Санкт-Петербургский государственный университет

Кафедра информационно-аналитических систем

Группа 23.М08-мм

Разработка серверной части личного кабинета для веб-приложения Mundane Assignment Police

ФРОЛОВ Андрей Александрович

Отчёт по учебной практике
в форме «Производственное задание»

Научный руководитель:
ассистент кафедры информационно-аналитических систем, Г. А. Чернышев

Санкт-Петербург
2023

Оглавление

Введение

“Mundane Assignment Police” [5] — это веб-приложение для проверки курсовых и дипломных работ, отчетов по практикам и прочих студенческих текстов. Оно представляет собой валидатор pdf-документов, в котором для пользователя доступна, по сути, только одна важная опция — загрузка файла.

В процессе проверки сервис автоматически находит часто встречающиеся проблемы (опечатки): использование тире неправильного типа, отсутствие ссылки, одна подсекция внутри секции, выход текста на поля и прочее. В результате, формируется список найденных недочетов, при выборе определенного элемента которого, приложение отображает страницу документа с ошибкой и выделяет нужную строку посредством подчеркивания. Таким образом, основная функциональность “Mundane Assignment Police” сводится к единственному возможному сценарию использования, что делает его очень простым в глазах пользователя.

В тоже время хотелось бы, чтобы продукт стал более разнообразным и, как следствие, более привлекательным. В связи с этим, предлагается разработать личный кабинет, минимально включающий в себя ведение истории проверок и возможность формирования собственных наборов правил к ним.

1. Постановка задачи

Целью работы является разработка серверной части личного кабинета пользователя для веб-приложения “Mundane Assignment Police”. Для ее выполнения были поставлены следующие задачи:

1. проанализировать требования к серверной части личного кабинета;
2. спроектировать серверную часть;
3. реализовать серверную часть личного кабинета для валидатора студенческих работ;

2. Обзор

Приложение “Mundane Assignment Police” написано на языке Kotlin [3]. Он полностью соответствует требованиям проекта, поэтому является исходной точкой при выборе инструментов для реализации серверной части. Рассмотрим стек технологий, которые применялись для решения проектных задач.

2.1. PostgreSQL

Ввиду того, что первоначально взаимодействие приложения с базой данных не производилось, было необходимо подумать над тем, какую СУБД для хранения данных использовать. Рассматривались как SQL, так и NoSQL решения, но в конечном итоге выбор пал на “PostgreSQL” [6]. На это повлияли следующие причины:

1. Проверенная временем надежность и устойчивость — “PostgreSQL” обеспечивает целостность данных и защиту от сбоев, а также предоставляет инструменты для резервного копирования и восстановления данных.

2. Большой функционал — “PostgreSQL” имеет множество функций и возможностей, которые делают его одним из самых гибких и мощных СУБД. Она поддерживает хранимые процедуры, триггеры, внешние ключи и т.д.

3. Большое сообщество пользователей и разработчиков, которые занимаются развитием и улучшением данной СУБД.

2.2. Spring

Для упрощения создания приложения было решено использовать один из двух наиболее популярных фреймворков: “Spring” [7] (пришедший из Java-разработки) или “Ktor” [4].

В пользу “Spring” выдвигались следующие пункты:

- Зрелость платформы и экосистема. Любые часто встречающиеся проблемы уже решены за счет всевозможных стартеров и автоконфигураций.
- Fullstack составляющая, что важно ввиду необходимости наличия пользовательского интерфейса.
- Перспективы в дальнейшей поддержке и развитии.

Из недостатков можно отметить:

- Метрики приложения (высокий размер исходного файла, увеличенное время запуска и т.д.)

За использование “Ktor” говорили:

- Возможность подключить только те функции, которые необходимы для решения поставленной задачи.
- В противовес “Spring”, более эффективные показатели по метрикам приложения

Из недостатков:

- Отсутствие важных компонентов для промышленной разработки, таких как, например, внедрение зависимостей.
- Слабо развитая экосистема и неясные перспективы.

Таким образом, в сравнении участвовали диаметрально противоположные фреймворки, в выборе между которыми баланс преимуществ и недостатков оказался на стороне “Spring”.

2.3. Exposed

В качестве ORM для представление структуры базы данных внутри приложения использован фреймворк “Kotlin Exposed” [1]. Здесь также было проведено сравнение с более направленным на Java инструментом — “JPA” [2], однако в данном случае более громоздкие метрики фреймворк “JPA” компенсировать не может.

3. Реализация

Рассмотрим основные моменты реализации.

3.1. Анализ требований

Первоначальным этапом реализации был анализ требований. В ходе него было определено, что реализации подлежит следующая функциональность:

1. Регистрация и Авторизация
2. Восстановление пароля
3. Редактирование профиля
4. CRUD персональных пресетов (наборов правил)
5. CRUD результатов проверки документов. С возможностью скачивания ранее загруженного файла.
6. CRUD проектов (под проектом подразумевается объединение разных версий одной работы)

3.2. ER-диаграмма

На основе анализа требований была спроектирована ER-диаграмма предметной области (Рис. 1).

- User — центральная сущность, описывающая пользователя
- Role — сущность, определяющая роль пользователя
- Rule — сущность, обозначающая правило проверки
- Preset — сущность, определяющая набор правил
- Review — сущность, обозначающая факт проверки
- Project — сущность, описывающая проект

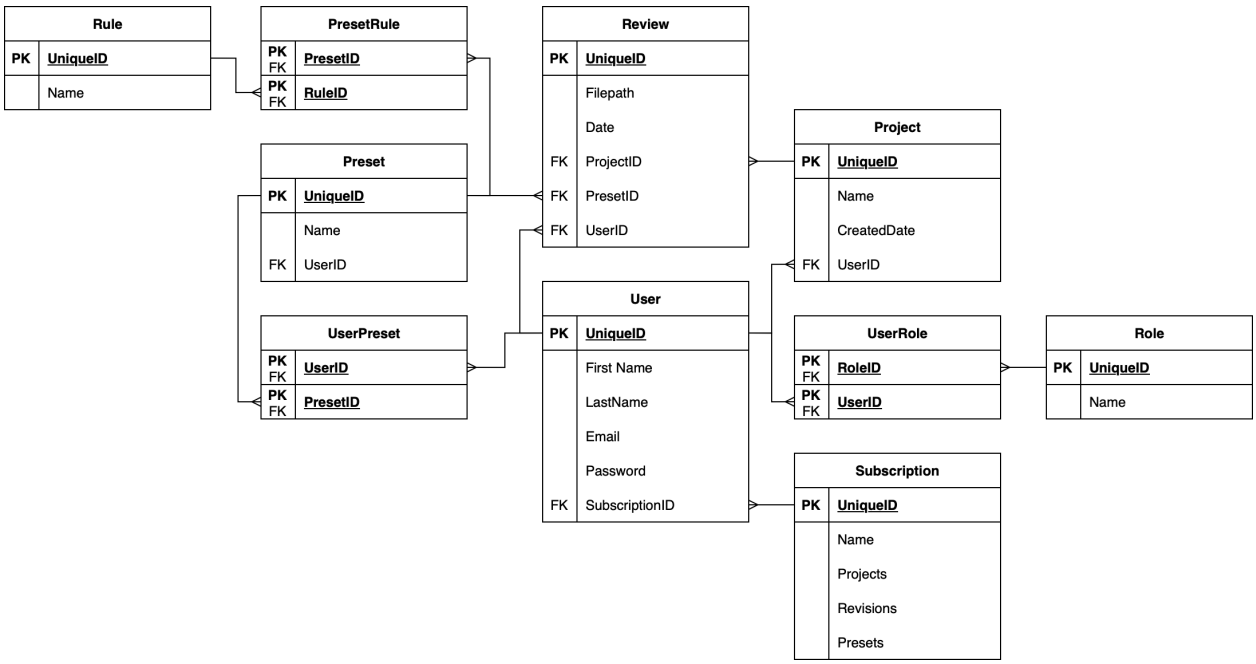

Рис. 1: ER-диаграмма предметной области

3.3. Архитектура приложения

Кроме предметной области была также разработана архитектура приложения (Рис. 2).

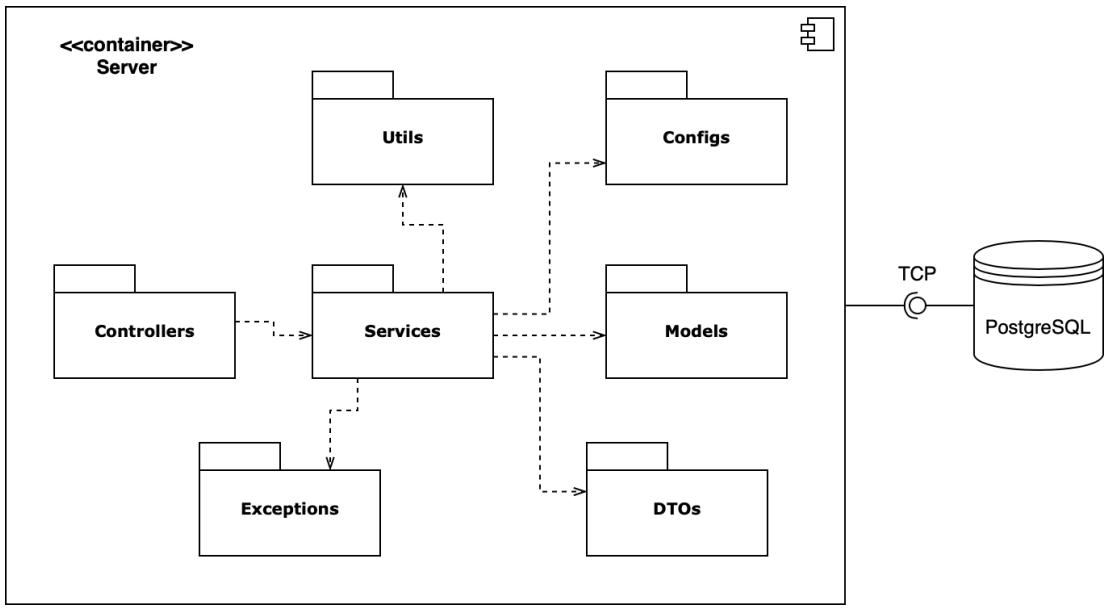

Рис. 2: Архитектура приложения

- Пакет Controllers содержит контроллеры для обработки запросов пользователя

- Пакет Models содержит классы, представляющие модель данных
- В пакете Services расположились сервисы, отвечающие за бизнес-логику приложения
- В пакете Configs расположились классы, отвечающие за конфигурацию приложения
- Пакет Exceptions содержит классы обработки ошибок
- В пакет Utils расположились классы утилит

3.4. Обработка запросов

Рассмотрим, как осуществляется обработка запросов к серверу. Концептуальная диаграмма изображена на рисунке 3.

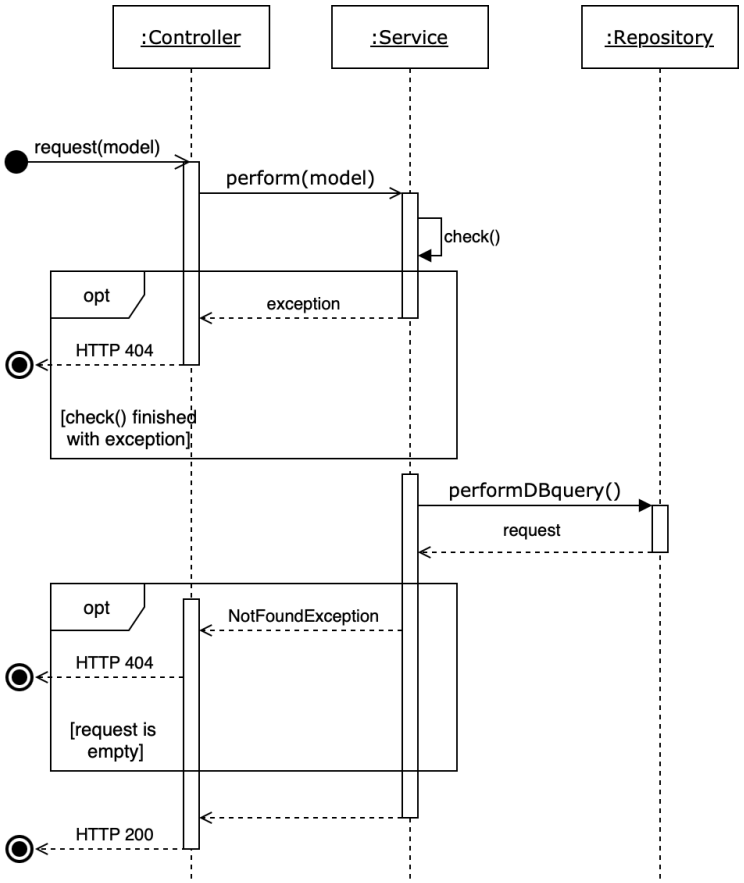

Рис. 3: Обработка запросов

- Контроллер принимает http-запрос с JSON обектом в виде модели.
- Сервис осуществляет проверку корректности запроса и инициирует запрос к базе данных.
- Если работа с базой завершилась успешно, возвращается 200 статускод.

Заключение

В ходе работы были достигнуты все поставленные задачи:

1. произведен анализ требований
2. спроетирован веб-сервер
3. реализован сервер для личного кабинета

Стоит подчеркнуть, что текущая версия проекта является начальной, в дальнейшем система будет расширятся и дорабатываться. Исходный код проекта расположен по ссылке: https://github.com/Andrew-develop/map/tree/new-backend.

Список литературы

[1] Exposed. — URL: https://github.com/JetBrains/Exposed (дата обращения: 20 декабря 2023 г.).

[2] JPA. — URL: https://spring.io/projects/spring-data-jpa/ (дата обращения: 20 декабря 2023 г.).

[3] Kotlin. — URL: https://kotlinlang.org (дата обращения: 20 декабря 2023 г.).

[4] Ktor. — URL: https://ktor.io (дата обращения: 20 декабря 2023 г.).

[5] Mundane Assignment Police. — URL: http://91.109.207.113/#/ (дата обращения: 27 октября 2023 г.).

[6] PostgreSQL. — URL: https://www.postgresql.org (дата обращения: 20 декабря 2023 г.).

[7] Spring. — URL: https://spring.io/projects/spring-boot/ (дата обращения: 20 декабря 2023 г.).