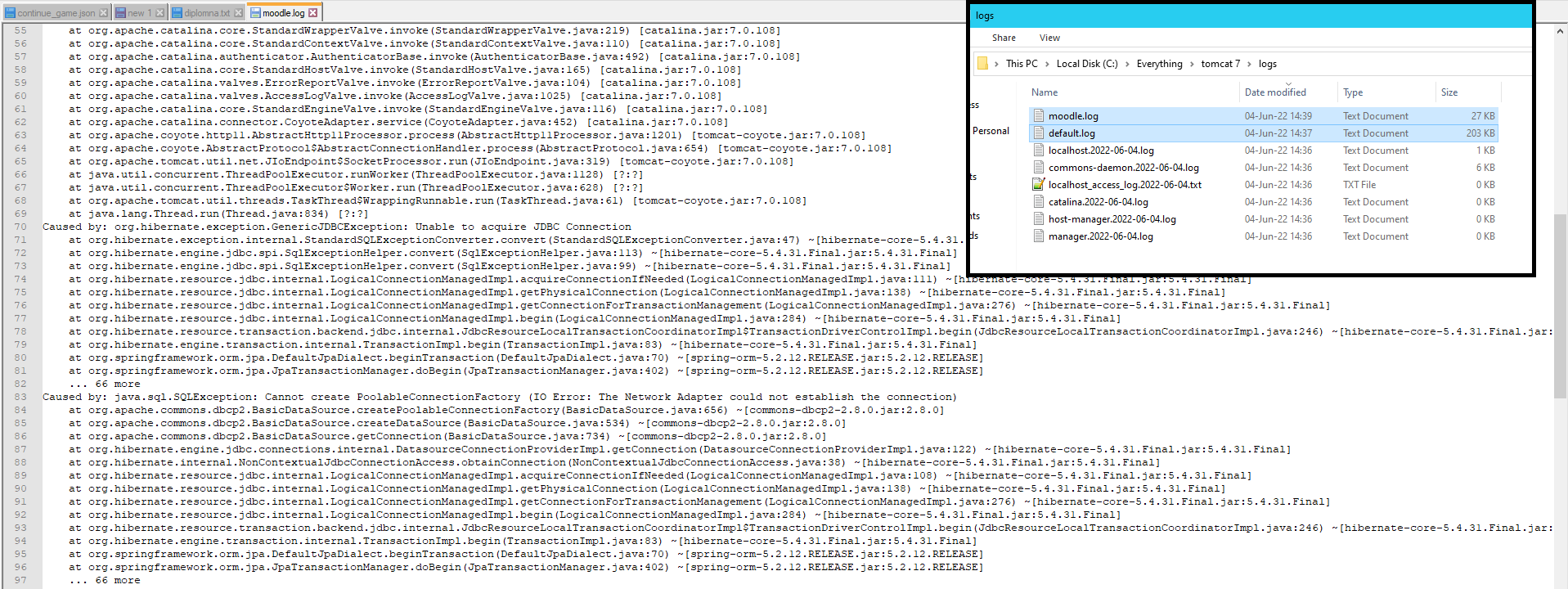


Фиг. 27 Конфигурация на логовете

Видимо на Фиг. 28 и Фиг. 29 представят набор от съобщения, които са записани при срещането на изключително събитие при нормалният процес на работа, записани последователно и образувайки така-нареченият „stack trace“. Четенето му е основно умение за всеки програмист, понеже е най-честият и стандартизиран начин за записването на възникнали събития и грешки при работата на едно приложение и използва същият принцип като при дебъг режим. Преведено „stack trace“ използва думите за стек и за следа/диря/път. Стека е линейна структура от данни, в която се добавят елементи един по един. При визуализирането на стека, първият елемент се явява дъното на структурата, а последният – върхът. При лог съобщенията се използва стек за проследяване пътят, по който е достигната грешката. Първият елемент на стека, неговият връх, е най-горното съобщение, което указва точно от кой ред е записана грешката.

****

Фиг. 28 Началото на лог съобщение и свързани парчета от кода на приложението



Фиг.29 Лог файл в папката на сървъра показващ изключително събитие при работата на програмата

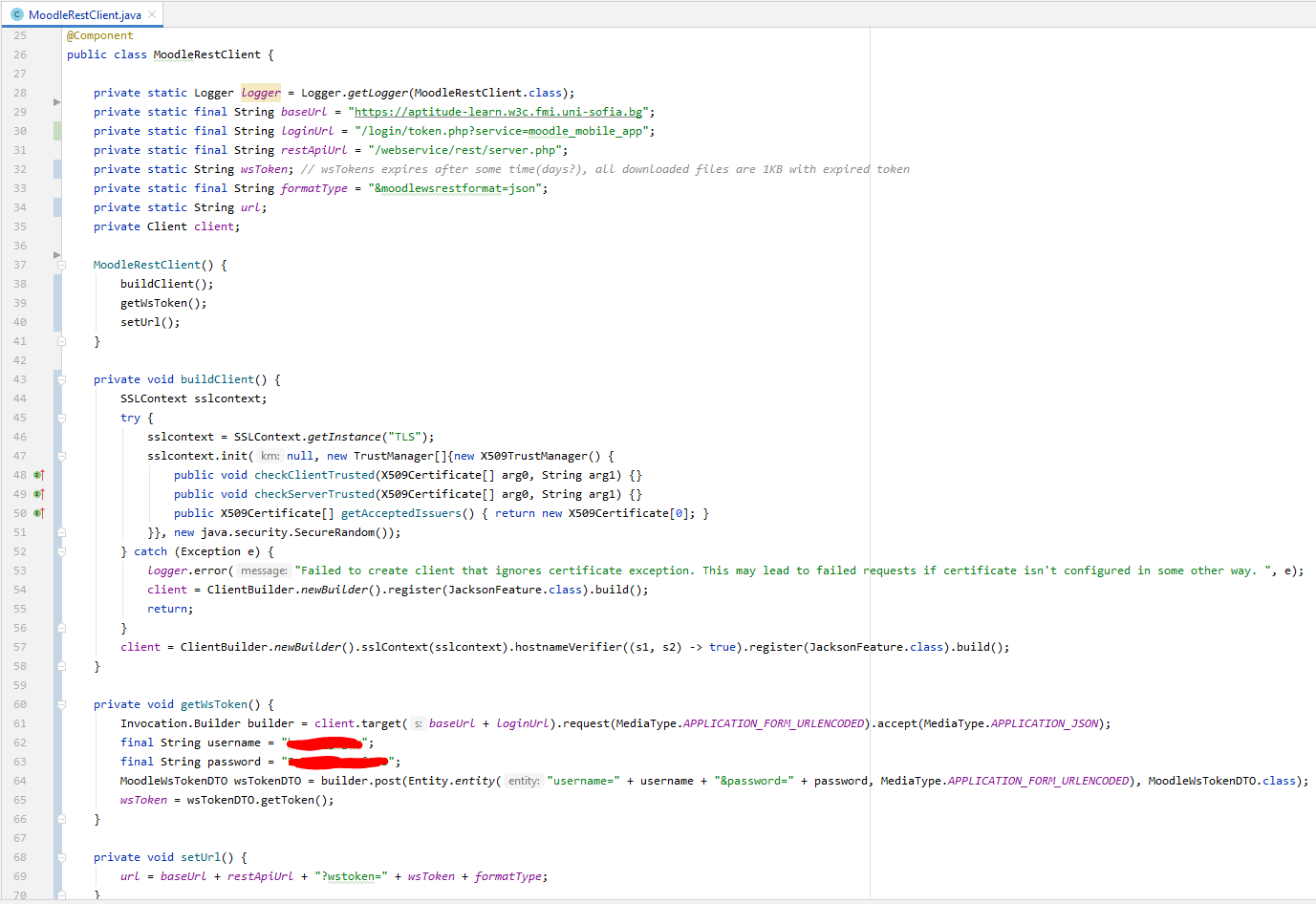
Показаният лог на Фиг. 28 показва записаното съобщение и указва че съобщението е записано от файла „MoodleService.java“ на ред 102, както можем да потвърдим на първата под-фигура. На вторият ред от файла е видимо и какво е изключителното събитие. От 3 до 70 ред е записан пътят, по който е достигнато до записване на събитието, показвайки на всеки нов ред адреса на файла, функцията, реда и евентуално библиотеката, в която се случва. Имайки в предвид структурата на папките в проекта, дори игнорирайки пътят преминат през библиотеките, можем да се фокусираме върху редове 12 и 20 от файла, които указват че се достига до „MoodleApiImpl“ ред 47 през „MoodleService“ ред 51.След дъното на всеки стек е възможно да срещнем нов стек с нова грешка (Фиг. 29), която е причината за последвалата я по-горна грешка. Често последователните стекове може да съдържат съкратена версия на стека, последвана от ред указващ че следват още редове, без да се изписват. Това се случва, само когато изписването им не е необходимо защото тези редове се повтарят. Обръщайки внимание на ред 81 от съобщението, можем да видим че той е практически идентичен с ред 3 и от него до дъното на най-горният стек има точно 66 реда. В действителност, множеството стекове общо образуват един общ стек, показващ причината за записване на съобщението с коренна причина най-долната грешка от най-долният стек (Фиг. 29).



Фиг. 30 Първоначалното събитие породило всички грешки.

Знаейки, че най-връхната грешка в написаният код е опит за достъп до „DAO“ клас, който поддържа връзка с базата данни за да може да записва и чете информация от нея, заедно с първоначалната грешка, може да се заключи че има проблем с връзката към базата данни. И наистина, показаният „stack trace“ е видим, когато сървърът е пуснат, но виртуалната машина не е и съответно е невъзможно да се осъществи връзка с базата данни. В случаят, понеже машината, върху която се намира базата данни е виртуална и комуникацията към нея се осъществява чрез същата машина използвана сървъра, съобщението не указва че не е осъществена връзка, а че тя е отказана. Това е защото машината успява да се свърже сама със себе си, но порта, чрез който се опитва да се свърже към виртуалната машина, не е отворен.

ж) „Representational State Transfer Application Interface“ (REST API) са интерфейси за комуникация чрез „HTTP“ заявки по определена архитектура за обекти, масиви и полета данни по стандартизиран начин.3 В проекта се използва „REST“ клиент за изграждането на комуникацията между Мудъл сървъра и сървъра на проектното приложение. Прави се опит за създаване на клиент с изключена проверка за „SSL“ сертификати, поради непостоянен проблем със сертифицирането на Мудъл сървъра, но при промяна на сървъра предоставящ приложението и разрешаване проблема със сертификатите, стандартната конфигурация е в „catch“ блока. За обртаботка на получените данни в „JSON“ формат и интеграция с „Jersey“ фреймуърка се регистрира „JacksonFeature“ класа (Фиг. 31).

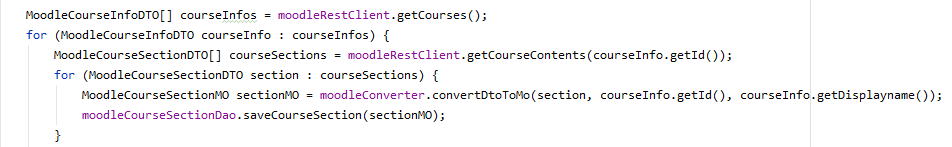


Фиг. 31 Конфигурация на „REST“ клиент за комуникация с Мудъл сървър посредством предварително създаден Мудъл потребител със специални права

з) „Builder“ шаблона за дизайн се дефинира чрез изграждането на строителен обект позволяващ модулярното изграждане на независими настройки на сложен обект чрез постъпковото извикване на методи на въпросният строителен обект, които връщат като резултат референция към самият него и имат един метод, който връща като резултат изграденият обект (Фиг. 32 и Фиг. 24).4

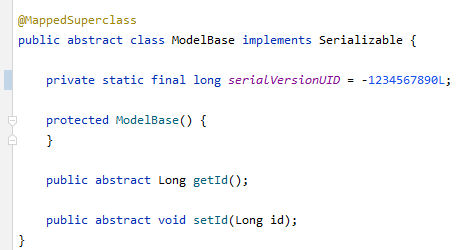
  
Фиг. 32 Използването на „Builder“ шаблона за изграждането на „REST“ клиент чрез задаване на винаги-валиден сертификат и на „Jackson“ библиотеката, която се използва за обработване на данните от „HTTP“ заявките в и от „JSON“ формат

и) „Data Transfer Object (DTO)“ шаблона е метод за дизайн за трансфер на данни наведнъж и по-нататъшната му трансформация в обекти, които не са непременно йерархически свързани с цел е намаляване броя изпълнения на ненужни заявки.5 проекта се използва при извличането наведнъж на данни от Мудъл отнасящи се за един курс (Фиг. 33).

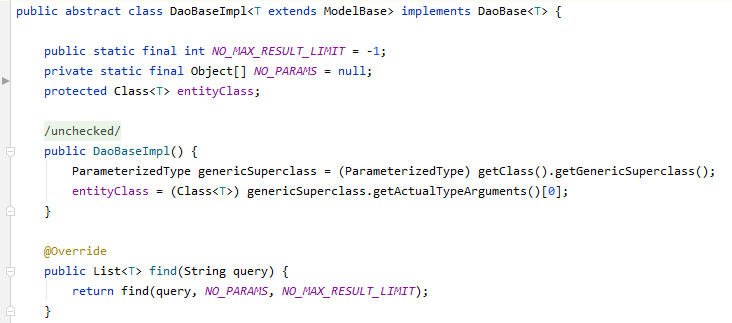


Фиг. 33 Извличане на „DTO“ посредством клиент за „HTTP“ заявки и трансформирането му в обекти преди запазването му в базата данни

й) „Template“ е шаблон използван за дизайн целящ стандартизация на някаква основна функционалност на наследяващи класове типично чрез абстрактен клас, който дефинира на високо ниво някаква основна функционалност необходима за всички класове наследяващи този шаблон (буквално преведено – template).6 В проекта това се използва за изграждането на шаблон за всички модел-обекти (Фиг. 35), които ще бъдат използвани от „Hibernate“ и задължаването на имплементация на някакъв уникален идентификатор, който фреймуърка изисква за да функционира (Фиг. 34).



Фиг. 34 Шаблон използван за всички модел-обекти дефиниращ нуждата от уникален идентификатор и имплементиращ интерфейс за сериализация



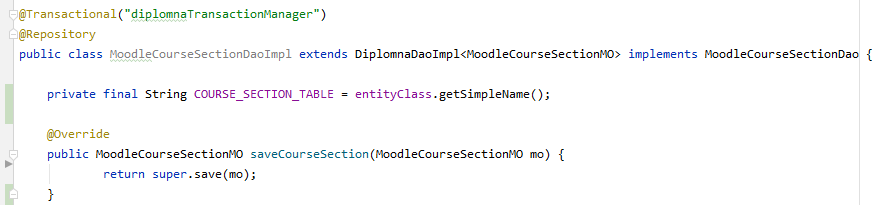
Фиг. 35 шаблон за всички „DAO“ класове, видимо е задължението те да работят с модел-обекти наследяващи шаблона за модел-обекти

к) „Dependency Injection“ шаблон е познат често с „Inversion of Control“ принципа, където при зависимоста за функционирането на един обект от друг, създаването на инстанцията на другият обект и продължителността на нейното съществуване не се определя от обекта имащ нужда от нея, а вместо това на него само се предоставя инстанция, с която да работи.7,8 В сървъра, това се осъществява посредством „Spring“ фреймуърка и „@Autowired“ анотацията (Фиг. 36). Тази анотация е използваема върху класовете използващи „Spring“ анотациите за различните видове компоненти, като „@Component“ и йерархически подчинените му „@Service“ и „@Repository“ компоненти, всичките реализирани и съответно менажирани от фреймъурка като основната за фреймуърка логическа единица - spring bean.9



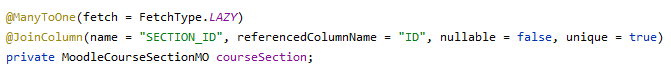
Фиг. 36 – Използването на „Spring“ за предоставяне достъп на компонента „MoodleApiImpl“ до други компоненти и използването на един от тях

м) „Data Access Object (DAO)“ шаблон за дизайн е похват целящ отделянето на комуникацията с базата данни в отделен слой на приложението с прост за използване интерфейс и скриване на това как е реализирана комуникацията с базата данни от други слоеве на програмата.10 В проекта, това е реализирано като за всяка таблица от базата данни има съответен модел-обект (MO) (Фиг. 12), който се използва посредством съответен за него „Dao“ интерфейс (Фиг. 35) и реализация на интерфейса и транзакцията му (Фиг. 37 и Фиг. 43). За комуникацията с базата данни се използва конфигурационен „datasource“ предоставен от и използван от съответно „Apache“ и „Spring“ фреймъурковете „dbcp2“ и „spring-jdbc“. За уникалните за приложението конфигурации, като адрес на базата данни, потребител и парола се използва .xml файл, който е в некриптиран формат за локалната виртуална база данни, но при необходимост от мерки за сигурност може да се работи с криптирани стойности или файловете да се добавят при изграждането на средата от друг източник, което означава, че за разработчика няма информацията за йерархически по-горни среди от локалната. Използва се „Oracle“ драйвера за комуникация с базата данни „ojdbc8“.



Фиг. 37 – Реализацията на „DAO“ интерфейс и задаването на транзакция за инстанцията

н) „Lazy loading“ е шаблон за дизайн, познат още като „Lazy initialization“ е подход при зареждането на информация или по-специфично отлагането или дори предотвратяването на зареждането на информация. Принципът на действие при този дизайн шаблон е една информация да не бъде заредена до момента, в който тя е използваната. В проекта се използва при извличането на информация от базата данни. Поради принципа на работа на „Hibernate“ модел-обектите, които репрезентират таблици в базите данни, съдържат в себе си други модел-обекти на таблиците, към които имат релации.Но, информацията от други таблици не е непременно или незабавно необходима и съответно има случаи, в които е за предпочитане тази информация да не бъде извлечена с първата заявка към базата данни. Промяната в заявката, за да не бъде извлечена допълнителна информация от базата данни и изпращане на нова заявка за допълнителната информация при необходимост, се извършва автоматично от „Hibernate“. Единствено е необходимо да бъде указано че е желано да се използва този подход (Фиг. 38).



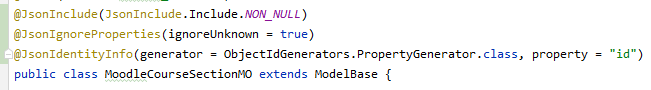
Фиг. 38 Релация „много към едно“ с друга репрезентирана с модел-обект таблица, чието извличане от базата данни е „Lazy“

Логически противопоставен на мързеливото зареждане е „Eager“ подхода, познат още като „алчен“. При него винаги се зарежда цялата налична информация в една заявка. Релацията „много към едно“ между един модул и курса към който принадлежи се използва защото когато ни е необходима информация за определен модул, то обикновено не ни е необходима голяма част от информацията за курса му, особено информацията за другите модули принадлежащи на курса, и съответно по-оптималното решение е да бъде използван мързеливият подход. От другата страна обаче, при релацията „едно към много“ между курс и всичките му модули често в проекта има нужда от информацията не само на курса но и на модулите му и съответно по-оптимално решение е да се използва „Eager“ подхода (Фиг. 39).



Фиг. 39. Показваща „едно към много“ релация между курс и модулите му, която използва „Eager“ подхода

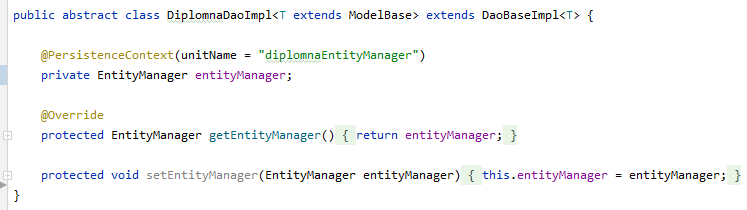
o) Сериализация в програмирането се нарича превръщането на обект в поток от данни за запазване или пренасяне на обекта през паметта. В този проект, сериализацията на модел-обектите има няколко особености. Поради репрезентацията на релации в базата данни посредством „Hibernate“ фреймуърка в модел класовете се образуват циклични структури. За да бъде представен във линейна структура (поток от данни) е необходимо да бъде указано как да бъде представен модел-обекта на сериализатора, в случаят „Jackson“ библиотеките. На Фиг. 40 са показани три анотации указващи тези инструкции. Най-долната разрешава проблема с циклични структури, уведомявайки библиотеката, че обекти с еднакъв параметър „id“ са еднакви и не е необходимо да бъдат представяни повторно. Втората анотация позволява десериализацията, т.е превръщането на поток данни в обект, на модел-обект да бъде извършена без изключителни събития от библиотеката в случай на подаване на параметри, които не съществуват в класа. Първата анотация пък позволява сериализатора да пропусне параметри без стойност при сериализацията.



Фиг. 40 Анотации за допълнителна конфигурация на сериализирането на клас

п) „Gson“ е библиотека разработена от „Google“ за сериализация на обекти и на места се използва за сериализация на обекти до „json“ формат за записване в логове или за тестване по време на дебъгване.

р) „Manager“ шаблон е дизайн за менажиране на обекти в проект с цел предоставяне на визуално чиста репрезентация на изпълнението на сложен код и съдържание на информация за всички менажирани обекти, позволявайки по този начин комуникация и достъп между тях, които могат да бъдат необходими по най-различни причини.11 Всяко приложение в „Java“ всъщност използва мениджъри, от които един от основните за функционирането им са „Class Loader“ класовете, които са едни от първите инстанции заредени в едно приложение и са отговорни за зареждането на останалите класове използвани от приложението, както системни, така и написани от програмиста. Съответно, грешки при мениджърите обикновено водят до срив на функционалността на системата, за която са отговорни. Особените и значими за този проект мениджъри са използваните от „Spring“ и „Hibernate“ мениджъри, които се грижат за всичките компоненти използвани от фреймуърковете (Фиг. 41, Фиг. 42). При „Spring“ ключовите използвани в проекта отговорности на мениджъра са „Dependency Injection“ и „Transactional“. От „Hibernate“ мениджъра е използвано разпознаването на модел-обектите като репрезентация на таблици от базата данни и използването на модел-обектите за изграждане на заявки с подходящи релации. Също така, „Hibernate“ предоставя и основна валидация като тип на данните и проверка дали на един обект записан/извлечен от базата данни му е променен уникалният идентификатор.

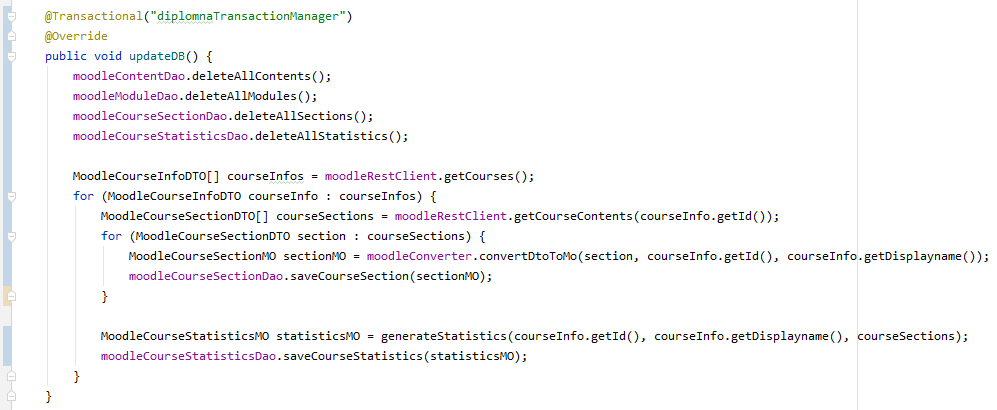


Фиг. 41 – Инициализацията на „Entity” мениджър в абстрактният клас „DiplomnaDaoImpl“ използван от всяко „DAO“



Фиг. 42 Файл за дефинирането на „Entity“ мениджърите

с) Транзакции са заявки към една база данни, които са обособени като една цяла единица и се изпълняват в строга последователност. При провал за изпълнението на коя да е от заявките се прекратява изпълнението на цялата транзакция и всички извършени от транзакцията промени се възстановяват до състоянието преди започване на транзакцията.12 Тоест, или се изпълняват всички заявки от транзакцията или не се изпълнява нито една от тях. Важно да да се спомене, че съществуват изключения, понеже не винаги всички събития са възстановими, както е често случаят при триене на информация от базата данни. Посредством „Spring“ и „Hibernate“ фреймуърковете и използваният от тях „Java Persistence API“ (JPA), в проекта се дефинират транзакции на ниво код. Това се осъществява, чрез „Spring“ анотацията „Transactional“ за създаване на транзакция използвана от „Entity“ мениджъра на „Hibernate“ (Фиг. 43).13



Фиг. 43 Обновяването на информацията в базата данни като част от транзакция

т) „Factory“ шаблона за дизайн, подобно на „Builder“ дизайна е похват за създаване на клас, предназначен за изграждане на други класове. За разлика от „Builder“ дизайна обаче, класовете-фабрики имат за цел да скрият функционалноста си и само да предоставят интерфейс, посредством който се предоставя възможност да се изиска производството на клас от фабриката. За разлика от стандартното създаване на обекти посредством конструктори, шаблона на фабриките позволява създаване на обекти от наследяващите го класове чрез полиморфизъм и има за принцип създаването най-специфичният обект, а не най-абстрактният обект. Поради това, типа на създаденият обект може да е неизвестен по време на компилация, а да стане известен чак по време на изпълнение. Фабрики са използвани при конфигурирането на „Spring“ в проекта, видимо на Фиг. 42 и Фиг. 45.14

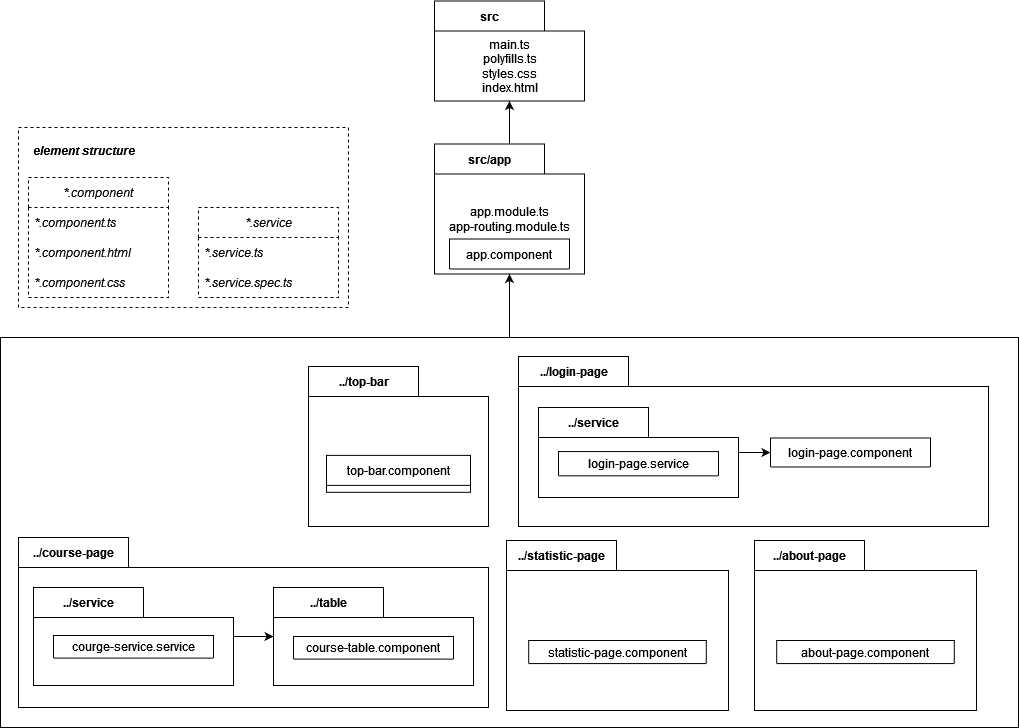
ф)За постигане на безпроблемна интеграция между сървър и база данни, се използва база данни разработена от „Oracle”с версия 18.4-XE, компанията която разработва езика използван за написването на кода използван от сървъра, Java 11, и съответно използваният драйвер за комуникация с базата данни, ojdbc8 (Фиг. 44, Фиг. 45).



Фиг. 44 Допълнителни „datasource“ конфигурации необходими за комуникация с базата данни

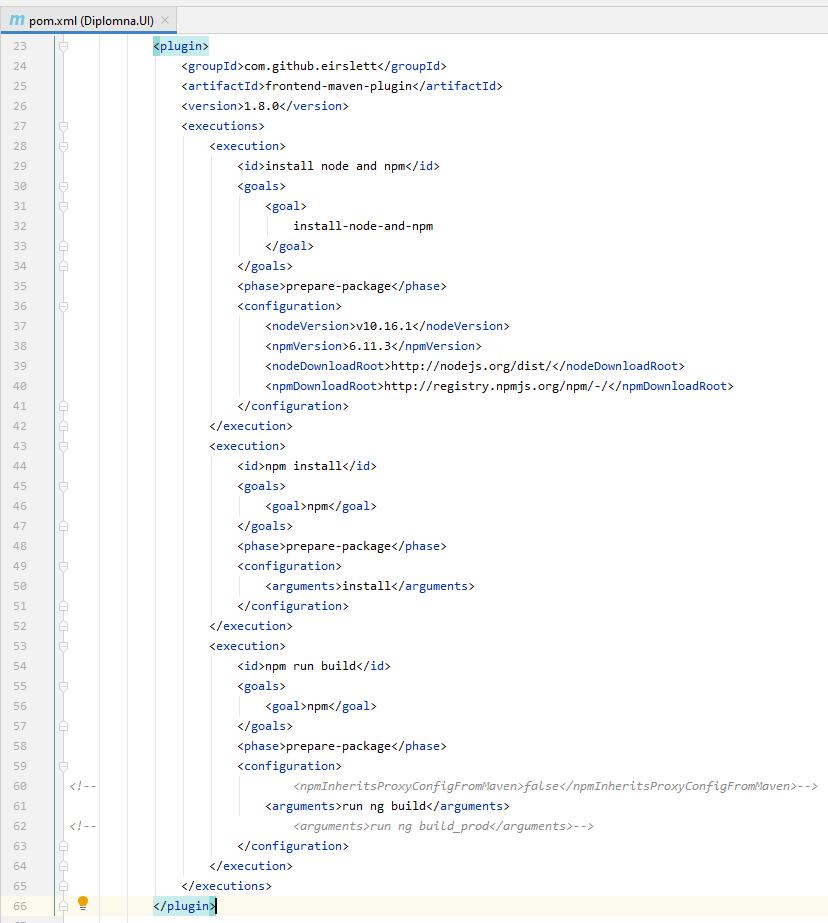
  
Фиг. 45 Конфигурациите в „commonContext.xml“ файла използван от „Spring“ за интегрирането му с „Hibernate“ и „datasource“ конфигурацията. На Фиг. 42 е видима връзката в конфигурацията на „Hibernate“ със „Spring“ и „datasource“ файла

**2.2 Визуален интерфейс**

****

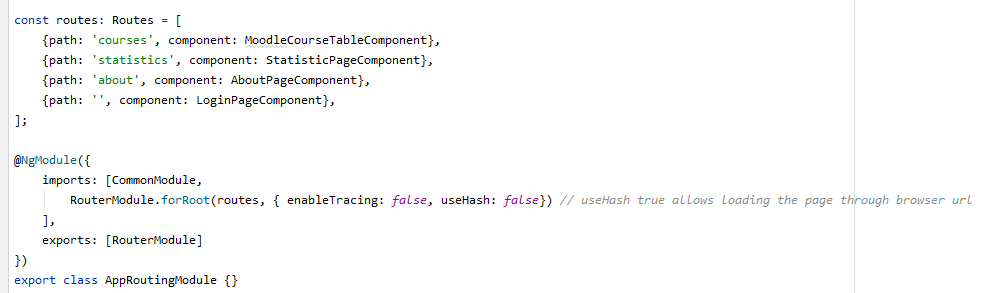
Фиг. 46 Архитектурата на визуалният интерфейс и легенда за съдържанието на компонент и service елементи

Визуалният интерфейс на приложението, още наричан клиента, е написан в отделен логически проект по смисъла, че езикът за написване, библиотеките, пътят и хранилището на визуалният интерфейс е изцяло логически отделено от сървъра и базата данни (Фиг. 46). Практически, стига стандарта за комуникация между сървъра и клиента да бъде запазен е напълно възможно да бъде подменен сървъра или клиента без това да се отрази върху функционирането на другият проект. Клиентът използва предимно „Angular“ фреймуърка за писането на компоненти на „Typescript“ и „HTML“ страници чиито стилове са дефинирани в „CSS“ или „SCSS“ файлове. Също като сървъра, за компилирането и пакетирането на кода за клиента се използва „maven“, но той не се използва за гарантирането на версиите на библиотеките, нито за тяхното теглене. За тази цел се използват „nodejs“ и „npm“, а версиите на всяка библиотека се дефинират в „package.json“ . Стартирането на „nodejs“ и „npm“ е опростено и се извършва посредством дефинираната в пом файл конфигурация. Автоматизирайки това се избягва необходимостта всеки разработчик да познава и работи с различна мениджмънт система (Фиг. 47).

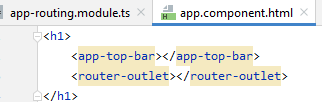
****

Фиг. 47 Използваната пом конфигурация за автоматизиране на използването на „nodejs“ и „npm“ в проекта

а) Навигацията в проекта е осъществена чрез „Routing“ модула, част от „Angular“ фреймуърка, и лента за навигация между различните компоненти. Посредством него е и забранено навигирането към страници посредством „URL“ адреси, по този начин налагайки контролирана навигация единствено чрез визуалният интерфейс без да се налага използването на бисквитки за валидиране на сесията. За да бъде използван в модула представляващ целият проект, рутирането трябва да бъде добавено в.html страницата на централният модул. Практически, въпреки че адреса на страницата да изглежда, че се променя е по-точно приложението да се разглежда като една единствена страница, която посредством указаната логика зарежда различни компоненти и влага техните .html страници в централният модул и неговата „app.component.html“ страница (Фиг. 48, Фиг. 49).15

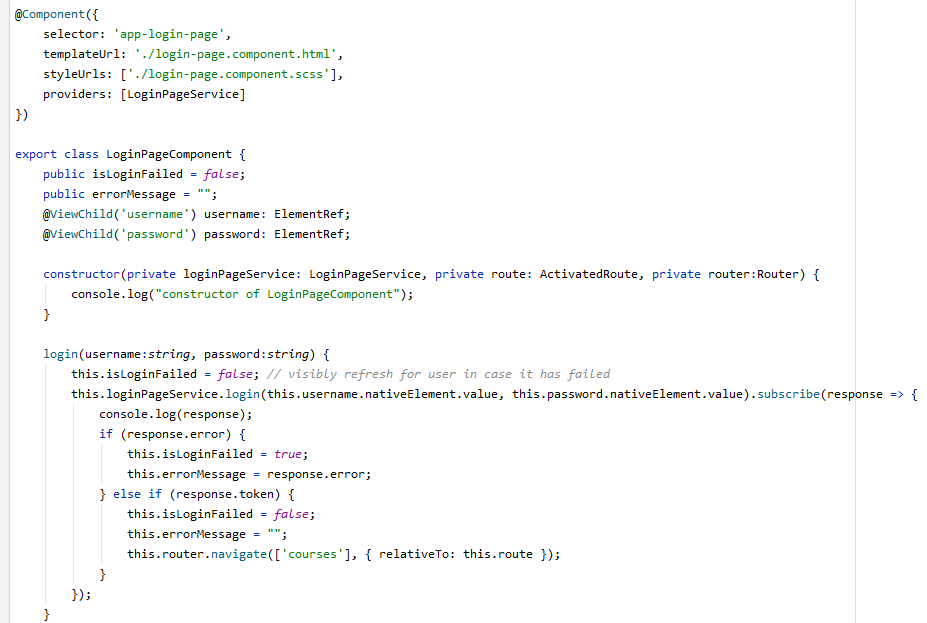


Фиг. 48 Конфигуриране на различните опции за навигация във визуалният интерфейс без съхранението на информация в браузъра на потребителя



Фиг. 49 Централно добавяне на рутиращият модул и лентата показваща различните възможности за навигация.

б) Както е видимо на по-горе (Фиг. 48), отварянето на приложението (началният адрес) визуализира компонента за автентикация и оторизация на потребителя – системата за вход. Поради неразривната връзка между това приложение и Мудъл системата, за целта оторизацията всъщност се извършва от Мудъл сървъра, с който приложението е интегрирано. Това се осъществява чрез опит за влизане в Мудъл системата през приложението (Фиг. 50, Фиг. 51). Осъществявайки оторизацията по този начин предоставя и възможноста при бъдеща разработка да бъдат интегрирани двете системи, така че да е необходимо потребителя да влезе успешно само в една от системите, за да има достъп до другата, осъществявайки така-наречената федерация на потребители.



Фиг. 50 Компонента за автентикация и оторизация. При успех, посредством рутера, се навигира към страницата с курсове. В този компонент е дефинирана връзката му със „service“ обект извършващ комуникацията със сървъра на Мудъл



Фиг. 51 Обект „service“, където се извършва оторизацията с външен сървър

в) „Observable“ шаблона на дизайн е похват използван при асинхронното програмиране за указване, че даден резултат е постоянно наблюдаван и се използва при очакване на стойност да бъде зададена на по-късен етап. Изпълнението на функция се осъществява само когато има абонати към резултата. Извикването на оторизиращият метод от „service“ класа и използването на .subscribe() метода указващ какво ще се случи при връщането на резултат са показани по-горе (Фиг. 50). Наблюдаемите обекти продължават да бъдат наблюдавани до изричното прекратяване на наблюдението, но всички инстанции в проекта наблюдават обекти с ограничен живот, чието наблюдение се прекратява автоматично от „rxjs“ библиотеката, от която са добавени.16

г) „Dependency Injection“ има и в проекта на клиента, използвано чрез „@Injectable“ анотацията (Фиг. 51), която указва че даден клас ще бъде използван в други класове. Това е едно от преимуществата на „Angular“ позволяващо ни да дефинираме функционалност на едно място и да я използваме на неограничен брой други места.17

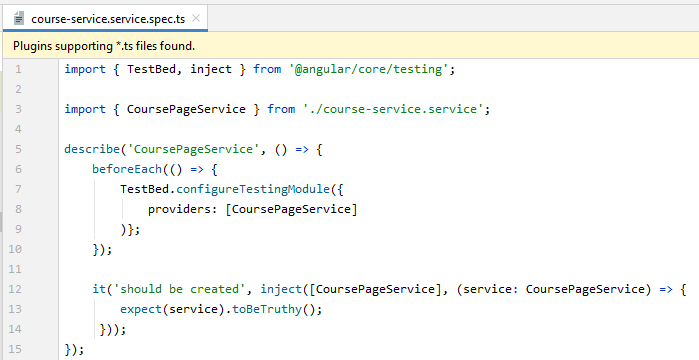
д) „Rest Client“ се използва и в клиент проекта за осъществяване на комуникация посредством „HTTP“ заявки (Фиг. 51).

е) „d3.js“ е библиотека предоставяща методи за програмното манипулиране на данни и динамично им визуализиране в диаграми посредством векторен формат за изображения „Scalable Vector Graphics“ (SVG).18,19 За показване статистически данни на курсовете намиращи се в базата данни се начертава стълбовидна диаграма в декартова координатна система. Реализацията на диаграмта се осъществява чрез динамичната и инкрементална трансформация на елементите на .html страниците, още познати като „Document Object Model“ (DOM) елементи. Постъпково към елемент се добавят различни атрибути, като ординатите, височина и ширина на стълбовете, цветове и др. като решението кога да се преначертае диаграмата може да бъде извършено във всеки един момент, стига да има смисъл от това. Поради архитектурата на .html страниците, сходните елементи имат общи класове, посредством които е възможно да се манипулират множество елементи едновременно. В проекта начертаването се извършва наведнъж при избирането на курс и изграждането на стълбовете едновременно, с разлика единствено в използваните параметри за всеки един от тях като височина, име и т.н (Фиг. 52).20,21



Фиг. 52 Чертането на стълбовидните елементи от диаграмата посредством програмното създаване на html елементи

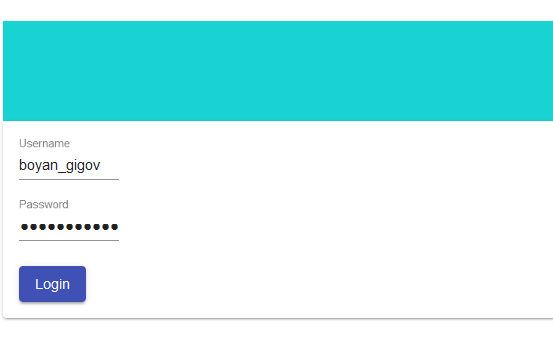
ж) „Unit“ тестове са автоматизирани тестове предназначени за изолиране на компоненти от проект и тестване на основните им функционалности за несъответствие с очакваното поведение (Фиг. 53). В „Angular“ проекта, това е осъществено посредством вградените в фреймуърка възможности и „jasmine“ и „karma“ библиотеките за визуализирането им.



Фиг. 53 Тест за създаването на service компонент

**V. Ръководство за потребителя**

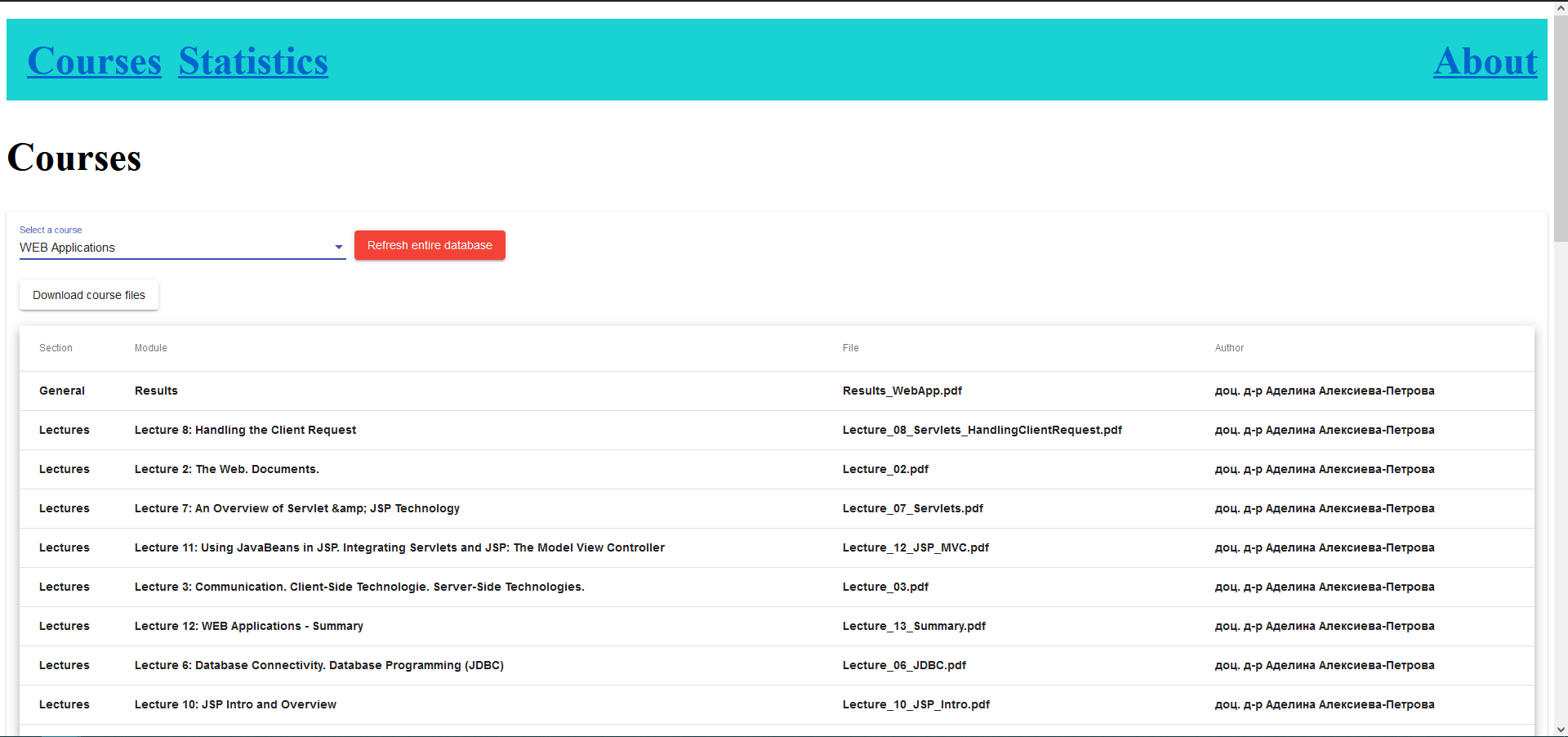
Ръководството за потребителя има за цел да обясни визуализираната функционалност на потребител с минимални компютърни умения, но все пак имащ опит при боравенето с стандартни компютърни функционалности на „Windows“ операционната система като интернет браузър и файлове.



Фиг. 54 Входната страницата и празна лента за менюта

Първата страница (Фиг. 54), която ще бъде видяна от всеки потребител. На нея има празна лента за менюта и в горният-ляв ъгъл на екрана се намират две полета, за потребителско име и за парола, и бутон за въвеждането им. Потребителското име и парола не се създават от това приложение, а се изпращат за валидация до Мудъл системата към което това приложение е свързано и автентикират въведените данни и оторизират, че потребителят има достъп до похватите използвани за извличане на информацията от Мудъл системата, основавайки се на идеята че стандартният Мудъл потребител няма достъп до тях и съответно трябва да бъдат предоставени специални права, права които биха били необходими за достъпването на информация необходима само на учител или администратор на Мудъл.

При валидни данни за потребител и парола и валидна оторизация, потребителят автоматично ще бъде прехвърлен на следващата страница. При грешка с автентикацията или оторизацията, под бутона в червен цвят ще бъде показано съобщението генерирано от Мудъл системата, указващо причина за недопускането на потребителя.



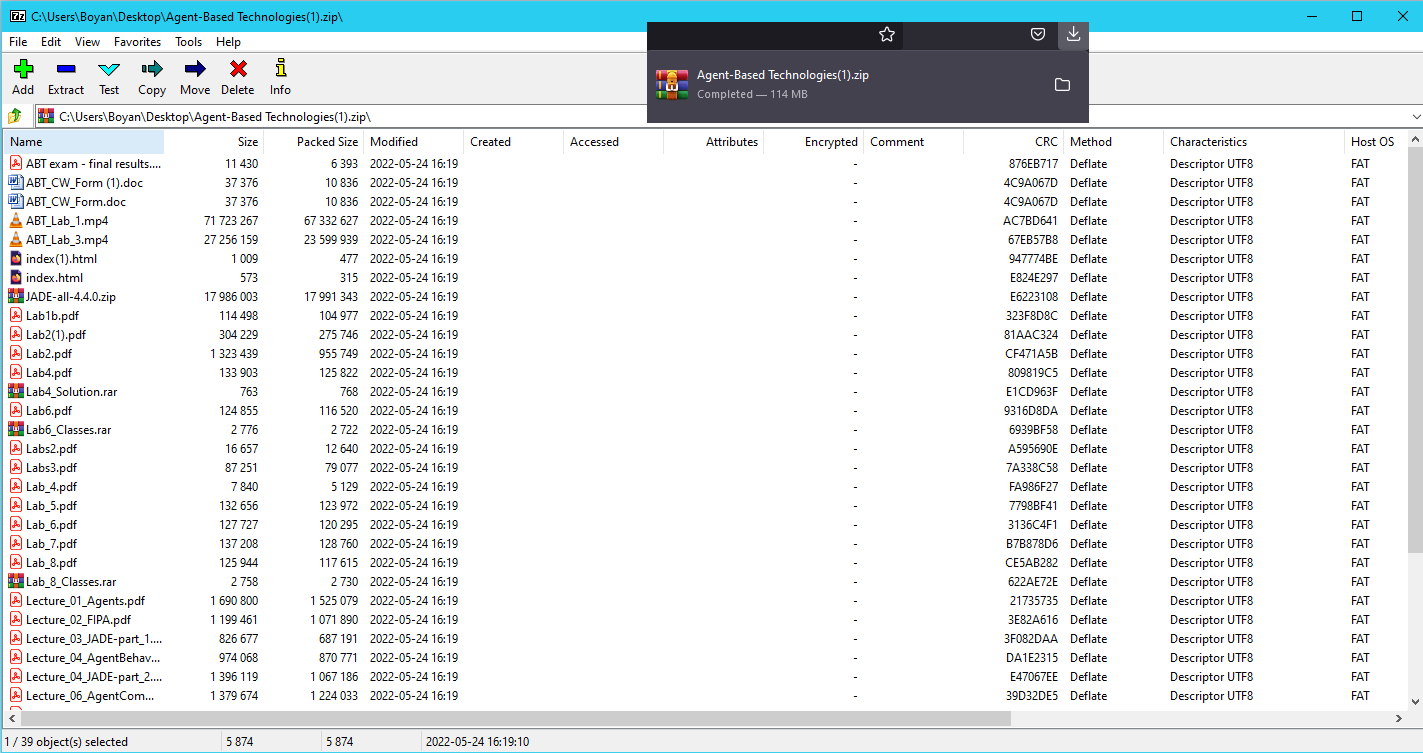
Фиг. 55 Страницата с курсовете и зареден един от курсовете

При успешна автентикация и оторизация потребителят ще бъде прехвърлен на страницата с курсовете (Фиг. 55). Лентата с менютата със зелен фон отгоре ще бъде изпълнена с всички възможности за навигация предоставени от приложението.

В горният ляв ъгъл на екрана ще бъде предоставен падащо меню съдържащо всички заредени в базата данни на приложението курсове. Натискайки върху него, потребителят има възможност да избере кой да е курс от списъка. При избирането на меню ще бъде заредена информацията за избраният курс в таблица. От ляво към дясно, първите две колони показват йерархическата позиция на файла, а следващите две показват името на файла и потребителят, който ги е качил в Мудъл. *За функционалноста* ***не е необходимо*** *приложението да е постоянно свързано към Мудъл система. Достатъчно е веднъж информацията да бъде заредена в базата данни на приложението, от където тя ще бъде четена до изтриването й.*

Отдясно на падащото меню има бутон в червено, позволяващ обновяването на базата данни. Това се осъществява чрез изтриване на информацията за курсовете записана в базата данни на приложението и изтеглянето на информацията за всички курсове от свързаната към приложението Мудъл система. *За функционалноста* ***е необходимо*** *приложението да е постоянно свързано към Мудъл система.*

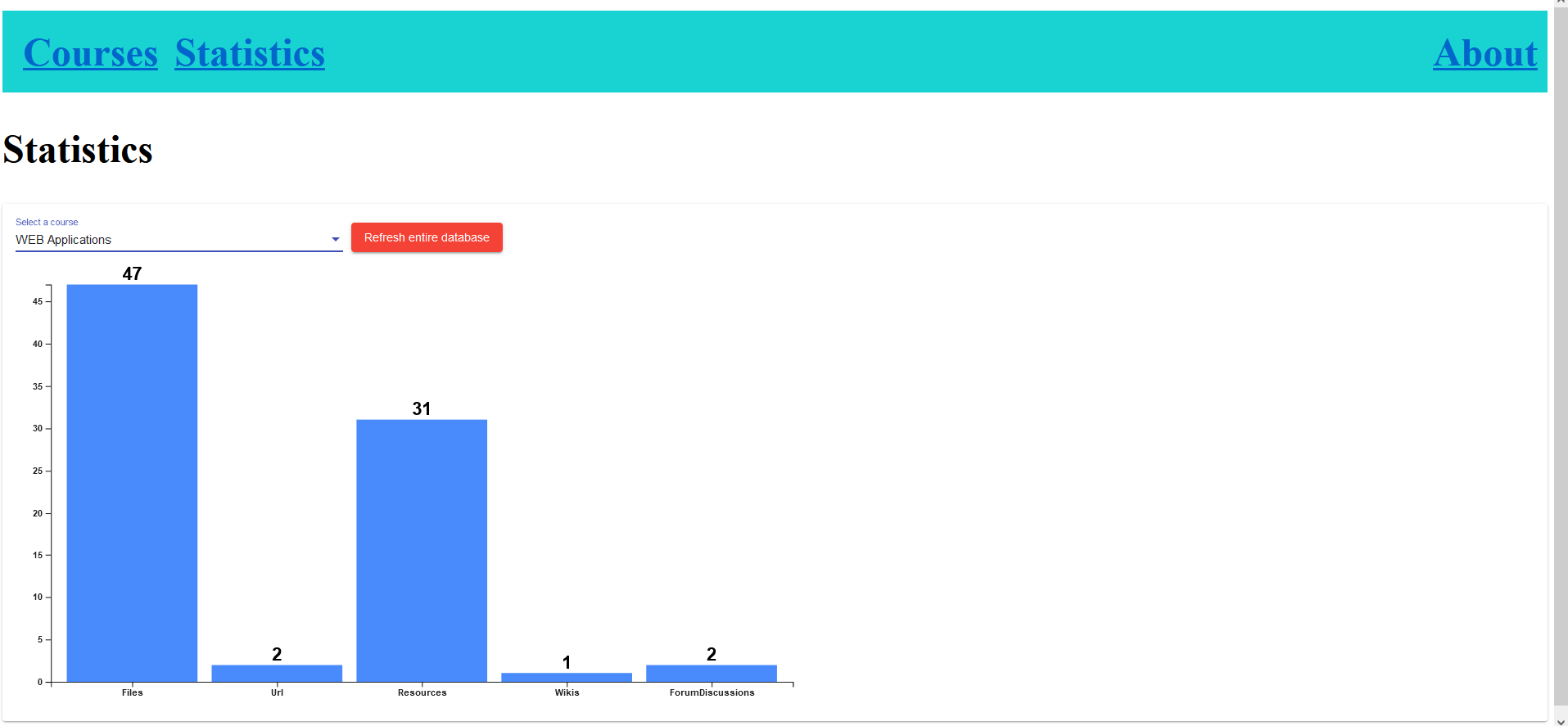
Под падащото меню, след зареждането на курс, има бутон за сваляне на показаните файлове. При натискането му, използвайки запазената в базата данни информация, от свързаната Мудъл система ще бъде направен опит за свалянето на всеки от показаните файлове. При неуспешно сваляне на кой да е файл, той ще бъде пропуснат и свалянето на другите файлове ще продължи. Съвкупността от свалени файлове ще бъде предоставена за сваляне на потребителя като архив в прозорец зависещ от използваният браузър. Архива съдържа всички файлове с показаното на страницата име на файла (Фиг. 56). При срещане на файлове с еднакви имена, файловете ще бъдат преименувани следвайки конвенцията установена в „Windows“ операционната система, където първият файл запазва името си, а всеки следващ има добавен номер в скоби. Приложението позволява до 20 файла с еднакво име. Архивът ще бъде именуван използвайки името на курса. *За функционалноста* ***е необходимо*** *приложението да е постоянно свързано към Мудъл система.*



Фиг. 56 Архив с всички файлове от курса свален през браузър. Файловете с еднакво име и размер (като ABT\_CW\_Form) се повтарят в самият Мудъл, а други имащи еднакво име, но различен размер (като Labs2.pdf) са различни файлове с еднакви имена.

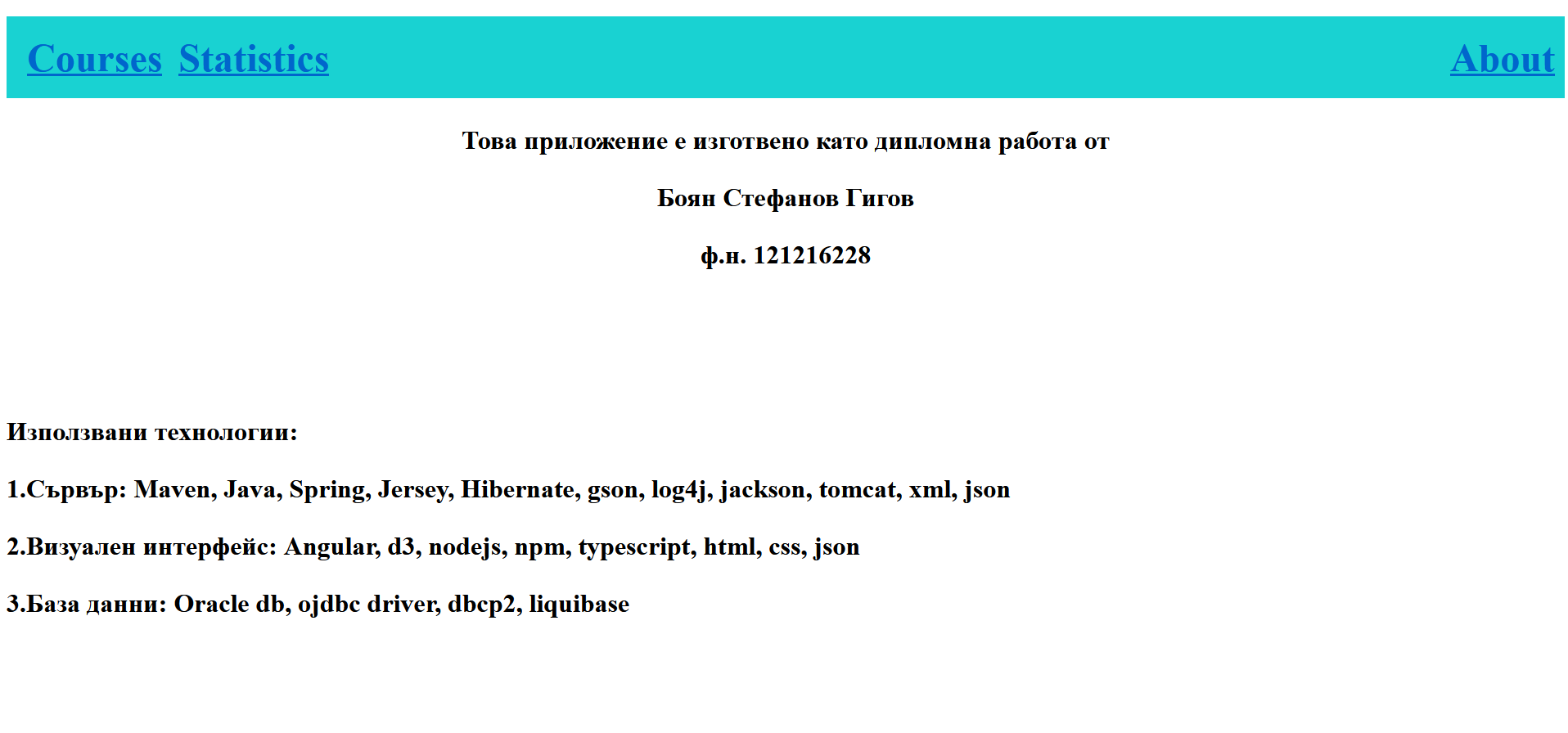
Навигирайки към страницата със статистиките (Фиг. 57), потребителят ще срещне подобен интерфейс на страницата със курсовете. Лентата с менютата ще остане непроменена. В горният ляв ъгъл се намира падащо меню съдържащо различните курсове. Избирайки един от тях ще начертае графика със статистическа информация за избраният курс. От ляво на дясно по абсцисата статистиката показва броят файлове, външни страници, ресурси, уикита и теми във форума на избраният курс. От долу на горе по ординатата е показан броят на срещаните елементи в курса. *За функционалноста* ***не е необходимо*** *приложението да е постоянно свързано към Мудъл система. Достатъчно е веднъж информацията да бъде заредена в базата данни на приложението, от където тя ще бъде четена до изтриването й.*

Подобно на страницата с курсовете, от дясно на падащото меню има червен бутон за обновяване на базата данни на приложението използвайки свързаната Мудъл система. *За функционалноста* ***е необходимо*** *приложението да е постоянно свързано към Мудъл система.*



Фиг. 57 Страницата със статистиките

Навигирайки към страницата с подробности (Фиг. 58), потребителят може да намери информация за използваните технологии за изграждането на приложението.



Фиг. 58 Страница с подробности за проекта

**VI. Заключение**

Задавайки си за цел улеснение свалянето на файлове от Мудъл система, проектът осъществява това чрез изготвянето на приложение използващо собствен сървър написан на „Java“ и използващ „Jersey“, „Spring“ и „Hibernate“ фреймуърци. Сървърът е интегриран със сървъра на Мудъл системата. Към сървъра на приложението са свързани и „Oracle“ база данни за складиране на информация като комуникацията между сървър и база данни се извършва чрез „Liquibase“ фреймуърка. Предоставянето на свалените файлове се извършва чрез сваляне на временни файлове върху сървъра, компресирането им по стандартният за „Java“ алгоритъм и предоставянето им за сваляне от потребителя във браузър. В отделен проект е изготвен е визуален интерфейс за браузър посредством „Angular“ фреймуърка, където е имплементирана система за вход интегрирана с Мудъл системата. Използвайки наличната информация за курсовете, във визуалният интерфейс са представени статистики за отделните курсове във вид на стълбовидни диаграми. Изготвянето на проектите е установено с фокус върху принципите на итеративната „Agile“ методология и позволява лесно и бързо изготвяне на версии на приложението. Това полага основи за постоянно тестване и разработка по приложението както по време на проекта, така и при евентуални бъдещи разширения на функционалността му.

В случай на допълнителна работа по приложението, могат да се направят набор от подобрения - Добавяне сертификата на интегрираната Мудъл система; Разделяне на файловете в крайният компресиран файл по папки съответстващи на подразделения за курса, когато едно подразделение има множество файлове; Разграничаването на потребител и администратор на приложението и ограничавайки част от функционалноста във визуалният интерфейс; Функционалност позволяваща промяната на настройки през визуалният интерфейс за администратори, например адрес на интегрираният Мудъл сървър и адрес на интегрираният сървър служещ за ауторизация и техните съответните потребителски имена и пароли; Допълнителни „unit“ тестове или интеграционни тестове и Допълнителни статистически данни.

**VII. Източници**

**1. Anonymous classes -** [**https://docs.oracle.com/javase/tutorial/java/javaOO/anonymousclasses.html**](https://docs.oracle.com/javase/tutorial/java/javaOO/anonymousclasses.html)

**2. Try blocks -** [**https://docs.oracle.com/javase/tutorial/essential/exceptions/handling.html**](https://docs.oracle.com/javase/tutorial/essential/exceptions/handling.html)

**3. REST -** [**https://www.redhat.com/en/topics/api/what-is-a-rest-api**](https://www.redhat.com/en/topics/api/what-is-a-rest-api)

**4. Builder pattern -** [**https://refactoring.guru/design-patterns/builder**](https://refactoring.guru/design-patterns/builder)

**5. DTO -** [**https://docs.microsoft.com/en-us/previous-versions/msp-n-p/ff649585(v=pandp.10)?redirectedfrom=MSDN**](https://docs.microsoft.com/en-us/previous-versions/msp-n-p/ff649585(v=pandp.10)?redirectedfrom=MSDN)

**6. Template pattern -** [**https://refactoring.guru/design-patterns/template-method**](https://refactoring.guru/design-patterns/template-method)

**7. Dependency Injection -** [**https://docs.spring.io/spring-framework/docs/3.0.0.M3/reference/html/ch04s03.html**](https://docs.spring.io/spring-framework/docs/3.0.0.M3/reference/html/ch04s03.html)

**8. Inversion Of Control -** [**https://docs.spring.io/spring-framework/docs/3.0.0.M3/reference/html/ch04.html**](https://docs.spring.io/spring-framework/docs/3.0.0.M3/reference/html/ch04.html)

**9. Spring bean -** [**https://www.baeldung.com/spring-bean**](https://www.baeldung.com/spring-bean)

**10. DAO -** [**https://www.oracle.com/java/technologies/dataaccessobject.html**](https://www.oracle.com/java/technologies/dataaccessobject.html)

**11. Manager pattern -** [**https://www.eventhelix.com/design-patterns/manager/**](https://www.eventhelix.com/design-patterns/manager/)

**12. Transactional –** [**https://developers.redhat.com/blog/2018/10/01/patterns-for-distributed-transactions-within-a-microservices-architecture#**](https://developers.redhat.com/blog/2018/10/01/patterns-for-distributed-transactions-within-a-microservices-architecture)

**13. Transactions -** [**https://www.tutorialspoint.com/dbms/dbms\_transaction.htm**](https://www.tutorialspoint.com/dbms/dbms_transaction.htm)

**14. Factory -** [**https://sourcemaking.com/design\_patterns/factory\_method**](https://sourcemaking.com/design_patterns/factory_method)

**15. Angular routing -** [**https://angular.io/guide/routing-with-urlmatcher**](https://angular.io/guide/routing-with-urlmatcher)

**16. Observable -** [**https://angular.io/guide/observables**](https://angular.io/guide/observables)

**17. Injectable -** [**https://angular.io/api/core/Injectable**](https://angular.io/api/core/Injectable)

**18. d3.js -** [**https://d3js.org**](https://d3js.org)

**19. SVG -** [**https://www.tutorialspoint.com/d3js/d3js\_introduction\_to\_svg.htm**](https://www.tutorialspoint.com/d3js/d3js_introduction_to_svg.htm)

**20. d3.js 2 -** [**https://blog.logrocket.com/data-visualization-angular-d3**](https://blog.logrocket.com/data-visualization-angular-d3)

**21. d3.js 3 -** [**https://edupala.com/how-to-use-angular-d3js-chart**](https://edupala.com/how-to-use-angular-d3js-chart)

**VIII. Приложение**

**1. Сървър**

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>BG.Diplomna</groupId>

<artifactId>Diplomna</artifactId>

<packaging>war</packaging>

<version>1.0-SNAPSHOT</version>

<properties>

<spring.version>5.2.12.RELEASE</spring.version>

<jersey.version>2.34</jersey.version>

<hibernate.version>5.4.31.Final</hibernate.version>

</properties>

<dependencies>

<!-- spring dependencies-->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-core</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aop</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-jdbc</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-beans</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-expression</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-orm</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-tx</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-web</artifactId>

<version>${spring.version}</version>

</dependency>

<!-- aspectjweaver 1.9.6 (autoloads spring ApplicationContext)-->

<dependency>

<groupId>org.aspectj</groupId>

<artifactId>aspectjweaver</artifactId>

<version>1.9.6</version>

</dependency>

<!-- ojdbc8 19.9.0.0 dependencies-->

<dependency>

<groupId>com.oracle.database.jdbc</groupId>

<artifactId>ojdbc8</artifactId>

<version>19.9.0.0</version>

</dependency>

<!-- jersey 2.34 dependencies-->

<dependency>

<groupId>jakarta.ws.rs</groupId>

<artifactId>jakarta.ws.rs-api</artifactId>

<version>2.1.6</version>

</dependency>

<dependency>

<groupId>org.glassfish.hk2</groupId>

<artifactId>hk2</artifactId>

<version>2.6.1</version>

</dependency>

<dependency>

<groupId>org.glassfish.hk2</groupId>

<artifactId>spring-bridge</artifactId>

<version>2.6.1</version>

</dependency>

<dependency>

<groupId>org.glassfish.jersey.containers</groupId>

<artifactId>jersey-container-servlet-core</artifactId>

<version>${jersey.version}</version>

</dependency>

<dependency>

<groupId>org.glassfish.jersey.core</groupId>

<artifactId>jersey-server</artifactId>

<version>${jersey.version}</version>

</dependency>

<dependency>

<groupId>org.glassfish.jersey.inject</groupId>

<artifactId>jersey-hk2</artifactId>

<version>${jersey.version}</version>

</dependency>

<dependency>

<groupId>org.glassfish.jersey.ext</groupId>

<artifactId>jersey-spring5</artifactId>

<version>${jersey.version}</version>

</dependency>

<dependency>

<groupId>org.glassfish.jersey.media</groupId>

<artifactId>jersey-media-json-jackson</artifactId>

<version>${jersey.version}</version>

</dependency>

<!-- hibernate dependencies -->

<dependency>

<groupId>org.hibernate.javax.persistence</groupId>

<artifactId>hibernate-jpa-2.1-api</artifactId>

<version>1.0.2.Final</version>

</dependency>

<dependency>

<groupId>org.hibernate</groupId>

<artifactId>hibernate-core</artifactId>

<version>${hibernate.version}</version>

</dependency>

<dependency>

<groupId>org.hibernate</groupId>

<artifactId>hibernate-entitymanager</artifactId>

<version>${hibernate.version}</version>

</dependency>

<dependency>

<groupId>org.hibernate</groupId>

<artifactId>hibernate-validator</artifactId>

<version>7.0.1.Final</version>

</dependency>

<!-- BasicDataSource provider-->

<dependency>

<groupId>org.apache.commons</groupId>

<artifactId>commons-dbcp2</artifactId>

<version>2.8.0</version>

</dependency>

<!-- gson dependencies-->

<dependency>

<groupId>com.google.code.gson</groupId>

<artifactId>gson</artifactId>

<version>2.8.6</version>

</dependency>

<!-- logger dependencies-->

<dependency>

<groupId>org.apache.logging.log4j</groupId>

<artifactId>log4j-api</artifactId>

<version>2.7</version>

</dependency>

<dependency>

<groupId>org.apache.logging.log4j</groupId>

<artifactId>log4j-core</artifactId>

<version>2.7</version>

</dependency>

<dependency>

<groupId>org.apache.logging.log4j</groupId>

<artifactId>log4j-web</artifactId>

<version>2.7</version>

</dependency>

<!-- json mapper dependency-->

<dependency>

<groupId>org.glassfish.jersey.media</groupId>

<artifactId>jersey-media-json-jackson</artifactId>

<version>${jersey.version}</version>

</dependency>

<!-- compile dependency-->

<dependency>

<groupId>org.apache.tomcat.embed</groupId>

<artifactId>tomcat-embed-core</artifactId>

<version>7.0.57</version>

<scope>compile</scope>

</dependency>

</dependencies>

<!-- build-->

<build>

<finalName>diplomna-webapp</finalName>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<!-- <version>3.6.3</version>-->

<configuration>

<forceJavacCompilerUse>true</forceJavacCompilerUse>

<!-- <release>11</release>-->

<source>11</source>

<target>11</target>

</configuration>

</plugin>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-war-plugin</artifactId>

<version>3.3.1</version>

<!-- <configuration>-->

<!-- <webXml>config/web.xml</webXml>-->

<!-- </configuration>-->

</plugin>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-resources-plugin</artifactId>

<version>3.2.0</version>

<executions>

<execution>

<id>replace-configurations</id>

<phase>initialize</phase>

<goals>

<goal>copy-resources</goal>

</goals>

<configuration>

<outputDirectory>target\diplomna-webapp\WEB-INF</outputDirectory>

<resources>

<resource>

<filtering>false</filtering>

<directory>C:\Users\Boyan\Desktop\Diplomna\config</directory>

</resource>

</resources>

</configuration>

</execution>

</executions>

</plugin>

</plugins>

</build>

</project>

<?xml version="1.0" encoding="ISO-8859-1" ?>

<web-app version="3.0" xmlns="http://java.sun.com/xml/ns/javaee" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app\_3\_0.xsd">

<welcome-file-list>

<welcome-file>\index.jsp</welcome-file>

</welcome-file-list>

<session-config>

<session-timeout>-1</session-timeout>

</session-config>

<context-param>

<param-name>contextConfigLocation</param-name>

<param-value>WEB-INF\classes\commonContext.xml</param-value>

</context-param>

<listener>

<listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>

</listener>

<listener>

<listener-class>org.springframework.web.context.request.RequestContextListener</listener-class>

</listener>

<servlet>

<servlet-name>rest</servlet-name>

<servlet-class>org.glassfish.jersey.servlet.ServletContainer</servlet-class>

<init-param>

<param-name>jersey.config.server.provider.packages</param-name>

<param-value>all.service</param-value>

</init-param>

<load-on-startup>1</load-on-startup>

</servlet>

<servlet-mapping>

<servlet-name>rest</servlet-name>

<!-- /myService/\*-->

<url-pattern>/\*</url-pattern>

</servlet-mapping>

</web-app>

<?xml version="1.0" encoding="UTF-8"?>

<persistence xmlns="http://java.sun.com/xml/ns/persistence"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://java.sun.com/xml/ns/persistence http://java.sun.com/xml/ns/persistence/persistence\_2\_0.xsd"

version="2.0">

<persistence-unit name="diplomnaEntityManager">

<provider>org.hibernate.jpa.HibernatePersistenceProvider</provider>

<class>all.component.diplomna.model.MoodleCourseSectionMO</class>

<class>all.component.diplomna.model.MoodleModuleMO</class>

<class>all.component.diplomna.model.MoodleContentMO</class>

<class>all.component.diplomna.model.MoodleCourseStatisticsMO</class>

<exclude-unlisted-classes>true</exclude-unlisted-classes>

<properties>

<property name="hibernate.connection.datasource" value="dataSourceXmlBean"/>

<property name="hibernate.cache.provider\_class" value="org.hibernate.cache.HashtableCacheProvider"/>

<property name="hibernate.show\_sql" value="true"/>

<property name="hibernate.format\_sql" value="true"/>

<property name="hibernate.connection.autocommit" value="true"/>

</properties>

</persistence-unit>

</persistence>

<?xml version="1.0" encoding="UTF-8" ?>

<Configuration status="debug" monitorInterval="30">

<Properties>

<Property name="defaultLayout">%d [%t] (%F:%L) %-5p - %m%n</Property>

<Property name="defaultLogSize">10 MB</Property>

<Property name="defaultRollingFiles">5</Property>

<Property name="defaultLevel">debug</Property>

</Properties>

<Appenders>

<RollingFile name="moodle" fileName="./logs//moodle.log" filePattern="./logs//moodle.log.%i">

<PatternLayout pattern="${defaultLayout}"/>

<Policies><SizeBasedTriggeringPolicy size="${defaultLogSize}"/></Policies>

<DefaultRolloverStrategy max="${defaultRollingFiles}"/>

</RollingFile>

<RollingFile name="defaultLog" fileName="./logs/default.log" filePattern="./logs/default.log.%i">

<PatternLayout pattern="${defaultLayout}"/>

<Policies><SizeBasedTriggeringPolicy size="${defaultLogSize}"/></Policies>

<DefaultRolloverStrategy max="${defaultRollingFiles}"/>

</RollingFile>

</Appenders>

<Loggers>

<!-- default-->

<Root level="debug">

<AppenderRef ref="defaultLog"/>

</Root>

<!-- moodle-->

<Logger name="all" level="debug" additivity="false">

<AppenderRef ref="moodle"/>

</Logger>

</Loggers>

</Configuration>

<?xml version="1.0" encoding="UTF-8" ?>

<beans xmlns="http://www.springframework.org/schema/beans"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xmlns:p="springframework.org/schema/p"

xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-4.0.xsd">

<bean id = "entityManagerFactoryDiplomna"

class="org.springframework.orm.jpa.LocalContainerEntityManagerFactoryBean"

p:jpaVendorAdapter-ref="jpaVendorAdapter"

p:persistenceUnitName="diplomnaEntityManager">

<property name="persistenceXmlLocation" value="WEB-INF\classes\persistence.xml"></property>

<property name="dataSource" ref="dataSourceXmlBean"></property>

<property name="jpaProperties">

<props>

<prop key="hibernate.dialect">org.hibernate.dialect.Oracle12cDialect</prop>

<!-- Oracle12c dialect to support v18 oracle db; also present in commonContext.xml-->

</props>

</property>

</bean>

</beans>

<?xml version="1.0" encoding="UTF-8"?>

<beans xmlns="http://www.springframework.org/schema/beans"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xmlns:jdbc="http://www.springframework.org/schema/jdbc"

xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-4.0.xsd

http://www.springframework.org/schema/jdbc http://www.springFramework.org/schema/jdbc/spring-jdbc-4.0.xsd">

<bean id="dataSourceXmlBean" class="all.persistence.DiplomnaDataSource" destroy-method="close">

<property name="driverClassName" value="oracle.jdbc.driver.OracleDriver"/>

<property name="url"

value="jdbc:oracle:thin:@(DESCRIPTION = (ADDRESS = (PROTOCOL = TCP)(HOST = localhost)(PORT = 1521))

(CONNECT\_DATA = (SERVER = DEDICATED) (SERVICE\_NAME = XEPDB1) ) )"/>

<!-- <property name="username" value="sys"/>-->

<!-- <property name="password" value="StarW"/>-->

<property name="username" value="DIPLOMNA"/>

<property name="password" value="123457"/>

<property name="validationQuery" value="SELECT 1 FROM DUAL"/>

<property name="testOnBorrow" value="true"/>

</bean>

</beans>

<?xml version="1.0" encoding="UTF-8"?>

<beans xmlns="http://www.springframework.org/schema/beans"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xmlns:p="http://www.springframework.org/schema/p"

xmlns:context="http://www.springframework.org/schema/context"

xmlns:tx="http://www.springframework.org/schema/tx"

xmlns:aop="http://www.springframework.org/schema/aop"

xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-4.0.xsd

http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop-4.0.xsd

http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-4.0.xsd

http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx-3.2.xsd">

<context:annotation-config/>

<aop:aspectj-autoproxy/>

<bean class="org.springframework.context.annotation.CommonAnnotationBeanPostProcessor"/>

<context:component-scan base-package="all, all.component,

all.component.diplomna,

all.\*,

all.component.\*

all.component.diplomna.\*

all.service"/>

<tx:annotation-driven/>

<!-- <bean id="API/DAO" class="IMPL class path"/>-->

<bean id="gson" class="com.google.gson.Gson"/>

<import resource="dataSources.xml"/>

<import resource="entityManagers.xml"/>

<bean id="sessionFactory" class="org.springframework.orm.hibernate5.LocalSessionFactoryBean">

<property name="dataSource" ref="dataSourceXmlBean"></property>

<!-- <property name="configurationClass"><value>org.hibernate.cfg.AnnotationConfiguratio</value></property>-->

<property name="hibernateProperties">

<props>

<prop key="hibernate.dialect">org.hibernate.dialect.Oracle12cDialect</prop>

<prop key="hibernate.show\_sql">true</prop>

</props>

</property>

</bean>

<bean id="transactionManager" class="org.springframework.orm.jpa.JpaTransactionManager"

p:entity-manager-factory-ref="entityManagerFactoryDiplomna">

<qualifier value="diplomnaTransactionManager"/>

</bean>

<beans profile="default">

<bean id="jpaVendorAdapter" class="org.springframework.orm.jpa.vendor.HibernateJpaVendorAdapter"></bean>

</beans>

</beans>

#### \_ \_ \_ \_ \_\_\_\_\_

## | | (\_) (\_) | | \_\_ \

## | | \_ \_\_ \_ \_ \_ \_| |\_\_ \_\_ \_ \_\_\_ \_\_\_ | |\_\_) | \_\_ \_\_\_

## | | | |/ \_` | | | | | '\_ \ / \_` / \_\_|/ \_ \ | \_\_\_/ '\_\_/ \_ \

## | |\_\_\_| | (\_| | |\_| | | |\_) | (\_| \\_\_ \ \_\_/ | | | | | (\_) |

## \\_\_\_\_\_/\_|\\_\_, |\\_\_,\_|\_|\_.\_\_/ \\_\_,\_|\_\_\_/\\_\_\_| |\_| |\_| \\_\_\_/

## | |

## |\_|

##

## The liquibase.sqlplus.conf file stores properties which are used during the

## execution of the Oracle SQLPLUS tool.

## Learn more: https://www.liquibase.org/documentation/config\_properties.html

####

####

## Note about relative and absolute paths:

## The liquibase.sqlplus.path must be a valid path to the SQLPlUS executable.

## The liquibase.sqlplus.timeout value can be one of:

## -1 - disable the timeout

## Any integer value > 0 (measured in seconds)

##

####

# The full path to the SQLPLUS executable.

# Sample linux path

# liquibase.sqlplus.path=/apps/app/12.2.0.1.0/oracle/product/12.2.0.1.0/client\_1/bin/sqlplus

# Sample windows path

# liquibase.sqlplus.path=c:\\oracle\\product\\11.2.0\\client\_1\\bin\\sqlplus.exe

# A valid timeout value for the execution of the SQLPLUS tool

liquibase.sqlplus.timeout=-1

# Flag to indicate whether or not to keep the temporary SQL file after exection of SQLPLUS.

# True = keep False = delete (default)

liquibase.sqlplus.keep.temp=true

#### \_ \_ \_ \_

## | | (\_) (\_) |

## | | \_ \_\_ \_ \_ \_ \_| |\_\_ \_\_ \_ \_\_\_ \_\_\_

## | | | |/ \_` | | | | | '\_ \ / \_` / \_\_|/ \_ \

## | |\_\_\_| | (\_| | |\_| | | |\_) | (\_| \\_\_ \ \_\_/

## \\_\_\_\_\_/\_|\\_\_, |\\_\_,\_|\_|\_.\_\_/ \\_\_,\_|\_\_\_/\\_\_\_|

## | |

## |\_|

##

## The liquibase.properties file stores properties which do not change often,

## such as database connection information. Properties stored here save time

## and reduce risk of mistyped command line arguments.

## Learn more: https://www.liquibase.org/documentation/config\_properties.html

####

####

## Note about relative and absolute paths:

## The liquibase.properties file requires paths for some properties.

## The classpath is the path/to/resources (ex. src/main/resources).

## The changeLogFile path is relative to the classpath.

## The url H2 example below is relative to 'pwd' resource.

####

# Enter the path for your changelog file.

changeLogFile=diplomnaChangelog.xml

#### Enter the Target database 'url' information ####

url=jdbc:oracle:thin:@(DESCRIPTION = (ADDRESS = (PROTOCOL = TCP)(HOST = localhost)(PORT = 1521)) (CONNECT\_DATA = (SERVER = DEDICATED) (SERVICE\_NAME = XEPDB1) ) )

# Enter the username for your Target database.

username: DIPLOMNA

# Enter the password for your Target database.

password: 123457

logLevel: INFO

strict=true

#### Enter the Source Database 'referenceUrl' information ####

## The source database is the baseline or reference against which your target database is compared for diff/diffchangelog commands.

# Enter URL for the source database

referenceUrl:

# Enter the username for your source database

referenceUsername:

# Enter the password for your source database

referencePassword:

# Logging Configuration

# logLevel controls the amount of logging information generated. If not set, the default logLevel is INFO.

# Valid values, from least amount of logging to most, are:

# OFF, ERROR, WARN, INFO, DEBUG, TRACE, ALL

# If you are having problems, setting the logLevel to DEBUG and re-running the command can be helpful.

# logLevel: DEBUG

# The logFile property controls where logging messages are sent. If this is not set, then logging messages are

# displayed on the console. If this is set, then messages will be sent to a file with the given name.

# logFile: liquibase.log

#### Liquibase Pro Key Information ####

# Learn more, contact support, or get or renew a Pro Key at https://www.liquibase.com/protrial

# liquibaseProLicenseKey:

#### Liquibase Hub Information ####

# Liquibase Hub is a free secure SaaS portal providing status reporting, monitoring & insights

# into your Liquibase database release automation.

# https://hub.liquibase.com

## Add your free Hub API key here

# liquibase.hub.apikey:

# liquibase.hub.mode:all

## Get documentation at docs.liquibase.com ##

## Get certified courses at learn.liquibase.com ##

## Get support at liquibase.com/support ##

liquibase.hub.mode=off

<?xml version="1.0" encoding="UTF-8"?>

<databaseChangeLog

xmlns="http://www.liquibase.org/xml/ns/dbchangelog"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xmlns:pro="http://www.liquibase.org/xml/ns/pro"

xsi:schemaLocation="http://www.liquibase.org/xml/ns/dbchangelog http://www.liquibase.org/xml/ns/dbchangelog/dbchangelog-4.2.xsd

http://www.liquibase.org/xml/ns/pro http://www.liquibase.org/xml/ns/pro/liquibase-pro-4.2.xsd ">

<!-- To run all files, use cmd to go to the directory this file is located in (and the liquibase.properties file)-->

<!-- and run liquibase update -->

<include file="DIPLOMNA-1.0.changelog.xml" relativeToChangelogFile="true"/>

</databaseChangeLog>

<?xml version="1.0" encoding="UTF-8"?>

<databaseChangeLog

xmlns="http://www.liquibase.org/xml/ns/dbchangelog"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xmlns:pro="http://www.liquibase.org/xml/ns/pro"

xsi:schemaLocation="http://www.liquibase.org/xml/ns/dbchangelog http://www.liquibase.org/xml/ns/dbchangelog/dbchangelog-4.2.xsd

http://www.liquibase.org/xml/ns/pro http://www.liquibase.org/xml/ns/pro/liquibase-pro-4.2.xsd ">

<changeSet author="Boyan.Gigov" id="create-MOODLE\_SECTION">

<preConditions onFail="MARK\_RAN">

<not>

<tableExists tableName="MOODLE\_SECTION"/>

</not>

</preConditions>

<createTable tableName="MOODLE\_SECTION">

<column name="ID" type="number" remarks="primary key">

<constraints primaryKey="true" nullable="false"/>

</column>

<column name="COURSE\_ID" type="number" remarks="id of the course as provided by moodle">

<constraints nullable="false"/>

</column>

<column name="SECTION\_ID" type="number" remarks="id of the section as provided by moodle">

<constraints nullable="false"/>

</column>

<column name="COURSE\_NAME" type="varchar2(300 char)">

<constraints nullable="false"/>

</column>

<column name="SECTION\_NAME" type="varchar2(300 char)">

<constraints nullable="false"/>

</column>

<column name="USER\_VISIBLE" type="boolean" defaultValue="0" remarks="can user see this section?">

<constraints nullable="false"/>

</column>

</createTable>

<createSequence sequenceName="MOODLE\_SECTION\_SEQ" incrementBy="1" minValue="1" startValue="1" />

<addUniqueConstraint

columnNames="COURSE\_ID, SECTION\_ID"

constraintName="UNIQUE\_COURSE\_SECTION"

tableName="MOODLE\_SECTION"

/>

</changeSet>

<changeSet author="Boyan.Gigov" id="create-MOODLE\_MODULE">

<preConditions onFail="MARK\_RAN">

<not>

<tableExists tableName="MOODLE\_MODULE"/>

</not>

</preConditions>

<createTable tableName="MOODLE\_MODULE">

<column name="ID" type="number" remarks="primary key">

<constraints primaryKey="true" nullable="false"/>

</column>

<column name="SECTION\_ID" type="number" remarks="id of the section from MOODLE\_SECTION table">

<constraints nullable="false"/>

</column>

<column name="NAME" type="varchar2(300 char)" remarks="display name">

<constraints nullable="false"/>

</column>

<column name="VISIBLE" type="number" remarks="???">

<constraints nullable="false"/>

</column>

<column name="USER\_VISIBLE" type="boolean" defaultValue="0" remarks="can user see this module?">

<constraints nullable="false"/>

</column>

</createTable>

<createSequence sequenceName="MOODLE\_MODULE\_SEQ" incrementBy="1" minValue="1" startValue="1" />

<addForeignKeyConstraint baseTableName="MOODLE\_MODULE"

baseColumnNames="SECTION\_ID"

constraintName="SECTION\_ID\_FK"

deferrable="true"

initiallyDeferred="true"

referencedTableName="MOODLE\_SECTION"

referencedColumnNames="ID"/>

</changeSet>

<changeSet author="Boyan.Gigov" id="create-MOODLE\_CONTENT">

<preConditions onFail="MARK\_RAN">

<not>

<tableExists tableName="MOODLE\_CONTENT"/>

</not>

</preConditions>

<createTable tableName="MOODLE\_CONTENT">

<column name="ID" type="number" remarks="primary key">

<constraints primaryKey="true" nullable="false"/>

</column>

<column name="MODULE\_ID" type="number" remarks="id of the module in MOODLE\_MODULE table">

<constraints nullable="false"/>

</column>

<column name="TYPE" type="varchar2(20 char)" remarks="type of the content, file or url">

<constraints nullable="false"/>

</column>

<column name="FILE\_NAME" type="varchar2(300 char)" remarks="name of the file as seen in moodle">

<constraints nullable="false"/>

</column>

<column name="FILE\_URL" type="varchar2(300 char)" remarks="url to download the file provided by moodle">

<constraints nullable="false"/>

</column>

<column name="MIME\_TYPE" type="varchar2(100 char)" remarks="mime type of the file provided by moodle">

</column>

<column name="AUTHOR" type="varchar2(200 char)" remarks="creator of the content in moodle">

</column>

</createTable>

<createSequence sequenceName="MOODLE\_CONTENT\_SEQ" incrementBy="1" minValue="1" startValue="1" />

<addForeignKeyConstraint baseTableName="MOODLE\_CONTENT"

baseColumnNames="MODULE\_ID"

constraintName="MODULE\_ID\_FK"

deferrable="true"

initiallyDeferred="true"

referencedTableName="MOODLE\_MODULE"

referencedColumnNames="ID"/>

</changeSet>

<changeSet author="Boyan.Gigov" id="create-MOODLE\_STATISTICS">

<preConditions onFail="MARK\_RAN">

<not>

<tableExists tableName="MOODLE\_STATISTICS"/>

</not>

</preConditions>

<createTable tableName="MOODLE\_STATISTICS">

<column name="ID" type="number" remarks="primary key">

<constraints primaryKey="true" nullable="false"/>

</column>

<column name="COURSE\_ID" type="number" remarks="course\_id as seen in MOODLE\_SECTION table">

<constraints nullable="false"/>

</column>

<column name="COURSE\_NAME" type="varchar2(300 char)" remarks="course\_name as seen in MOODLE\_SECTION table">

<constraints nullable="false"/>

</column>

<column name="FILES" type="number" remarks="number of files in this course"/>

<column name="URLS" type="number" remarks="number of urls in this course"/>

<column name="RESOURCES" type="number" remarks="number of resources in this course"/>

<column name="WIKIS" type="number" remarks="number of wikis in this course"/>

<column name="FORUM\_DISCUSSIONS" type="number" remarks="number of forum threads in this course"/>

</createTable>

<createSequence sequenceName="MOODLE\_STATISTICS\_SEQ" incrementBy="1" minValue="1" startValue="1" />

</changeSet>

</databaseChangeLog>

package all.service;

import all.component.diplomna.api.MoodleApi;

import all.component.diplomna.model.MoodleCourseSectionMO;

import all.component.diplomna.model.MoodleCourseStatisticsMO;

import com.google.gson.Gson;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import javax.ws.rs.\*;

import javax.ws.rs.core.Context;

import javax.ws.rs.core.MediaType;

import javax.ws.rs.core.Response;

import javax.servlet.http.HttpServletRequest;

import javax.ws.rs.core.StreamingOutput;

import org.apache.logging.log4j.Logger;

import org.apache.logging.log4j.LogManager;

import java.io.IOException;

import java.io.OutputStream;

import java.util.List;

@Service

@Path("/myService")

public class MoodleService {

private static Logger logger = LogManager.getLogger(MoodleService.class);

@Autowired

private MoodleApi moodleApi;

@Path("/updateDB")

@PATCH

@Consumes(MediaType.APPLICATION\_JSON)

public Response updateDB(@Context HttpServletRequest request) {

try {

moodleApi.updateDB();

return Response.ok().build();

} catch (Exception e) {

return buildServerError("updateDB", e);

}

}

@Path("/getAllCourses")

@GET

@Consumes(MediaType.APPLICATION\_JSON)

public Response getAllCouses(@Context HttpServletRequest request) {

try {

List<MoodleCourseSectionMO> courseSections = moodleApi.getAllCourses();

logSuccess("/getAllCourses", courseSections);

return Response.ok(courseSections, MediaType.APPLICATION\_JSON).build();

} catch (Exception e) {

return buildServerError("getAllCourses", e);

}

}

@Path("/getCourseData")

@GET

@Consumes(MediaType.APPLICATION\_JSON)

public Response getCourseData(@Context HttpServletRequest request,

@QueryParam("courseId") Long courseId) {

try {

List<MoodleCourseSectionMO> courseSections = moodleApi.getArchivedCourseFilesFromMoodle(courseId);

return Response.ok(courseSections, MediaType.APPLICATION\_JSON).build();

} catch (Exception e) {

return buildServerError("getCourseData", e);

}

}

@Path("/getCourseStatistics")

@GET

@Consumes(MediaType.APPLICATION\_JSON)

public Response getCourseStatistics(@Context HttpServletRequest request,

@QueryParam("courseId") Long courseId) {

try {

MoodleCourseStatisticsMO courseStatistics = moodleApi.getCourseStatistics(courseId);

logSuccess("/getCourseStatistics", courseStatistics);

return Response.ok(courseStatistics, MediaType.APPLICATION\_JSON).build();

} catch (Exception e) {

return buildServerError("getCourseStatistics", e);

}

}

@Path("/downloadCourseFiles")

@GET

@Consumes(MediaType.APPLICATION\_OCTET\_STREAM)

public Response downloadCourseFiles(@Context HttpServletRequest request,

@QueryParam("courseId") Long courseId) {

String fileName = "course\_"+courseId+".zip";

try {

return Response.ok(getArchivedCourseFilesAsStreamingOutput(courseId), MediaType.APPLICATION\_OCTET\_STREAM).header(

"content-disposition", "attachment; filename = " + fileName).build();

} catch (Exception e) {

return buildServerError("downloadCourseFiles", e);

}

}

private Response buildServerError(String requestName, Exception e) {

logger.error("Internal Server Error (500) in " + requestName, e);

return Response.serverError().build();

}

private StreamingOutput getArchivedCourseFilesAsStreamingOutput(Long courseId) {

return new StreamingOutput() {

@Override

public void write(OutputStream outputStream) throws IOException, WebApplicationException {

moodleApi.getArchivedCourseFilesFromMoodle(courseId, outputStream);

// moodleApi.testArchiving(courseId, outputStream);

outputStream.flush();

}

};

}

private void logSuccess(String requestName, Object returnObject) {

logger.info(requestName + " successfully returned" + new Gson().toJson(returnObject));

}

}

package all.persistence;

import javax.persistence.MappedSuperclass;

import java.io.Serializable;

@MappedSuperclass

public abstract class ModelBase implements Serializable {

private static final long serialVersionUID = -1234567890L;

protected ModelBase() {

}

public abstract Long getId();

public abstract void setId(Long id);

}

package all.persistence;

import org.apache.commons.dbcp2.BasicDataSource;

import org.apache.logging.log4j.Logger;

import org.apache.logging.log4j.LogManager;

public class DiplomnaDataSource extends BasicDataSource {

private static Logger logger = LogManager.getLogger(DiplomnaDataSource.class);

public DiplomnaDataSource() {

super();

}

// public synchronized void setPassword(String encryptedPassword) {

//

// }

}

package all.persistence.dao;

import all.persistence.ModelBase;

import org.springframework.transaction.annotation.Transactional;

import javax.persistence.EntityManager;

import javax.persistence.PersistenceContext;

import javax.persistence.Query;

public abstract class DiplomnaDaoImpl<T extends ModelBase> extends DaoBaseImpl<T> {

@PersistenceContext(unitName = "diplomnaEntityManager")

private EntityManager entityManager;

@Override

protected EntityManager getEntityManager() {

return entityManager;

}

protected void setEntityManager(EntityManager entityManager) {

this.entityManager = entityManager;

}

}

package all.persistence.dao;

import java.lang.reflect.ParameterizedType;

import java.util.List;

import all.persistence.ModelBase;

import javax.persistence.EntityManager;

import javax.persistence.FlushModeType;

import javax.persistence.Query;

public abstract class DaoBaseImpl<T extends ModelBase> implements DaoBase<T> {

public static final int NO\_MAX\_RESULT\_LIMIT = -1;

private static final Object[] NO\_PARAMS = null;

protected Class<T> entityClass;

@SuppressWarnings("unchecked")

public DaoBaseImpl() {

ParameterizedType genericSuperclass = (ParameterizedType) getClass().getGenericSuperclass();

entityClass = (Class<T>) genericSuperclass.getActualTypeArguments()[0];

}

@Override

public List<T> find(String query) {

return find(query, NO\_PARAMS, NO\_MAX\_RESULT\_LIMIT);

}

@Override

public List<T> find(String query, Object... params) {

return find(query, params, NO\_MAX\_RESULT\_LIMIT);

}

@SuppressWarnings("unchecked")

@Override

public List<T> find(final String query, final Object[] params, final int size) {

Query queryObj = getEntityManager().createQuery(query);

if (size != -1) {

queryObj.setMaxResults(size);

}

setParams(queryObj, params);

return queryObj.getResultList();

}

@Override

public List<T> findAll() {

return find("from " + entityClass.getSimpleName());

}

@Override

public T save (T modelBaseObj) {

return getEntityManager().merge(modelBaseObj);

}

@Override

public T saveAndCommit(T modelBaseObj) {

getEntityManager().setFlushMode(FlushModeType.COMMIT);

return getEntityManager().merge(modelBaseObj);

}

@Override

public T get(long id) {

List<T> result = find("from " + entityClass.getSimpleName() + " where id = " + id);

if (result == null || result.size() == 0) {

return null;

} else {

return result.get(0);

}

}

@Override

public void delete(long id) {

T modelBaseObj = get(id);

getEntityManager().remove(modelBaseObj);

}

@Override

public void deleteAll() {

String deleteStr = "delete from " + entityClass.getSimpleName();

Query query = getEntityManager().createQuery(deleteStr);

query.executeUpdate();

}

private void setParams(Query query, final Object[] params) {

if (params != null) {

for (int index = 0; index < params.length; index++) {

query.setParameter(index + 1, params[index]);

}

}

}

protected abstract EntityManager getEntityManager();

@Override

public void persist(T modelBaseObj) {

getEntityManager().persist(modelBaseObj);

}

@Override

public T persistAndFlush(T modelBaseObj) {

getEntityManager().persist(modelBaseObj);

getEntityManager().flush();

return modelBaseObj;

}

}

package all.persistence.dao;

import all.persistence.ModelBase;

import java.util.List;

public interface DaoBase<T extends ModelBase> {

List<T> find(String query);

List<T> find(String query, Object... params);

@SuppressWarnings("unchecked")

List<T> find(String query, Object[] params, int size);

List<T> findAll();

T save (T modelBaseObj);

T saveAndCommit(T modelBaseObj);

T get(long id);

void delete(long id);

void deleteAll();

void persist(T modelBaseObj);

T persistAndFlush(T modelBaseObj);

}

package all.component.diplomna.rest;

import all.component.diplomna.model.dto.\*;

import org.apache.logging.log4j.Logger;

import org.apache.logging.log4j.LogManager;

//import org.glassfish.jersey.jackson1.Jackson1Feature;

import org.glassfish.jersey.jackson.JacksonFeature;

import org.springframework.http.HttpStatus;

import org.springframework.stereotype.Component;

import javax.net.ssl.SSLContext;

import javax.net.ssl.TrustManager;

import javax.net.ssl.X509TrustManager;

import javax.ws.rs.client.Client;

import javax.ws.rs.client.ClientBuilder;

import javax.ws.rs.client.Entity;

import javax.ws.rs.client.Invocation;

import javax.ws.rs.core.MediaType;

import javax.ws.rs.core.Response;

import java.io.File;

import java.security.cert.X509Certificate;

@Component

public class MoodleRestClient {

private static Logger logger = LogManager.getLogger(MoodleRestClient.class);

private static final String baseUrl = "https://aptitude-learn.w3c.fmi.uni-sofia.bg";

private static final String loginUrl = "/login/token.php?service=moodle\_mobile\_app";

private static final String restApiUrl = "/webservice/rest/server.php";

private static String wsToken; // wsTokens expires after some time(days?), all downloaded files are 1KB with expired token

private static final String formatType = "&moodlewsrestformat=json";

private static String url;

private Client client;

// https://aptitude-learn.w3c.fmi.uni-sofia.bg/webservice/rest/server.php?wstoken=92b109d211f2b52d15d042b4f925f437&moodlewsrestformat=json&wsfunction=core\_calendar\_get\_action\_events\_by\_courses&courseids[0]=2

// https://aptitude-learn.w3c.fmi.uni-sofia.bg/webservice/rest/server.php?wstoken=92b109d211f2b52d15d042b4f925f437&moodlewsrestformat=json&wsfunction=core\_course\_get\_contents&courseid=2

MoodleRestClient() {

buildClient();

getWsToken();

setUrl();

}

private void buildClient() {

SSLContext sslcontext;

try {

sslcontext = SSLContext.getInstance("TLS");

sslcontext.init(null, new TrustManager[]{new X509TrustManager() {

public void checkClientTrusted(X509Certificate[] arg0, String arg1) {}

public void checkServerTrusted(X509Certificate[] arg0, String arg1) {}

public X509Certificate[] getAcceptedIssuers() { return new X509Certificate[0]; }

}}, new java.security.SecureRandom());

} catch (Exception e) {

logger.error("Failed to create client that ignores certificate exception. This may lead to failed requests if certificate isn't configured in some other way. ", e);

client = ClientBuilder.newBuilder().register(JacksonFeature.class).build();

return;

}

client = ClientBuilder.newBuilder().sslContext(sslcontext).hostnameVerifier((s1, s2) -> true).register(JacksonFeature.class).build();

}

private void getWsToken() {

Invocation.Builder builder = client.target(baseUrl + loginUrl).request(MediaType.APPLICATION\_FORM\_URLENCODED).accept(MediaType.APPLICATION\_JSON);

final String username = "boyan\_gigov";

final String password = "Boyan\_gigov@123";

MoodleWsTokenDTO wsTokenDTO = builder.post(Entity.entity("username=" + username + "&password=" + password, MediaType.APPLICATION\_FORM\_URLENCODED), MoodleWsTokenDTO.class);

wsToken = wsTokenDTO.getToken();

}

private void setUrl() {

url = baseUrl + restApiUrl + "?wstoken=" + wsToken + formatType;

}

private Invocation.Builder buildRequest(String path, MediaType requestType, MediaType acceptType) {

String requestUrl = url + path;

return client.target(requestUrl).request(requestType).accept(acceptType);

}

private Invocation.Builder buildDownloadRequest(String downloadUrl, MediaType requestType, MediaType acceptType) {

String requestUrl = downloadUrl + "&token=" + wsToken;

return client.target(requestUrl).request(requestType).accept(acceptType);

}

public MoodleCourseInfoDTO[] getCourses() {

Response response = buildRequest("&wsfunction=core\_course\_get\_courses", MediaType.TEXT\_PLAIN\_TYPE, MediaType.APPLICATION\_JSON\_TYPE).get(Response.class);

if (response.getStatus() == HttpStatus.OK.value()) {

return response.readEntity(MoodleCourseInfoDTO[].class);

} else {

// String responseString = response.readEntity(String.class);

logger.error("getCourses response: " + response.toString());

// logger.error("getCourses response: " + response.getStatus() + " " + responseString);

return null;

}

}

public MoodleCourseSectionDTO[] getCourseContents(Long courseId) {

Response response = buildRequest("&wsfunction=core\_course\_get\_contents&courseid=" + courseId, MediaType.TEXT\_PLAIN\_TYPE, MediaType.APPLICATION\_JSON\_TYPE).get(Response.class);

if (response.getStatus() == HttpStatus.OK.value()) {

return response.readEntity(MoodleCourseSectionDTO[].class);

} else {

logger.error("getCourseContents response: " + response.toString());

return null;

}

}

public File getFile(String fileUrl, String fileName) {

Response response = buildDownloadRequest(fileUrl, MediaType.TEXT\_PLAIN\_TYPE, MediaType.APPLICATION\_JSON\_TYPE).get(Response.class);

if (response.getStatus() == HttpStatus.OK.value()) {

File downloadedFile = response.readEntity(File.class);

String absPath = downloadedFile.getAbsolutePath();

String renamedFilePath = absPath.replace(absPath.substring(absPath.lastIndexOf('\\')+1), fileName);

if (downloadedFile.renameTo(new File(renamedFilePath))) {

downloadedFile = new File(renamedFilePath);

} else {

logger.warn("failed to rename file " + downloadedFile.getAbsolutePath() + " to " + renamedFilePath);

int indexOfExtension = renamedFilePath.lastIndexOf(".");

String fileExtension = renamedFilePath.substring(indexOfExtension);

for (int attempt = 1; attempt < 20; attempt++) {

String renamedFilePathAttepmt = renamedFilePath.substring(0, indexOfExtension) + "(" + attempt + ")" + fileExtension;

logger.info("retrying to rename file " + downloadedFile.getAbsolutePath() + " to " + renamedFilePathAttepmt);

if (downloadedFile.renameTo(new File(renamedFilePathAttepmt))) {

downloadedFile = new File(renamedFilePathAttepmt);

return downloadedFile;

}

}

logger.error("Failed to properly name the file for " + fileName);

}

return downloadedFile;

} else {

logger.error("buildGetFileRequest response: " + response.toString());

return null;

}

}

public MoodleForumDTO[] getForums(long courseId) {

// this is working on the presumption that each forum id is equal to its course id, which was the case with test data

Response response = buildRequest("&wsfunction=mod\_forum\_get\_forums\_by\_courses&courseids[0]=" + courseId, MediaType.TEXT\_PLAIN\_TYPE, MediaType.APPLICATION\_JSON\_TYPE).get(Response.class);

if (response.getStatus() == HttpStatus.OK.value()) {

return response.readEntity(MoodleForumDTO[].class);

} else {

logger.error("getForums response: " + response.toString());

return null;

}

}

public MoodleResourceDTO[] getResources(long courseId) {

Response response = buildRequest("&wsfunction=mod\_resource\_get\_resources\_by\_courses&courseids[0]=" + courseId, MediaType.TEXT\_PLAIN\_TYPE, MediaType.APPLICATION\_JSON\_TYPE).get(Response.class);

if (response.getStatus() == HttpStatus.OK.value()) {

return response.readEntity(MoodleResourceArrayDTO.class).getResources();

} else {

logger.error("getForums response: " + response.toString());

return new MoodleResourceDTO[0];

}

}

public MoodleWikiDTO[] getWikis(long courseId) {

Response response = buildRequest("&wsfunction=mod\_wiki\_get\_wikis\_by\_courses&courseids[0]=" + courseId, MediaType.TEXT\_PLAIN\_TYPE, MediaType.APPLICATION\_JSON\_TYPE).get(Response.class);

if (response.getStatus() == HttpStatus.OK.value()) {

return response.readEntity(MoodleWikiArrayDTO.class).getWikis();

} else {

logger.error("getForums response: " + response.toString());

return new MoodleWikiDTO[0];

}

}

}

package all.component.diplomna.model;

import all.persistence.ModelBase;

import com.fasterxml.jackson.annotation.JsonIdentityInfo;

import com.fasterxml.jackson.annotation.JsonIgnoreProperties;

import com.fasterxml.jackson.annotation.JsonInclude;

import com.fasterxml.jackson.annotation.ObjectIdGenerators;

import javax.persistence.\*;

import java.util.HashSet;

import java.util.Set;

@Entity

@Table(name = "MOODLE\_MODULE")

@JsonInclude(JsonInclude.Include.NON\_NULL)

@JsonIgnoreProperties(ignoreUnknown = true)

@JsonIdentityInfo(generator = ObjectIdGenerators.PropertyGenerator.class, property = "id")

public class MoodleModuleMO extends ModelBase {

private static final long serialVersionUID = -12345678901L;

@Id

@Column(name = "ID")

@GeneratedValue(strategy = GenerationType.AUTO, generator = "MoodleModuleGenerator")

@SequenceGenerator(name = "MoodleModuleGenerator", sequenceName = "MOODLE\_MODULE\_SEQ", allocationSize = 1)

private Long id;

@ManyToOne(fetch = FetchType.LAZY)

@JoinColumn(name = "SECTION\_ID", referencedColumnName = "ID", nullable = false, unique = true)

private MoodleCourseSectionMO courseSection;

@OneToMany(fetch = FetchType.EAGER, mappedBy = "module", cascade = {CascadeType.ALL}, orphanRemoval = true)

private Set<MoodleContentMO> contents;

@Column(name = "NAME", nullable = false)

private String name;

@Column(name = "VISIBLE", nullable = false)

private Integer visible;

@Column(name = "USER\_VISIBLE", nullable = false)

private Boolean userVisible;

@Transient // will not be saved to the DB

private String modName;

@Override

public Long getId() {

return id;

}

@Override

public void setId(Long id) {

this.id = id;

}

public Set<MoodleContentMO> getContents() {

return contents;

}

public MoodleCourseSectionMO getCourseSection() {

return courseSection;

}

public void setCourseSection(MoodleCourseSectionMO courseSection) {

this.courseSection = courseSection;

}

public void setContents(Set<MoodleContentMO> contents) {

if (this.contents == null) {

this.contents = new HashSet<>();

}

this.contents.addAll(contents);

}

public void addContent(MoodleContentMO content) {

if (this.contents == null) {

this.contents = new HashSet<>();

}

this.contents.add(content);

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public Integer getVisible() {

return visible;

}

public void setVisible(Integer visible) {

this.visible = visible;

}

public Boolean getUserVisible() {

return userVisible;

}

public void setUserVisible(Boolean userVisible) {

this.userVisible = userVisible;

}

public String getModName() {

return modName;

}

public void setModName(String modName) {

this.modName = modName;

}

}

package all.component.diplomna.model;

import all.persistence.ModelBase;

import com.fasterxml.jackson.annotation.JsonIdentityInfo;

import com.fasterxml.jackson.annotation.JsonIgnoreProperties;

import com.fasterxml.jackson.annotation.JsonInclude;

import com.fasterxml.jackson.annotation.ObjectIdGenerators;

import javax.persistence.\*;

@Entity

@Table(name = "MOODLE\_STATISTICS")

@JsonInclude(JsonInclude.Include.NON\_NULL)

@JsonIgnoreProperties(ignoreUnknown = true)

@JsonIdentityInfo(generator = ObjectIdGenerators.PropertyGenerator.class, property = "id")

public class MoodleCourseStatisticsMO extends ModelBase {

private static final long serialVersionUID = -12345678902L;

@Id

@Column(name = "ID")

@GeneratedValue(strategy = GenerationType.AUTO, generator = "MoodleStatisticsGenerator")

@SequenceGenerator(name = "MoodleStatisticsGenerator", sequenceName = "MOODLE\_STATISTICS\_SEQ", allocationSize = 1)

private Long id;

@Column(name = "COURSE\_ID", nullable = false)

private Long courseId;

@Column(name = "COURSE\_NAME", nullable = false)

private String courseName;

@Column(name = "FILES")

private Integer numberOfFiles;

@Column(name = "URLS")

private Integer numberOfUrls;

@Column(name = "RESOURCES")

private Integer numberOfResources;

@Column(name = "WIKIS")

private Integer numberOfWikis;

@Column(name = "FORUM\_DISCUSSIONS")

private Integer numberOfForumDiscussions;

@Override

public Long getId() {

return id;

}

@Override

public void setId(Long id) {

this.id = id;

}

public static long getSerialVersionUID() {

return serialVersionUID;

}

public Long getCourseId() {

return courseId;

}

public void setCourseId(Long courseId) {

this.courseId = courseId;

}

public String getCourseName() {

return courseName;

}

public void setCourseName(String courseName) {

this.courseName = courseName;

}

public Integer getNumberOfFiles() {

return numberOfFiles;

}

public void setNumberOfFiles(Integer numberOfFiles) {

this.numberOfFiles = numberOfFiles;

}

public Integer getNumberOfUrls() {

return numberOfUrls;

}

public void setNumberOfUrls(Integer numberOfUrls) {

this.numberOfUrls = numberOfUrls;

}

public Integer getNumberOfResources() {

return numberOfResources;

}

public void setNumberOfResources(Integer numberOfResources) {

this.numberOfResources = numberOfResources;

}

public Integer getNumberOfWikis() {

return numberOfWikis;

}

public void setNumberOfWikis(Integer numberOfWikis) {

this.numberOfWikis = numberOfWikis;

}

public Integer getNumberOfForumDiscussions() {

return numberOfForumDiscussions;

}

public void setNumberOfForumDiscussions(Integer numberOfForumDiscussions) {

this.numberOfForumDiscussions = numberOfForumDiscussions;

}

}

package all.component.diplomna.model;

import all.persistence.ModelBase;

import com.fasterxml.jackson.annotation.JsonIdentityInfo;

import com.fasterxml.jackson.annotation.JsonIgnoreProperties;

import com.fasterxml.jackson.annotation.JsonInclude;

import com.fasterxml.jackson.annotation.ObjectIdGenerators;

import javax.persistence.\*;

import java.util.HashSet;

import java.util.Set;

@Entity

@Table(name = "MOODLE\_SECTION")

@JsonInclude(JsonInclude.Include.NON\_NULL)

@JsonIgnoreProperties(ignoreUnknown = true)

@JsonIdentityInfo(generator = ObjectIdGenerators.PropertyGenerator.class, property = "id")

public class MoodleCourseSectionMO extends ModelBase {

private static final long serialVersionUID = -12345678903L;

public MoodleCourseSectionMO() {

}

public MoodleCourseSectionMO(Long courseId, String courseName) {

this.courseId = courseId;

this.courseName = courseName;

}

@Id

@Column(name = "ID")

@GeneratedValue(strategy = GenerationType.AUTO, generator = "MoodleCourseSectionGenerator")

@SequenceGenerator(name = "MoodleCourseSectionGenerator", sequenceName = "MOODLE\_SECTION\_SEQ", allocationSize = 1)

private Long id;

@OneToMany(fetch = FetchType.EAGER, mappedBy = "courseSection", cascade = {CascadeType.ALL}, orphanRemoval = true)

private Set<MoodleModuleMO> modules;

@Column(name = "COURSE\_ID", nullable = false)

private Long courseId;

@Column(name = "SECTION\_ID", nullable = false, unique = true)

private Long sectionId;

@Column(name = "COURSE\_NAME", nullable = false)

private String courseName;

@Column(name = "SECTION\_NAME", nullable = false)

private String sectionName;

@Column(name = "USER\_VISIBLE", nullable = false)

private Boolean uservisible;

@Override

public Long getId() {

return id;

}

@Override

public void setId(Long id) {

this.id = id;

}

public Set<MoodleModuleMO> getModules() {

return modules;

}

public void setModules(Set<MoodleModuleMO> modules) {

if (this.modules == null) {

this.modules = new HashSet<>();

} else {

this.modules.addAll(modules);

}

}

public void addMoudle(MoodleModuleMO module) {

if (this.modules == null) {

this.modules = new HashSet<>();

} else {

this.modules.add(module);

}

}

public Long getCourseId() {

return courseId;

}

public void setCourseId(Long courseId) {

this.courseId = courseId;

}

public Long getSectionId() {

return sectionId;

}

public void setSectionId(Long sectionId) {

this.sectionId = sectionId;

}

public String getCourseName() {

return courseName;

}

public void setCourseName(String courseName) {

this.courseName = courseName;

}

public String getSectionName() {

return sectionName;

}

public void setSectionName(String sectionName) {

this.sectionName = sectionName;

}

public Boolean getUservisible() {

return uservisible;

}

public void setUservisible(Boolean uservisible) {

this.uservisible = uservisible;

}

}

package all.component.diplomna.model;

import all.persistence.ModelBase;

import com.fasterxml.jackson.annotation.JsonIdentityInfo;

import com.fasterxml.jackson.annotation.ObjectIdGenerators;

import com.fasterxml.jackson.annotation.JsonIgnoreProperties;

import com.fasterxml.jackson.annotation.JsonInclude;

//import org.codehaus.jackson.annotate.JsonIgnoreProperties;

//import org.codehaus.jackson.map.annotate.JsonSerialize;

import javax.persistence.\*;

@Entity

@Table(name = "MOODLE\_CONTENT")

@JsonInclude(JsonInclude.Include.NON\_NULL)

@JsonIgnoreProperties(ignoreUnknown = true)

@JsonIdentityInfo(generator = ObjectIdGenerators.PropertyGenerator.class, property = "id")

public class MoodleContentMO extends ModelBase {

private static final long serialVersionUID = -12345678904L;

@Id

@Column(name = "ID", unique = true)

@GeneratedValue(strategy = GenerationType.AUTO, generator = "MoodleContentGenerator")

@SequenceGenerator(name = "MoodleContentGenerator", sequenceName = "MOODLE\_CONTENT\_SEQ", allocationSize = 1)

private Long id;

@ManyToOne(fetch = FetchType.LAZY)

@JoinColumn(name = "MODULE\_ID", referencedColumnName = "ID", nullable = false, unique = true)

private MoodleModuleMO module;

@Column(name = "TYPE", nullable = false)

private String type; // "file" or "url"

@Column(name = "FILE\_NAME", nullable = false)

private String fileName;

@Column(name = "FILE\_URL", nullable = false)

private String fileurl;

@Column(name = "MIME\_TYPE")

private String mimetype; // null if type is "url"

@Column(name = "AUTHOR")

private String author;

@Override

public Long getId() {

return id;

}

@Override

public void setId(Long id) {

this.id = id;

}

public MoodleModuleMO getModule() {

return module;

}

public void setModule(MoodleModuleMO module) {

this.module = module;

}

public String getType() {

return type;

}

public void setType(String type) {

this.type = type;

}

public String getFileName() {

return fileName;

}

public void setFileName(String fileName) {

this.fileName = fileName;

}

// public Long getFilesize() {

// return filesize;

// }

//

// public void setFilesize(Long filesize) {

// this.filesize = filesize;

// }

public String getFileurl() {

return fileurl;

}

public void setFileurl(String fileurl) {

this.fileurl = fileurl;

}

public String getMimetype() {

return mimetype;

}

public void setMimetype(String mimetype) {

this.mimetype = mimetype;

}

public String getAuthor() {

return author;

}

public void setAuthor(String author) {

this.author = author;

}

}

package all.component.diplomna.model.dto;

public class MoodleWsTokenDTO {

private String token;

private String privatetoken;

public String getToken() {

return token;

}

public void setToken(String token) {

this.token = token;

}

public String getPrivatetoken() {

return privatetoken;

}

public void setPrivatetoken(String privatetoken) {

this.privatetoken = privatetoken;

}

}

package all.component.diplomna.model.dto;

import com.fasterxml.jackson.annotation.JsonIgnoreProperties;

import com.fasterxml.jackson.annotation.JsonInclude;

@JsonInclude(JsonInclude.Include.NON\_NULL)

@JsonIgnoreProperties(ignoreUnknown = true)

public class MoodleWikiDTO {

private Long id;

MoodleWikiDTO() {

}

public Long getId() {

return id;

}

public void setId(Long id) {

this.id = id;

}

}

package all.component.diplomna.model.dto;

import com.fasterxml.jackson.annotation.JsonIgnoreProperties;

import com.fasterxml.jackson.annotation.JsonInclude;

@JsonInclude(JsonInclude.Include.NON\_NULL)

@JsonIgnoreProperties(ignoreUnknown = true)

public class MoodleWikiArrayDTO {

private MoodleWikiDTO[] wikis;

MoodleWikiArrayDTO() {

}

public MoodleWikiDTO[] getWikis() {

return wikis;

}

public void setWikis(MoodleWikiDTO[] wikis) {

this.wikis = wikis;

}

}

package all.component.diplomna.model.dto;

import com.fasterxml.jackson.annotation.JsonIgnoreProperties;

import com.fasterxml.jackson.annotation.JsonInclude;

@JsonInclude(JsonInclude.Include.NON\_NULL)

@JsonIgnoreProperties(ignoreUnknown = true)

public class MoodleResourceDTO {

private long id;

MoodleResourceDTO() {

}

public long getId() {

return id;

}

public void setId(long id) {

this.id = id;

}

}

package all.component.diplomna.model.dto;

import com.fasterxml.jackson.annotation.JsonIgnoreProperties;

import com.fasterxml.jackson.annotation.JsonInclude;

@JsonInclude(JsonInclude.Include.NON\_NULL)

@JsonIgnoreProperties(ignoreUnknown = true)

public class MoodleResourceArrayDTO {

private MoodleResourceDTO[] resources;

MoodleResourceArrayDTO() {

}

public MoodleResourceDTO[] getResources() {

return resources;

}

public void setResources(MoodleResourceDTO[] resources) {

this.resources = resources;

}

}

package all.component.diplomna.model.dto;

import com.fasterxml.jackson.annotation.JsonIgnoreProperties;

import com.fasterxml.jackson.annotation.JsonInclude;

@JsonInclude(JsonInclude.Include.NON\_NULL)

@JsonIgnoreProperties(ignoreUnknown = true)

public class MoodleModuleDTO {

private Long id;

private String name;

private Integer visible;

private Boolean uservisible;

private String modname;

private MoodleContentDTO[] contents;

MoodleModuleDTO() {

}

public Long getId() {

return id;

}

public void setId(Long id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public Integer getVisible() {

return visible;

}

public void setVisible(Integer visible) {

this.visible = visible;

}

public Boolean isUserVisible() {

return uservisible;

}

public void setUservisible(Boolean uservisible) {

this.uservisible = uservisible;

}

public String getModname() {

return modname;

}

public void setModname(String modname) {

this.modname = modname;

}

public MoodleContentDTO[] getContents() {

return contents != null ? contents : new MoodleContentDTO[0];

}

public void setContents(MoodleContentDTO[] contents) {

this.contents = contents;

}

}

package all.component.diplomna.model.dto;

import com.fasterxml.jackson.annotation.JsonIgnoreProperties;

import com.fasterxml.jackson.annotation.JsonInclude;

@JsonInclude(JsonInclude.Include.NON\_NULL)

@JsonIgnoreProperties(ignoreUnknown = true)

public class MoodleForumDTO {

private Integer numdiscussions;

MoodleForumDTO() {

}

public Integer getNumdiscussions() {

return numdiscussions;

}

public void setNumdiscussions(Integer numdiscussions) {

this.numdiscussions = numdiscussions;

}

}

package all.component.diplomna.model.dto;

import com.fasterxml.jackson.annotation.JsonIgnoreProperties;

import com.fasterxml.jackson.annotation.JsonInclude;

import java.util.List;

@JsonInclude(JsonInclude.Include.NON\_NULL)

@JsonIgnoreProperties(ignoreUnknown = true)

public class MoodleCourseSectionDTO {

private Long id;

private String name;

private Integer visible;

private Boolean uservisible;

private String summary;

private MoodleModuleDTO[] modules;

MoodleCourseSectionDTO() {

}

public Long getId() {

return id;

}

public void setId(Long id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public Integer getVisible() {

return visible;

}

public void setVisible(Integer visible) {

this.visible = visible;

}

public Boolean getUservisible() {

return uservisible;

}

public void setUservisible(Boolean uservisible) {

this.uservisible = uservisible;

}

public String getSummary() {

return summary;

}

public void setSummary(String summary) {

this.summary = summary;

}

public MoodleModuleDTO[] getModules() {

return modules != null ? modules : new MoodleModuleDTO[0];

}

public void setModules(MoodleModuleDTO[] modules) {

this.modules = modules;

}

}

package all.component.diplomna.model.dto;

import com.fasterxml.jackson.annotation.JsonIgnoreProperties;

import com.fasterxml.jackson.annotation.JsonInclude;

@JsonInclude(JsonInclude.Include.NON\_NULL)

@JsonIgnoreProperties(ignoreUnknown = true)

public class MoodleCourseInfoDTO {

private Long id;

private String displayname;

private String summary; // html format

private int newsitems;

private int numsections;

public Long getId() {

return id;

}

public void setId(Long id) {

this.id = id;

}

public String getDisplayname() {

return displayname;

}

public void setDisplayname(String displayname) {

this.displayname = displayname;

}

public String getSummary() {

return summary;

}

public void setSummary(String summary) {

this.summary = summary;

}

public int getNewsitems() {

return newsitems;

}

public void setNewsitems(int newsitems) {

this.newsitems = newsitems;

}

public int getNumsections() {

return numsections;

}

public void setNumsections(int numsections) {

this.numsections = numsections;

}

}

package all.component.diplomna.model.dto;

import com.fasterxml.jackson.annotation.JsonIgnoreProperties;

import com.fasterxml.jackson.annotation.JsonInclude;

@JsonInclude(JsonInclude.Include.NON\_NULL)

@JsonIgnoreProperties(ignoreUnknown = true)

public class MoodleContentDTO {

private Long id;

private String type; // "file" or "url"

private String filename;

private Long filesize;

private String fileurl;

private String mimetype; // null if type is "url"

private String author;

MoodleContentDTO() {

}

public Long getId() {

return id;

}

public void setId(Long id) {

this.id = id;

}

public String getType() {

return type;

}

public void setType(String type) {

this.type = type;

}

public String getFilename() {

return filename;

}

public void setFilename(String filename) {

this.filename = filename;

}

public Long getFilesize() {

return filesize;

}

public void setFilesize(Long filesize) {

this.filesize = filesize;

}

public String getFileurl() {

return fileurl;

}

public void setFileurl(String fileurl) {

this.fileurl = fileurl;

}

public String getMimetype() {

return mimetype;

}

public void setMimetype(String mimetype) {

this.mimetype = mimetype;

}

public String getAuthor() {

return author;

}

public void setAuthor(String author) {

this.author = author;

}

}

package all.component.diplomna.dao;

import all.component.diplomna.dao.interfaces.MoodleModuleDao;

import all.component.diplomna.model.MoodleModuleMO;

import all.persistence.dao.DiplomnaDaoImpl;

import org.springframework.stereotype.Repository;

import org.springframework.transaction.annotation.Transactional;

import javax.persistence.NoResultException;

import javax.persistence.Query;

@Transactional("diplomnaTransactionManager")

@Repository

public class MoodleModuleDaoImpl extends DiplomnaDaoImpl<MoodleModuleMO> implements MoodleModuleDao {

private final String COURSE\_MODULE\_TABLE = entityClass.getSimpleName();

// @Override

// public MoodleModuleMO saveCourseInfo(MoodleModuleMO mo) {

// MoodleModuleMO dbValue = getModuelByModuleId(mo.getId());

// if (dbValue != null) {

// // update values

// return super.save(dbValue);

// } else {

// return super.save(mo);

// }

// }

//

// @Override

// public MoodleModuleMO getModuelByModuleId(Long moduleId) {

// String selectStr = "from " + COURSE\_SECTION\_TABLE + " mo" +

// " where mo.id = (:moduleId)";

// Query query = getEntityManager().createQuery(selectStr);

// query.setParameter("moduleId", moduleId);

// try {

// MoodleModuleMO retVal = (MoodleModuleMO) query.getSingleResult();

// return retVal;

// } catch (NoResultException e) {

// return null;

// }

// }

// no save method, should only be saved (via Hibernate) when a MoodleModuleMO is saved

@Override

public void deleteAllModules() {

super.deleteAll();

}

}

package all.component.diplomna.dao;

import all.component.diplomna.dao.interfaces.MoodleCourseStatisticsDao;

import all.component.diplomna.model.MoodleCourseStatisticsMO;

import all.persistence.dao.DiplomnaDaoImpl;

import org.springframework.stereotype.Repository;

import org.springframework.transaction.annotation.Transactional;

import javax.persistence.NoResultException;

import javax.persistence.Query;

import java.util.List;

@Transactional("diplomnaTransactionManager")

@Repository

public class MoodleCourseStatisticsDaoImpl extends DiplomnaDaoImpl<MoodleCourseStatisticsMO> implements MoodleCourseStatisticsDao {

private final String COURSE\_STATISTICS\_TABLE = entityClass.getSimpleName();

@Override

public MoodleCourseStatisticsMO saveCourseStatistics(MoodleCourseStatisticsMO mo) {

return super.save(mo);

}

@Override

public void deleteAllStatistics() {

super.deleteAll();

}

@Override

public MoodleCourseStatisticsMO getCourseStatisticsByCourseId(Long courseId) {

String selectStr = "from " + COURSE\_STATISTICS\_TABLE + " mo" +

" where mo.courseId = (:courseId)";

Query query = getEntityManager().createQuery(selectStr);

query.setParameter("courseId", courseId);

try {

MoodleCourseStatisticsMO retVal = (MoodleCourseStatisticsMO) query.getSingleResult();

return retVal;

} catch (NoResultException e) {

return null;

}

}

}

package all.component.diplomna.dao;

import all.component.diplomna.dao.interfaces.MoodleCourseSectionDao;

import all.component.diplomna.model.MoodleCourseSectionMO;

import all.persistence.dao.DiplomnaDaoImpl;

import org.springframework.stereotype.Repository;

import org.springframework.transaction.annotation.Transactional;

import javax.persistence.NoResultException;

import javax.persistence.Query;

import java.util.ArrayList;

import java.util.List;

@Transactional("diplomnaTransactionManager")

@Repository

public class MoodleCourseSectionDaoImpl extends DiplomnaDaoImpl<MoodleCourseSectionMO> implements MoodleCourseSectionDao {

private final String COURSE\_SECTION\_TABLE = entityClass.getSimpleName();

@Override

public MoodleCourseSectionMO saveCourseSection(MoodleCourseSectionMO mo) {

return super.save(mo);

}

@Override

public void deleteAllSections() {

super.deleteAll();

}

@Override

public List<MoodleCourseSectionMO> getAllCourses() {

String selectStr = "select distinct courseId, courseName from " + COURSE\_SECTION\_TABLE;

Query query = getEntityManager().createQuery(selectStr);

try {

List<MoodleCourseSectionMO> retVal = new ArrayList<>();

((List<Object[]>) query.getResultList()).forEach(element -> retVal.add(new MoodleCourseSectionMO((Long) element[0], (String) element[1])));

return retVal;

} catch (NoResultException e) {

return null;

}

}

@Override

public List<MoodleCourseSectionMO> getAllCourseSectionsByCourseId(Long courseId) {

String selectStr = "from " + COURSE\_SECTION\_TABLE + " mo" +

" where mo.courseId = (:courseId)";

Query query = getEntityManager().createQuery(selectStr);

query.setParameter("courseId", courseId);

try {

List<MoodleCourseSectionMO> retVal = query.getResultList();

return retVal;

} catch (NoResultException e) {

return null;

}

}

@Override

public MoodleCourseSectionMO getCourseSection(Long courseId, Long sectionId) {

String selectStr = "from " + COURSE\_SECTION\_TABLE + " mo" +

" where mo.courseId = (:courseId) and mo.sectionId = (:sectionId)";

Query query = getEntityManager().createQuery(selectStr);

query.setParameter("courseId", courseId);

query.setParameter("sectionId", sectionId);

try {

MoodleCourseSectionMO retVal = (MoodleCourseSectionMO) query.getSingleResult();

return retVal;

} catch (NoResultException e) {

return null;

}

}

}

package all.component.diplomna.dao;

import all.component.diplomna.dao.interfaces.MoodleContentDao;

import all.component.diplomna.model.MoodleContentMO;

import all.persistence.dao.DiplomnaDaoImpl;

import org.springframework.stereotype.Repository;

import org.springframework.transaction.annotation.Transactional;

@Transactional("diplomnaTransactionManager")

@Repository

public class MoodleContentDaoImpl extends DiplomnaDaoImpl<MoodleContentMO> implements MoodleContentDao {

// no save method, should only be saved (via Hibernate) when a MoodleModuleMO is saved

@Override

public void deleteAllContents() {

super.deleteAll();

}

}

package all.component.diplomna.dao.interfaces;

import all.component.diplomna.model.MoodleModuleMO;

public interface MoodleModuleDao {

// MoodleModuleMO saveCourseInfo(MoodleModuleMO mo);

//

// MoodleModuleMO getModuelByModuleId(Long courseId);

void deleteAllModules();

}

package all.component.diplomna.dao.interfaces;

import all.component.diplomna.model.MoodleCourseSectionMO;

import all.component.diplomna.model.MoodleCourseStatisticsMO;

import java.util.List;

public interface MoodleCourseStatisticsDao {

MoodleCourseStatisticsMO saveCourseStatistics(MoodleCourseStatisticsMO mo);

void deleteAllStatistics();

MoodleCourseStatisticsMO getCourseStatisticsByCourseId(Long courseId);

}

package all.component.diplomna.dao.interfaces;

import all.component.diplomna.model.MoodleCourseSectionMO;

import java.util.List;

public interface MoodleCourseSectionDao {

MoodleCourseSectionMO saveCourseSection(MoodleCourseSectionMO mo);

void deleteAllSections();

List<MoodleCourseSectionMO> getAllCourses();

List<MoodleCourseSectionMO> getAllCourseSectionsByCourseId(Long courseId);

MoodleCourseSectionMO getCourseSection(Long courseId, Long sectionId);

}

package all.component.diplomna.dao.interfaces;

public interface MoodleContentDao {

void deleteAllContents();

}

package all.component.diplomna.converter;

import org.apache.logging.log4j.Logger;

import org.apache.logging.log4j.LogManager;

import org.springframework.stereotype.Component;

import java.io.\*;

import java.util.List;

import java.util.Map;

import java.util.zip.ZipEntry;

import java.util.zip.ZipOutputStream;

@Component

public class MoodleFileArchiver {

private static Logger logger = LogManager.getLogger(MoodleFileArchiver.class);

public synchronized static void compressFilesToZip(List<File> fileList, OutputStream outputStream) {

try (ZipOutputStream zipOutputStream = new ZipOutputStream(outputStream)) {

for (File file : fileList) {

try (FileInputStream inputStream = new FileInputStream(file)) {

ZipEntry zipEntry = new ZipEntry(file.getName());

zipOutputStream.putNextEntry(zipEntry);

int length;

byte[] buffer = new byte[1024];

while ((length = inputStream.read(buffer)) > 0) {

zipOutputStream.write(buffer, 0, length);

}

// closes a single zipEntry, not the entire stream, so it's necessary

// to close despite try-with-resources

zipOutputStream.closeEntry();

} catch (Exception e) {

logger.error("Failed to compress file [1]", e);

throw e;

}

}

} catch (Exception e) {

logger.error("Failed to compress file [2]", e);

}

}

}

package all.component.diplomna.converter;

import all.component.diplomna.model.MoodleContentMO;

import all.component.diplomna.model.MoodleCourseSectionMO;

import all.component.diplomna.model.MoodleCourseStatisticsMO;

import all.component.diplomna.model.MoodleModuleMO;

import all.component.diplomna.model.dto.MoodleContentDTO;

import all.component.diplomna.model.dto.MoodleCourseSectionDTO;

import all.component.diplomna.model.dto.MoodleModuleDTO;

import org.apache.logging.log4j.Logger;

import org.apache.logging.log4j.LogManager;

import org.springframework.stereotype.Component;

import java.util.ArrayList;

import java.util.Arrays;

import java.util.List;

@Component

public class MoodleConverter {

private static Logger logger = LogManager.getLogger(MoodleConverter.class);

public MoodleCourseSectionMO convertDtoToMo(MoodleCourseSectionDTO section, Long courseId, String courseDisplayName) {

MoodleCourseSectionMO mo = new MoodleCourseSectionMO();

mo.setCourseId(courseId);

mo.setSectionId(section.getId());

mo.setCourseName(courseDisplayName);

mo.setSectionName(section.getName());

mo.setUservisible(section.getUservisible());

if (section.getModules() != null) {

Arrays.stream(section.getModules()).forEach(moduleDTO -> mo.addMoudle(convertDtoToMo(moduleDTO, mo)));

}

return mo;

}

public MoodleModuleMO convertDtoToMo(MoodleModuleDTO moduleDTO, MoodleCourseSectionMO sectionMO) {

MoodleModuleMO mo = new MoodleModuleMO();

mo.setName(moduleDTO.getName());

mo.setVisible(moduleDTO.getVisible());

mo.setUserVisible(moduleDTO.isUserVisible());

mo.setCourseSection(sectionMO);

mo.setModName(moduleDTO.getModname());

if (moduleDTO.getContents() != null) {

Arrays.stream(moduleDTO.getContents()).forEach(contentDTO -> mo.addContent(convertDtoToMo(contentDTO, mo)));

}

return mo;

}

public MoodleContentMO convertDtoToMo(MoodleContentDTO contentDTO, MoodleModuleMO moduleMO) {

MoodleContentMO mo = new MoodleContentMO();

mo.setType(contentDTO.getType());

mo.setFileName(contentDTO.getFilename());

mo.setFileurl(contentDTO.getFileurl());

mo.setMimetype(contentDTO.getMimetype());

mo.setAuthor(contentDTO.getAuthor());

mo.setModule(moduleMO);

return mo;

}

}

package all.component.diplomna.api;

import all.component.diplomna.converter.MoodleConverter;

import all.component.diplomna.converter.MoodleFileArchiver;

import all.component.diplomna.dao.interfaces.MoodleModuleDao;

import all.component.diplomna.dao.interfaces.MoodleCourseStatisticsDao;

import all.component.diplomna.dao.interfaces.MoodleContentDao;

import all.component.diplomna.dao.interfaces.MoodleCourseSectionDao;

import all.component.diplomna.model.MoodleCourseSectionMO;

import all.component.diplomna.model.MoodleCourseStatisticsMO;

import all.component.diplomna.model.MoodleModuleMO;

import all.component.diplomna.model.dto.MoodleCourseInfoDTO;

import all.component.diplomna.model.dto.MoodleCourseSectionDTO;

import all.component.diplomna.rest.MoodleRestClient;

import org.apache.logging.log4j.Logger;

import org.apache.logging.log4j.LogManager;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Component;

import org.springframework.transaction.annotation.Transactional;

import java.io.File;

import java.io.OutputStream;

import java.util.\*;

import java.util.concurrent.atomic.AtomicInteger;

@Component

public class MoodleApiImpl implements MoodleApi {

private static final Logger logger = LogManager.getLogger(MoodleApiImpl.class);

private static final String FILE = "file";

@Autowired

private MoodleCourseSectionDao moodleCourseSectionDao;

@Autowired

private MoodleModuleDao moodleModuleDao;

@Autowired

private MoodleContentDao moodleContentDao;

@Autowired

private MoodleCourseStatisticsDao moodleCourseStatisticsDao;

@Autowired

private MoodleRestClient moodleRestClient;

@Autowired

private MoodleConverter moodleConverter;

@Override

public List<MoodleCourseSectionMO> getAllCourses() {

return moodleCourseSectionDao.getAllCourses();

}

@Override

public List<MoodleCourseSectionMO> getArchivedCourseFilesFromMoodle(Long courseId) {

return moodleCourseSectionDao.getAllCourseSectionsByCourseId(courseId);

}

@Override

public MoodleCourseStatisticsMO getCourseStatistics(Long courseId) {

return moodleCourseStatisticsDao.getCourseStatisticsByCourseId(courseId);

}

@Override

public void getArchivedCourseFilesFromMoodle(Long courseId, OutputStream outputStream) {

List<MoodleCourseSectionMO> sections = moodleCourseSectionDao.getAllCourseSectionsByCourseId(courseId);

archiveCourseSectionFiles(sections, outputStream);

}

@Override

public void testArchiving(Long courseId, OutputStream outputStream) {

MoodleCourseSectionDTO[] courseSections = moodleRestClient.getCourseContents(courseId);

MoodleCourseSectionMO sectionMO = moodleConverter.convertDtoToMo(courseSections[1], courseId, "testDisplayName");

archiveCourseSectionFiles(Collections.singletonList(sectionMO), outputStream);

}

@Transactional("diplomnaTransactionManager")

@Override

public void updateDB() {

moodleContentDao.deleteAllContents();

moodleModuleDao.deleteAllModules();

moodleCourseSectionDao.deleteAllSections();

moodleCourseStatisticsDao.deleteAllStatistics();

MoodleCourseInfoDTO[] courseInfos = moodleRestClient.getCourses();

for (MoodleCourseInfoDTO courseInfo : courseInfos) {

MoodleCourseSectionDTO[] courseSections = moodleRestClient.getCourseContents(courseInfo.getId());

for (MoodleCourseSectionDTO section : courseSections) {

MoodleCourseSectionMO sectionMO = moodleConverter.convertDtoToMo(section, courseInfo.getId(), courseInfo.getDisplayname());

moodleCourseSectionDao.saveCourseSection(sectionMO);

}

MoodleCourseStatisticsMO statisticsMO = generateStatistics(courseInfo.getId(), courseInfo.getDisplayname(), courseSections);

moodleCourseStatisticsDao.saveCourseStatistics(statisticsMO);

}

}

private void archiveCourseSectionFiles(List<MoodleCourseSectionMO> sections, OutputStream outputStream) {

List<File> fileList = new ArrayList<>();

for (MoodleCourseSectionMO section : sections) {

downloadModules(section, fileList);

}

try {

MoodleFileArchiver.compressFilesToZip(fileList, outputStream);

} finally {

cleanTempFiles(fileList);

}

}

private void downloadModules(MoodleCourseSectionMO section, List<File> fileList) {

section.getModules().forEach(module -> {

// String directoryName = getDirectoryName(module, directoryList);

downloadFiles(module, fileList);

});

}

// private String getDirectoryName(MoodleModuleMO module, List<File> directoryList) {

// File directory = new File("temp\\module\_" + module.getId());

// if (directory.mkdir()) {

// directoryList.add(directory); // add to list of directories to be deleted

// return directory.getName() + "\\";

// } else {

// logger.warn("Failed to make directory " + directory.getAbsolutePath());

// return ""; // failed to create directory

// }

// }

private void downloadFiles(MoodleModuleMO module, List<File> fileList) {

module.getContents().forEach(moduleContent -> {

if (moduleContent.getType().equalsIgnoreCase(FILE)) {

File file = moodleRestClient.getFile(moduleContent.getFileurl(), moduleContent.getFileName());

if (file != null) {

fileList.add(file);

}

}

});

}

private void cleanTempFiles(List<File> fileList) {

try {

fileList.forEach(file -> {

if (!file.delete()) {

logger.warn("failed to delete file: " + file.getAbsolutePath());

file.deleteOnExit(); // attempt to have Java delete it automatically once JVM is stopped

}

});

} catch (Exception e) {

logger.error("failed to delete downloaded files", e);

}

// try {

// directoryList.forEach(directory -> {

// if (!directory.delete()) {

// logger.warn("failed to delete file: " + directory.getAbsolutePath());

// directory.deleteOnExit(); // attempt to have Java delete it automatically once JVM is stopped

// }

// });

// } catch (Exception e) {

// logger.error("failed to delete download directories", e);

// }

}

public MoodleCourseStatisticsMO generateStatistics(long courseId, String courseDisplayName, MoodleCourseSectionDTO[] courseSections) {

MoodleCourseStatisticsMO mo = new MoodleCourseStatisticsMO();

mo.setCourseId(courseId);

mo.setCourseName(courseDisplayName);

AtomicInteger fileCount = new AtomicInteger();

AtomicInteger urlCount = new AtomicInteger();

AtomicInteger discussionsCount = new AtomicInteger();

Arrays.stream(courseSections).forEach(section ->

Arrays.stream(section.getModules()).forEach(module ->

Arrays.stream(module.getContents()).forEach(content -> {

if ("url".equalsIgnoreCase(content.getType())) {

urlCount.getAndIncrement();

} else if ("file".equalsIgnoreCase(content.getType())) {

fileCount.getAndIncrement();

}

}

)

)

);

Arrays.stream(moodleRestClient.getForums(courseId)).forEach(forum ->

discussionsCount.addAndGet(forum.getNumdiscussions())

);

mo.setNumberOfFiles(fileCount.get());

mo.setNumberOfUrls(urlCount.get());

mo.setNumberOfResources(moodleRestClient.getResources(courseId).length);

mo.setNumberOfWikis(moodleRestClient.getWikis(courseId).length);

mo.setNumberOfForumDiscussions(discussionsCount.get());

return mo;

}

}

package all.component.diplomna.api;

import all.component.diplomna.model.MoodleCourseSectionMO;

import all.component.diplomna.model.MoodleCourseStatisticsMO;

import java.io.IOException;

import java.io.OutputStream;

import java.util.List;

public interface MoodleApi {

List<MoodleCourseSectionMO> getAllCourses();

List<MoodleCourseSectionMO> getArchivedCourseFilesFromMoodle(Long courseId);

MoodleCourseStatisticsMO getCourseStatistics(Long courseId);

void getArchivedCourseFilesFromMoodle(Long courseId, OutputStream outputStream) throws IOException;

void testArchiving(Long courseId, OutputStream outputStream) throws Exception;

void updateDB();

}

**2. Визуален интерфейс**

{

"compileOnSave": false,

"compilerOptions": {

"baseUrl": "./",

"outDir": "./dist/out-tsc",

"sourceMap": true,

"declaration": false,

"downlevelIteration": true,

"experimentalDecorators": true,

"module": "esnext",

"moduleResolution": "node",

"importHelpers": true,

"target": "es2015",

"typeRoots": [

"node\_modules/@types"

],

"lib": [

"es2018",

"dom"

]

},

"angularCompilerOptions": {

"enableIvy": true,

"fullTemplateTypeCheck": true,

"strictInjectionParameters": true

}

}

// protractor config file

// https://github.com/angular/protractor/blob/master/lib/config.ts

const { SpecReporter } = require('jasmine-spec-reporter');

exports.config = {

allScriptsTimeout: 11000,

specs: [

'./e2e/\*\*/\*.e2e-spec.ts'

],

capabilities: {

'browserName': 'chrome'

},

directConnect: true,

baseUrl: 'http://localhost:4200/',

framework: 'jasmine',

jasmineNodeOpts: {

showColors: true,

defaultTimeoutInterval: 30000,

print: function() {}

},

onPrepare() {

require('ts-node').register({

project: 'e2e/tsconfig.e2e.json'

});

jasmine.getEnv().addReporter(new SpecReporter({spec: { displayStackTrace: true } }));

}

};

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>BG.Diplomna.UI</groupId>

<artifactId>Diplomna.UI</artifactId>

<packaging>war</packaging>

<version>1.0-SNAPSHOT</version>

<!-- build-->

<build>

<finalName>diplomna-UI-webapp</finalName>

<plugins>

<!-- client-->

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-site-plugin</artifactId>

<version>3.7.1</version>

</plugin>

<plugin>

<groupId>com.github.eirslett</groupId>

<artifactId>frontend-maven-plugin</artifactId>

<version>1.8.0</version>

<executions>

<execution>

<id>install node and npm</id>

<goals>

<goal>

install-node-and-npm

</goal>

</goals>

<phase>prepare-package</phase>

<configuration>

<nodeVersion>v10.16.1</nodeVersion>

<npmVersion>6.11.3</npmVersion>

<nodeDownloadRoot>http://nodejs.org/dist/</nodeDownloadRoot>

<npmDownloadRoot>http://registry.npmjs.org/npm/-/</npmDownloadRoot>

</configuration>

</execution>

<execution>

<id>npm install</id>

<goals>

<goal>npm</goal>

</goals>

<phase>prepare-package</phase>

<configuration>

<arguments>install</arguments>

</configuration>

</execution>

<execution>

<id>npm run build</id>

<goals>

<goal>npm</goal>

</goals>

<phase>prepare-package</phase>

<configuration>

<arguments>run ng build</arguments>

</configuration>

</execution>

</executions>

</plugin>

<!-- war-->

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-war-plugin</artifactId>

<version>3.3.1</version>

<configuration>

<archiveClasses>true</archiveClasses>

<webXml>config/web.xml</webXml>

<warSourceDirectory>dist</warSourceDirectory>

<!-- <webResources>-->

<!-- <resource>C:\Users\Boyan\Desktop\Diplomna\_UI\dist</resource>-->

<!-- </webResources>-->

</configuration>

</plugin>

<!-- resource output dir-->

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-resources-plugin</artifactId>

<version>3.2.0</version>

<executions>

<execution>

<id>replace-configurations</id>

<phase>initialize</phase>

<goals>

<goal>copy-resources</goal>

</goals>

<configuration>

<outputDirectory>target\diplomna-UI-webapp\WEB-INF</outputDirectory>

<resources>

<resource>

<filtering>false</filtering>

<directory>C:\Users\Boyan\Desktop\Diplomna\_UI\config</directory>

</resource>

</resources>

</configuration>

</execution>

</executions>

</plugin>

</plugins>

</build>

</project>

{

"name": "client",

"version": "1.0.0",

"license": "MIT",

"scripts": {

"ng": "ng",

"start": "ng serve",

"build": "ng build",

"test": "ng test",

"lint": "ng lint",

"e2e": "ng e2e"

},

"private": true,

"dependencies": {

"@angular/animations": "^9.0.5",

"@angular/cdk": "^9.1.1",

"@angular/common": "^9.0.5",

"@angular/compiler": "^9.0.5",

"@angular/core": "^9.0.5",

"@angular/forms": "^9.0.5",

"@angular/material": "^9.1.1",

"@angular/platform-browser": "^9.0.5",

"@angular/platform-browser-dynamic": "^9.0.5",

"@angular/router": "^9.0.5",

"ag-grid": "^18.1.0",

"ag-grid-angular": "^18.1.0",

"ag-grid-community": "^24.1.0",

"body-parser": "^1.19.0",

"chart.js": "^2.9.3",

"chartjs-plugin-annotation": "^0.5.7",

"core-js": "^3.6.4",

"d3": "^7.3.0",

"express": "^4.17.1",

"file-saver": "^2.0.2",

"moment": "^2.26.0",

"ng-starrating": "^1.0.20",

"ng2-charts": "^2.3.1",

"ngx-cookie-service": "^2.4.0",

"ngx-infinite-scroll": "^8.0.1",

"ngx-select-dropdown": "^1.4.0",

"request": "^2.88.2",

"rxjs": "^6.5.4",

"tslib": "^1.10.0",

"zone.js": "^0.10.2"

},

"devDependencies": {

"@angular-devkit/build-angular": "~0.900.5",

"@angular/cli": "~9.0.5",

"@angular/compiler-cli": "^9.0.5",

"@angular/language-service": "^9.0.5",

"@types/jasmine": "~3.5.8",

"@types/jasminewd2": "~2.0.8",

"@types/node": "^12.11.1",

"codelyzer": "^5.1.2",

"jasmine-core": "~3.5.0",

"jasmine-spec-reporter": "~4.2.1",

"karma": "~4.4.1",

"karma-chrome-launcher": "~3.1.0",

"karma-coverage-istanbul-reporter": "~2.1.1",

"karma-jasmine": "~3.1.1",

"karma-jasmine-html-reporter": "^1.5.2",

"protractor": "~5.4.3",

"ts-node": "~8.6.2",

"tslint": "~6.0.0",

"typescript": "~3.7.5"

}

}

// Karma configuration file

// https://karma-runner.github.io/6.3/config/configuration-file.html

module.exports = function(config) {

config.set({

basepath: '',

frameworks: ['jasmine', '@angular-devkit/build-angular'],

plugins: [

require('karma-jasmine'),

require('karma-chrome-launcher'),

require('karma-jasmine-html-reporter'),

require('karma-coverage-istanbul-reporter'),

require('@angular-devkit/build/angular/plugins/karma'),

],

client: {

clearContext: false // Jasmine Spec Runner output visible in browser

},

coverageIstanbulReporter: {

dir: require('path').join(\_\_dirname, 'coverage'), reports: ['html', 'lcovonly'],

fixWebpackSourcePaths: true

},

reporters: ['progress', 'kjhtml'],

port: 9876,

colors: true,

logLevel: config.LOG\_INFO,

autoWatch: true,

browsers: ['Chrome'],

singleRun: false

});

};

{

"$schema": "./node\_modules/@angular/cli/lib/config/schema.json",

"version": 1,

"newProjectRoot": "projects",

"projects": {

"client": {

"root": "",

"sourceRoot": "src",

"projectType": "application",

"prefix": "app",

"schematics": {

"@schematics/angular:application": {

"strict": true

}

},

"architect": {

"build": {

"builder": "@angular-devkit/build-angular:browser",

"options": {

"outputPath": "dist",

"index": "src/index.html",

"main": "src/main.ts",

"polyfills": "src/polyfills.ts",

"tsConfig": "src/tsconfig.app.json",

"assets": [

"src/favicon.ico",

"src/assets"

],

"styles": [

"src/styles.css"

],

"scripts": []

},

"configurations": {

"production": {

"fileReplacements": [

{

"replace": "src/environments/environment.ts",

"with": "src/environments/environment.prod.ts"

}

],

"optimization": true,

"outputHashing": "all",

"sourceMap": false,

"extractCss": true,

"namedChunks": false,

"aot": true,

"extractLicenses": true,

"vendorChunk": false,

"buildOptimizer": true

}

}

},

"serve": {

"builder": "@angular-devkit/build-angular:dev-server",

"options": {

"browserTarget": "client:build"

},

"configurations": {

"production": {

"browserTarget": "client:build:production"

}

}

},

"extract-i18n": {

"builder": "@angular-devkit/build-angular:extract-i18n",

"options": {

"browserTarget": "client:build"

}

},

"test": {

"builder": "@angular-devkit/build-angular:karma",

"options": {

"main": "src/test.ts",

"polyfills": "src/polyfills.ts",

"tsConfig": "src/tsconfig.spec.json",

"karmaConfig": "karma.conf.js",

"styles": [

"styles.css"

],

"scripts": [],

"assets": [

"src/favicon.ico",

"src/assets"

]

}

},

"lint": {

"builder": "@angular-devkit/build-angular:tslint",

"options": {

"tsConfig": [

"src/tsconfig.app.json",

"src/tsconfig.spec.json"

],

"exclude": [

"\*\*/node\_modules/\*\*"

]

}

}

}

}

},

"defaultProject": "client"

}

<?xml version="1.0" encoding="ISO-8859-1" ?>

<web-app version="3.0" xmlns="http://java.sun.com/xml/ns/javaee" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app\_3\_0.xsd">

<welcome-file-list>

<welcome-file>index.html</welcome-file>

</welcome-file-list>

</web-app>

{

"extends": "../tsconfig.json",

"compilerOptions": {

"outDir": "../out-tsc/e2e",

"baseUrl": "./",

"module": "commonjs",

"target": "es5",

"types": [

"jasmine",

"jasminewd2",

"node"

]

}

}

import { browser, by, element } from 'protractor';

export class AppPage {

navigateTo() {

return browser.get('/');

}

getParagraphText() {

return element(by.css('app-root h1')).getText();

}

}

import { AppPage } from './app.po';

describe('client App', () => {

let page: AppPage;

beforeEach(() => {

page = new AppPage();

});

it('should display welcome message', () => {

page.navigateTo();

expect(page.getParagraphText()).toEqual('Welcome to app!');

});

});

{

"extends": "../tsconfig.json",

"compilerOptions": {

"outDir": "../out-tsc/spec",

"baseUrl": "./",

"types": [

"jasmine",

"node"

]

},

"files": [

"test.ts",

"polyfills.ts"

],

"include": [

"\*\*/\*.spec.ts",

"\*\*/\*.d.ts"

]

}

{

"extends": "../tsconfig.json",

"compilerOptions": {

"outDir": "..out-tsc/app",

"baseUrl": "./.",

"types": []

},

"files": [

"main.ts",

"polyfills.ts"

],

"include": [

"src/\*\*/\*.d.ts"

]

}

// required by karma.conf.js

import 'zone.js/dist/zone-testing';

import { getTestBed } from '@angular/core/testing';

import {

BrowserDynamicTestingModule,

platformBrowserDynamicTesting

} from '@angular/platform-browser/dynamic/testing';

declare const require: any;

getTestBed().initTestEnvironment(

BrowserDynamicTestingModule,

platformBrowserDynamicTesting()

);

const context = require.context('./', true, /\.spec\.ts$/);

context.keys().map(context);

@import "~@angular/material/prebuilt-themes/indigo-pink.css";

/\* Top Bar \*/

app-top-bar {

width: auto;

height: 68px;

background-color: #19d2d2;

padding: 16px;

padding-left: 25px;

display: flex;

flex-direction: row;

justify-content: space-between;

align-items: center;

}

app-top-bar h1 {

color: mediumblue;

margin: 0;

}

app-top-bar h2 {

margin: 0;

}

/\* Fancy Button \*/

.fancy-button {

/\*background-color: white;\*/

color: #191ad2;

}

.fancy-button i.material-icons {

color: #191ad2;

padding-right: 5px;

}

/\* generic \*/

.center {

text-align: center;

position: relative;

top: 50%;

}

/\*\*

\* This file includes polyfills needed by Angular and is loaded before the app.

\* You can add your own extra polyfills to this file.

\*

\* This file is divided into 2 sections:

\* 1. Browser polyfills. These are applied before loading ZoneJS and are sorted by browsers.

\* 2. Application imports. Files imported after ZoneJS that should be loaded before your main

\* file.

\*

\* The current setup is for so-called "evergreen" browsers; the last versions of browsers that

\* automatically update themselves. This includes Safari >= 10, Chrome >= 55 (including Opera),

\* Edge >= 13 on the desktop, and iOS 10 and Chrome on mobile.

\*

\* Learn more in https://angular.io/docs/ts/latest/guide/browser-support.html

\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* BROWSER POLYFILLS

\*/

/\*\* IE9, IE10 and IE11 requires all of the following polyfills. \*\*/

// import 'core-js/es6/symbol';

// import 'core-js/es6/object';

// import 'core-js/es6/function';

// import 'core-js/es6/parse-int';

// import 'core-js/es6/parse-float';

// import 'core-js/es6/number';

// import 'core-js/es6/math';

// import 'core-js/es6/string';

// import 'core-js/es6/date';

// import 'core-js/es6/array';

// import 'core-js/es6/regexp';

// import 'core-js/es6/map';

// import 'core-js/es6/set';

/\*\* IE10 and IE11 requires the following for NgClass support on SVG elements \*/

// import 'classlist.js'; // Run `npm install --save classlist.js`.

/\*\* IE10 and IE11 requires the following to support `@angular/animation`. \*/

// import 'web-animations-js'; // Run `npm install --save web-animations-js`.

/\*\* Evergreen browsers require these. \*\*/

// import 'core-js/es6/reflect';

// import 'core-js/es7/reflect';

/\*\*

\* Web Animations `@angular/platform-browser/animations`

\* Only required if AnimationBuilder is used within the application and using IE/Edge or Safari.

\* Standard animation support in Angular DOES NOT require any polyfills (as of Angular 6.0).

\*/

// import 'web-animations-js'; // Run `npm install --save web-animations-js`.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Zone JS is required by Angular itself.

\*/

import 'zone.js/dist/zone'; // Included with Angular CLI.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* APPLICATION IMPORTS

\*/

/\*\*

\* Date, currency, decimal and percent pipes.

\* Needed for: All but Chrome, Firefox, Edge, IE11 and Safari 10

\*/

// import 'intl'; // Run `npm install --save intl`.

import './polyfills';

import { enableProdMode } from '@angular/core';

import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';

import { AppModule } from './app/app.module';

platformBrowserDynamic().bootstrapModule(AppModule).then(ref => {

// Ensure Angular destroys itself on hot reloads.

if (window['ngRef']) {

window['ngRef'].destroy();

}

window['ngRef'] = ref;

// Otherwise, log the boot error

}).catch(err => console.error(err));

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Client</title>

<base href="./"/>

<meta name="viewport" content="width=device-width, initial-scale=1">

<link rel="icon" type="image/x-icon" href="favicon.ico">

<link href="https://fonts.googleapis.com/icon?family=Material+Icons" rel="stylesheet"/>

</head>

<body>

<app-root></app-root>

</body>

</html>

import {NgModule} from '@angular/core';

import {Routes, RouterModule} from '@angular/router';

import {CommonModule} from "@angular/common";

import {MoodleCourseTableComponent} from './course-page/table/course-table.component';

import {StatisticPageComponent} from './statistic-page/statistic-page.component';

import {LoginPageComponent} from './login-page/login-page.component';

import {AboutPageComponent} from './about-page/about-page.component';

const routes: Routes = [

{path: 'courses', component: MoodleCourseTableComponent},

{path: 'statistics', component: StatisticPageComponent},

{path: 'about', component: AboutPageComponent},

{path: '', component: LoginPageComponent},

];

@NgModule({

imports: [CommonModule,

RouterModule.forRoot(routes, { enableTracing: false, useHash: false}) // useHash true allows loading the page through browser url

],

exports: [RouterModule]

})

export class AppRoutingModule {}

import {NgModule} from '@angular/core';

import {BrowserModule} from '@angular/platform-browser';

import {FormsModule} from '@angular/forms';

import {HttpClientModule} from '@angular/common/http';

import {AgGridModule} from 'ag-grid-angular';

import {AppRoutingModule} from './app-routing.module';

import {RouterModule} from '@angular/router';

import {BrowserAnimationsModule} from '@angular/platform-browser/animations';

import {MatTableModule} from '@angular/material/table'

import {MatSelectModule} from '@angular/material/select';

import {MatListModule} from '@angular/material/list';

import {MatCardModule} from '@angular/material/card';

import {MatTooltipModule} from '@angular/material/tooltip';

import {MatButtonModule} from '@angular/material/button';

import {MatInputModule} from '@angular/material/input';

import {TopBarComponent} from './top-bar/top-bar.component';

import {MoodleCourseTableComponent} from './course-page/table/course-table.component';

import {StatisticPageComponent} from './statistic-page/statistic-page.component';

import {LoginPageComponent} from './login-page/login-page.component';

import {AppComponent} from './app.component';

@NgModule({

imports:

[ BrowserModule,

FormsModule,

AppRoutingModule,

HttpClientModule,

BrowserAnimationsModule,

MatTableModule,

MatSelectModule,

MatListModule,

MatCardModule,

MatTooltipModule,

MatButtonModule,

MatInputModule,

],

declarations:

[ AppComponent,

TopBarComponent,

MoodleCourseTableComponent,

StatisticPageComponent,

LoginPageComponent

],

bootstrap: [ AppComponent ]

})

export class AppModule { }

import { Component, VERSION } from '@angular/core';

@Component({

selector: 'app-root',

templateUrl: './app.component.html',

styleUrls: [ './app.component.css' ]

})

export class AppComponent {

name = 'Angular ' + VERSION.major;

title = 'app works..!';

}

import { TestBed, async } from '@angular/core/testing';

import { RouterTestingModule } from '@angular/router/testing';

import { AppComponent } from './app.component';

describe('AppComponent', () => {

beforeEach(async(() => {

TestBed.configureTestingModule({

imports: [

RouterTestingModule

],

declarations: [

AppComponent

],

}).compileComponents();

}));

it('should create the app', async(() => {

const fixture = TestBed.createComponent(AppComponent);

const app = fixture.debugElement.componentInstance;

expect(app).toBeTruthy();

}));

it('should have as title \'app\'', async(() => {

const fixture = TestBed.createComponent(AppComponent);

const app = fixture.debugElement.componentInstance;

expect(app.title).toEqual('app');

}));

it('should render title in a h1 tag', async(() => {

const fixture = TestBed.createComponent(AppComponent);

fixture.detectChanges();

const compiled = fixture.debugElement.nativeElement;

expect(compiled.querySelector('h1').textContext).toContain('Welcome to app!');

}))

});

<h1>

<app-top-bar></app-top-bar>

</h1>

<router-outlet></router-outlet>

import { Component } from '@angular/core';

@Component({

selector: 'app-top-bar',

templateUrl: './top-bar.component.html',

styleUrls: ['./top-bar.component.css']

})

export class TopBarComponent {

isShowNavigation() {

// default page is login page and should allow navigation only through login

return document.location.href != document.baseURI;

}

}

<!--<a [routerLink]="['/']">-->

<!-- <h1>Home</h1>-->

<!--</a>-->

<div \*ngIf=isShowNavigation()>

<a style="float: inline-start; padding-right: 20px;" [routerLink]="['/courses']">

<h2>Courses</h2>

</a>

<a style="float: inline-start; padding-right: 20px;" [routerLink]="['/statistics']">

<h2>Statistics</h2>

</a>

<a style="position: absolute; right: 0; padding-right: 20px;" [routerLink]="['/about']">

<h2>About</h2>

</a>

</div>

import {Component, OnInit, AfterViewInit, ViewChild} from '@angular/core';

import {ActivatedRoute} from '@angular/router';

import {MatSort} from '@angular/material/sort';

import {MatPaginator} from '@angular/material/paginator';

import {MatOption} from '@angular/material/core';

import {MatSelect} from '@angular/material/select';

import {MatProgressSpinnerModule} from '@angular/material/progress-spinner';

import {MatOptionSelectionChange} from '@angular/material/core';

import {MatButtonModule} from '@angular/material/button';

import {CoursePageService, MoodleCourseSection, MoodleCourseStatistic} from '../course-page/service/course-service.service';

import \* as d3 from 'd3-selection';

import \* as d3Scale from 'd3-scale';

import \* as d3Array from 'd3-array';

import \* as d3Axis from 'd3-axis';

@Component({

selector: 'app-statistic-page',

templateUrl: './statistic-page.component.html',

styleUrls: ['./statistic-page.component.css'],

providers: [CoursePageService]

})

export class StatisticPageComponent implements OnInit {

public isLoginFailed = false;

private statisticBarFormat: MoodleCourseStatisticBar[];

public moodleAllCoursesData: MoodleCourseSection[];

@ViewChild(MatPaginator, { static: true }) paginator: MatPaginator;

@ViewChild(MatSort, { static: true }) sort: MatSort;

public currentRate = 8;

public title = 'D3 Barchart with Angular 10';

public width: number;

public height: number;

public margin = { top: 20, right: 20, bottom: 30, left: 40 };

public x: any;

public y: any;

public svg: any;

public g: any;

constructor(private coursePageService: CoursePageService, private route: ActivatedRoute) {

console.log("constructor of StatisticPageComponent");

this.width = 900 - this.margin.left - this.margin.right;

this.height = 500 - this.margin.top - this.margin.bottom;

}

ngOnInit() {

this.getAllCourses().subscribe(data => {

this.moodleAllCoursesData = data;

});

}

ngAfterViewInit(): void {

this.paginator = this.paginator;

this.sort = this.sort;

}

getAllCourses() {

return this.coursePageService.getAllCourses();

}

updateCourses() {

this.coursePageService.updateCourses().subscribe(() => {

this.getAllCourses().subscribe(data => {

this.moodleAllCoursesData = data;

});

});

}

changeCourse(event: MatOptionSelectionChange, selectedCourseId:string, selectedCourseName:string) {

this.clearSvg();

this.coursePageService.getCourseStatistics(selectedCourseId).subscribe(data => {

this.statisticBarFormat = [];

this.transformStatisticsIntoBarFormat(data);

this.initSvg();

this.initAxis();

this.drawAxis();

this.drawBars();

});

}

clearSvg() {

d3.selectAll("svg").selectAll("\*").remove();

}

transformStatisticsIntoBarFormat(data: MoodleCourseStatistic) {

for (const [key, value] of Object.entries(data)) {

if (key.startsWith("numberOf")) {

var statisticBarObj = new MoodleCourseStatisticBar(key.substring("numberOf".length), value);

this.statisticBarFormat.push(statisticBarObj);

}

}

}

initSvg() {

if (!this.svg) {

this.svg = d3.select('#barChart')

.append('svg')

.attr('height', '100%')

.attr('viewBox', '0 0 900 500')

}

this.g = this.svg.append('g')

.attr('transform', 'translate(' + this.margin.left + ',' + this.margin.top + ')');

}

initAxis() {

this.x = d3Scale.scaleBand().rangeRound([0, this.width]).padding(0.1);

this.y = d3Scale.scaleLinear().rangeRound([this.height, 0]);

this.x.domain(this.statisticBarFormat.map((data) => data.axisX));

this.y.domain([0, d3Array.max(this.statisticBarFormat, (data) => data.axisY)]);

}

drawAxis() {

this.g.append('g')

.attr('class', 'axis axis--x')

.attr('transform', 'translate(0,' + this.height + ')')

.call(d3Axis.axisBottom(this.x));

this.g.append('g')

.attr('class', 'axis axis--y')

.call(d3Axis.axisLeft(this.y))

.append('text')

.attr('class', 'axis-title')

.attr('transform', 'rotate(-90)')

.attr('y', 6)

.attr('dy', '0.71em')

.attr('text-anchor', 'end');

}

drawBars() {

var bar = this.g.selectAll('.bar')

.data(this.statisticBarFormat)

.enter()

.append('g')

.attr("class","bar")

.attr('x', (data) => this.x(data.axisX))

.attr('y', (data) => this.y(data.axisY))

.attr('width', this.x.bandwidth())

.attr('height', (data) => this.height - this.y(data.axisY));

bar.append('rect')

.attr('class', 'barRect')

.attr('x', (data) => this.x(data.axisX))

.attr('y', (data) => this.y(data.axisY))

.attr('width', this.x.bandwidth())

.attr('fill', '#498bfc')

.attr('height', (data) => this.height - this.y(data.axisY));

// text for the bar

this.g.selectAll('.bar')

.append("text")

.attr("x", (data) => this.x(data.axisX) + (this.x.bandwidth() / 2) )

.attr("y", (data) => this.y(data.axisY) - 5)

//.attr("dy", ".35em")

.attr('fill', 'black')

.style("font-size", "20px")

.style("text-anchor", "middle")

.text(data => data.axisY);

}

}

export class MoodleCourseStatisticBar {

axisX: any;

axisY: any;

constructor(axisX, axisY) {

this.axisX = axisX;

this.axisY = axisY;

}

}

<div>

<h1>Statistics</h1>

</div>

<mat-card>

<div>

<div style="float:left">

<mat-form-field style="width:400px; font-size:15px;">

<mat-label>Select a course</mat-label>

<mat-select multiple="false"

[(ngModel)]="course"

style="width:400px">

<mat-option \*ngFor="let course of moodleAllCoursesData"

[value]="course.courseName"

matTooltip="{{course.courseName}}"

(click)="changeCourse($event, course.courseId, course.courseName)"

style="width:400px">

{{course.courseName}}

</mat-option>

</mat-select>

</mat-form-field>

</div>

<div style="float:inline-start; padding-left: 10px">

<button mat-raised-button color="warn" (click)="updateCourses()">Refresh entire database</button>

</div>

</div>

<br style="clear:both" /> <!--- clears the float -->

<div id="barChart" style="height:60vh; width:80vw"></div>

</mat-card>

@import "~@angular/material/prebuilt-themes/indigo-pink.css";

import {Component, ElementRef, OnInit, AfterViewInit, ViewChild} from '@angular/core';

import {ActivatedRoute, RouterModule, Router} from '@angular/router';

import {MatInputModule} from '@angular/material/input';

import {LoginPageService} from './service/login-page.service';

@Component({

selector: 'app-login-page',

templateUrl: './login-page.component.html',

styleUrls: ['./login-page.component.css'],

providers: [LoginPageService]

})

export class LoginPageComponent {

public isLoginFailed = false;

public errorMessage = "";

@ViewChild('username') username: ElementRef;

@ViewChild('password') password: ElementRef;

constructor(private loginPageService: LoginPageService, private route: ActivatedRoute, private router:Router) {

console.log("constructor of LoginPageComponent");

}

login(username:string, password:string) {

this.isLoginFailed = false; // visibly refresh for user in case it has failed

this.loginPageService.login(this.username.nativeElement.value, this.password.nativeElement.value).subscribe(response => {

console.log(response);

if (response.error) {

this.isLoginFailed = true;

this.errorMessage = response.error;

} else if (response.token) {

this.isLoginFailed = false;

this.errorMessage = "";

this.router.navigate(['courses'], { relativeTo: this.route });

}

});

}

}

<mat-card>

<div>

<mat-form-field style="width:100px; font-size:15px;">

<mat-label>Username</mat-label>

<input matInput type="text" #username/>

</mat-form-field>

</div>

<div>

<mat-form-field style="width:100px; font-size:15px;">

<mat-label>Password</mat-label>

<input matInput type="password" #password/>

</mat-form-field>

</div>

<div>

<button mat-raised-button color="primary" (click)="login()">Login</button>

</div>

<div \*ngIf="isLoginFailed" style="padding-top: 10px; font-size: small; color: red;"><i><strong>{{errorMessage}}</strong></i></div>

</mat-card>

@import "~@angular/material/prebuilt-themes/indigo-pink.css";

import {Injectable} from '@angular/core';

import {HttpClient, HttpHeaders} from '@angular/common/http';

import {Observable, Subject} from 'rxjs';

@Injectable()

export class LoginPageService {

private baseConfig = { withCredentials: false, headers: { 'Content-Type': 'application/x-www-form-urlencoded'} };

base\_url: string = 'https://aptitude-learn.w3c.fmi.uni-sofia.bg';

constructor(private httpClient: HttpClient) {

console.log("service constructor");

console.log("this.base\_url " + this.base\_url);

}

login(username:string, password:string): Observable<any> {

let loginUrl: string = this.base\_url + "/login/token.php?service=moodle\_mobile\_app";

console.log("loginUrl to: " + loginUrl);

let payload = "username="+username+"&password="+password;

return this.httpClient.post<any>(loginUrl, payload, this.baseConfig);

}

}

import {TestBed, inject} from '@angular/core/testing';

import {LoginPageService} from './login-page.service';

describe('LoginPageService', () => {

beforeEach(() => {

TestBed.configureTestingModule({

providers: [LoginPageService]

)};

});

it('should be created', inject([LoginPageService], (service: LoginPageService) => {

expect(service).toBeTruthy();

}));

});

import {Component, OnInit, AfterViewInit, ViewChild} from '@angular/core';

import {ActivatedRoute} from '@angular/router';

import {MatTableDataSource} from '@angular/material/table';

import {MatSort} from '@angular/material/sort';

import {MatPaginator} from '@angular/material/paginator';

import {MatOption} from '@angular/material/core';

import {MatSelect} from '@angular/material/select';

import {MatProgressSpinnerModule} from '@angular/material/progress-spinner';

import {MatOptionSelectionChange} from '@angular/material/core';

import {MatButtonModule} from '@angular/material/button';

import {CoursePageService, MoodleCourseSection, MoodleModule, MoodleContent, MoodleVO} from '../service/course-service.service';

@Component({

selector: 'app-course-table',

templateUrl: './course-table.component.html',

styleUrls: ['./course-table.component.css'],

providers: [CoursePageService]

})

export class MoodleCourseTableComponent implements OnInit {

displayedColumns: string[] = ['sectionName', 'moduleName', 'fileName', 'author'];

public selectedCourseId;

public selectedCourseName;

public moodleAllCoursesData: MoodleCourseSection[];

public moodleVOsData: MoodleVO[];

public moodleCourseTableData = new MatTableDataSource<MoodleVO>();

@ViewChild(MatPaginator, { static: true }) paginator: MatPaginator;

@ViewChild(MatSort, { static: true }) sort: MatSort;

constructor(private coursePageService: CoursePageService, private route: ActivatedRoute) {

console.log("constructor of MoodleCourseTableComponent");

}

ngOnInit() {

this.getAllCourses().subscribe(data => {

this.moodleAllCoursesData = data;

});

}

ngAfterViewInit(): void {

this.moodleCourseTableData.paginator = this.paginator;

this.moodleCourseTableData.sort = this.sort;

}

getAllCourses() {

return this.coursePageService.getAllCourses();

}

getCourseData(courseId:string) {

return this.coursePageService.getCourseDataVOs(courseId);

}

changeCourse(event: MatOptionSelectionChange, selectedCourseId:number, selectedCourseName:string) {

this.selectedCourseId = selectedCourseId;

this.selectedCourseName = selectedCourseName;

this.getCourseData(this.selectedCourseId).subscribe(data => {

this.moodleCourseTableData.data = data;

});

}

downloadCourseFiles() {

this.coursePageService.downloadCourseFiles(this.selectedCourseId).subscribe(data => {

const downloadUrl = window.URL.createObjectURL(data);

var anchor = document.createElement("a"); // "a" for an anchor element

anchor.download = this.selectedCourseName + ".zip";

anchor.href = downloadUrl;

anchor.click();

window.URL.revokeObjectURL(downloadUrl);

});

}

updateCourses() {

this.coursePageService.updateCourses().subscribe(() => {

this.getAllCourses().subscribe(data => {

this.moodleAllCoursesData = data;

});

});

}

}

<div>

<h1>Courses</h1>

</div>

<mat-card>

<div>

<div style="float:left">

<mat-form-field style="width:400px; font-size:15px;">

<mat-label>Select a course</mat-label>

<mat-select multiple="false"

[(ngModel)]="course"

style="width:400px">

<mat-option \*ngFor="let course of moodleAllCoursesData"

[value]="course.courseName"

matTooltip="{{course.courseName}}"

(click)="changeCourse($event, course.courseId, course.courseName)"

style="width:400px">

{{course.courseName}}

</mat-option>

</mat-select>

</mat-form-field>

</div>

<div style="float:inline-start; padding-left: 10px">

<button mat-raised-button color="warn" (click)="updateCourses()">Refresh entire database</button>

</div>

</div>

<br style="clear:both" /> <!--- clears the float -->

<div style="padding-bottom: 20px">

<button \*ngIf="selectedCourseId" mat-raised-button (click)="downloadCourseFiles()">Download course files</button>

</div>

<table \*ngIf="selectedCourseId" mat-table [dataSource]="moodleCourseTableData" class="mat-elevation-z8">

<!--- Note that these columns can be defined in any order.

The actual rendered columns are set as a property on the row definition" -->

<!-- Section Column -->

<ng-container matColumnDef="sectionName">

<th mat-header-cell \*matHeaderCellDef> Section</th>

<td mat-cell \*matCellDef="let element"> {{element.sectionName}}</td>

</ng-container>

<!-- Module Column -->

<ng-container matColumnDef="moduleName">

<th mat-header-cell \*matHeaderCellDef> Module</th>

<td mat-cell \*matCellDef="let element"> {{element.moduleName}}</td>

</ng-container>

<!-- File Column -->

<ng-container matColumnDef="fileName">

<th mat-header-cell \*matHeaderCellDef> File</th>

<td mat-cell \*matCellDef="let element"> {{element.fileName}}</td>

</ng-container>

<!-- Author Column -->

<ng-container matColumnDef="author">

<th mat-header-cell \*matHeaderCellDef> Author</th>

<td mat-cell \*matCellDef="let element"> {{element.author}}</td>

</ng-container>

<tr mat-header-row \*matHeaderRowDef="displayedColumns"></tr>

<tr mat-row \*matRowDef="let row; columns: displayedColumns;"></tr>

</table>

</mat-card>

@import "~@angular/material/prebuilt-themes/indigo-pink.css";

table {

width: 100%;

}

import {Injectable} from '@angular/core';

import {HttpClient, HttpHeaders} from '@angular/common/http';

import {Observable, Subject} from 'rxjs';

import {MatTableDataSource} from '@angular/material/table';

@Injectable()

export class CoursePageService {

private baseConfig = { withCredentials: true, headers: { 'Content-Type': 'application/json;charset=utf-8;'} };

base\_url: string = window.location.protocol + "//" + window.location.hostname + ":8080" + "/diplomna-webapp/myService";

constructor(private httpClient: HttpClient) {

console.log("service constructor");

console.log("this.base\_url " + this.base\_url);

}

getAllCourses(): Observable<MoodleCourseSection[]> {

let getAllCoursesUrl: string = this.base\_url + "/getAllCourses";

console.log("getAllCourses to: " + getAllCoursesUrl);

return this.httpClient.get<MoodleCourseSection[]>(getAllCoursesUrl, this.baseConfig);

}

getCourseData(courseId:string): Observable<MoodleCourseSection[]> {

let getCourseDataUrl: string = this.base\_url + "/getCourseData/?courseId="+courseId;

console.log("getCourseData to: " + getCourseDataUrl);

return this.httpClient.get<MoodleCourseSection[]>(getCourseDataUrl, this.baseConfig);

}

getCourseStatistics(courseId:string): Observable<MoodleCourseStatistic> {

let getCourseStatisticsUrl: string = this.base\_url + "/getCourseStatistics/?courseId=" + courseId;

console.log("getStatisticsUrl: " + getCourseStatisticsUrl);

return this.httpClient.get<MoodleCourseStatistic>(getCourseStatisticsUrl, this.baseConfig);

}

updateCourses() {

let updateCoursesUrl: string = this.base\_url + "/updateDB";

return this.httpClient.patch(updateCoursesUrl, this.baseConfig);

}

getCourseDataVOs(courseId:string): Observable<MoodleVO[]> {

var subject = new Subject<MoodleVO[]>();

this.getCourseData(courseId).subscribe(moodleCourseSectionData => {

let convertedData = this.convertToVOs(moodleCourseSectionData);

subject.next(convertedData);

});

return subject.asObservable();

}

convertToVOs(moodleCourseSections: MoodleCourseSection[]): MoodleVO[] {

var moodleVOs: MoodleVO[] = [];

moodleCourseSections.forEach(moodleCourseSection => {

moodleCourseSection.modules.forEach(module => {

module.contents.forEach(content => {

let moodleVO: MoodleVO = {

sectionName: moodleCourseSection.sectionName,

moduleName: module.name,

fileName: content.fileName,

mimetype: content.mimetype,

author: content.author

};

moodleVOs.push(moodleVO);

});

});

});

return moodleVOs;

}

downloadCourseFiles(courseId:string) {

let downloadCourseFilesUrl: string = this.base\_url + "/downloadCourseFiles/?courseId=" + courseId;

console.log("downloadCourseFiles request sent to: " + downloadCourseFilesUrl);

let downloadHeaders = new HttpHeaders({'Content-Type': 'application/octet-stream', 'Accept':'application/octet-stream'});

let downloadConfig = { headers: downloadHeaders, responseType: 'blob' as const }

return this.httpClient.get(downloadCourseFilesUrl, downloadConfig);

}

}

export class MoodleCourseSection {

id: number;

modules: MoodleModule[];

courseId: number;

sectionId: number;

courseName: string;

sectionName: string;

userVisible: any;

constructor(id, modules, courseId, sectionId, courseName, sectionName, userVisible) {

this.id = id || 'N/A';

this.modules = modules;

this.courseId = courseId;

this.sectionId = sectionId;

this.courseName = courseName;

this.sectionName = sectionName;

this.userVisible = userVisible;

}

}

export class MoodleModule {

id: number;

courseSectionId: number;

contents: MoodleContent[];

name: string;

visible: any;

userVisible: any;

constructor(id, courseSection, contents, name, visible, userVisible) {

this.id = id || 'N/A';

this.courseSectionId = courseSection.id;

this.contents = contents;

this.name = name;

this.visible = visible;

this.userVisible = userVisible;

}

}

export class MoodleContent {

id: number;

moduleId: number;

type: string;

fileName: string;

fileurl: string;

mimetype: string;

author: string;

constructor(id, module, type, fileName, fileurl, userVisible, mimetype, author) {

this.id = id || 'N/A';

this.moduleId = module.id;

this.type = type;

this.fileName = fileName;

this.fileurl = fileurl;

this.mimetype = mimetype;

this.author = author;

}

}

export class MoodleVO {

sectionName: string;

moduleName: string;

fileName: string;

mimetype: string;

author: string;

constructor(sectionName, moduleName, fileName, mimetype, author) {

this.sectionName = sectionName || 'N/A';

this.moduleName = moduleName;

this.fileName = fileName;

this.mimetype = mimetype;

this.author = author;

}

}

export class MoodleCourseStatistic {

id: number;

courseId: number;

courseName: string;

numberOfFiles: number;

numberOfUrls: number;

numberOfResources: number;

numberOfWikis: number;

numberOfForumDiscussions: number;

constructor(id, courseId, courseName, numberOfFiles, numberOfUrls, numberOfResources, numberOfWikis, numberOfForumDiscussions) {

this.id = id || 'N/A';

this.courseId = courseId;

this.courseName = courseName;

this.numberOfFiles = numberOfFiles;

this.numberOfUrls = numberOfUrls;

this.numberOfResources = numberOfResources;

this.numberOfWikis = numberOfWikis;

this.numberOfForumDiscussions = numberOfForumDiscussions;

}

}

import { TestBed, inject } from '@angular/core/testing';

import { CoursePageService } from './course-service.service';

describe('CoursePageService', () => {

beforeEach(() => {

TestBed.configureTestingModule({

providers: [CoursePageService]

)};

});

it('should be created', inject([CoursePageService], (service: CoursePageService) => {

expect(service).toBeTruthy();

}));

});

import { Component } from '@angular/core';

@Component({

selector: 'app-about-page',

templateUrl: './about-page.component.html',

styleUrls: ['./about-page.component.css']

})

export class AboutPageComponent {

}

<h1>

<div class="center">

<p>Това приложение е изготвено като дипломна работа от</p>

<div>Боян Стефанов Гигов</div>

<p>ф.н. 121216228</p>

</div>

</h1>

<br>

<br>

<br>

<h2>

<div>

<p>Използвани технологии:</p>

<p>1.Сървър: Maven, Java, Spring, Jersey, Hibernate, gson, log4j, jackson, tomcat, xml, json</p>

<p>2.Визуален интерфейс: Angular, d3, nodejs, npm, typescript, html, css, json</p>

<p>3.База данни: Oracle db, ojdbc driver, dbcp2, liquibase</p>

</div>

</h2>