Санкт-Петербургский государственный университет

Математико-механический факультет

Тетин Илья Николаевич

Создание веб-приложения с использованием технологий Java Spring

Отчёт по учебной практике

Научный руководитель:
к. т. н., Брыксин Т. А.

Санкт-Петербург
2020

Оглавление

Введение **3**

Постановка задачи **4**

1. Обзор инструментов, используемых в приложении **5**

1.1. FreeMarker 5

1.2. Bootstrap 5

2. Реализация **6**

2.1. Архитектура 6

2.2. База данных 7

2.3. Активация аккаунта 7

2.4. Страница с сообщениями 8

2.4.1. Добавление новых сообщений 10

2.4.2. Реализвация лайков 11

2.5. Окно администратора 11

2.6. Профиль пользователя 12

Результаты **13**

Приложения **14**

Список литературы **15**

Введение

Фреймворк Spring[1] является одним из ключевых технологий, используемых при разработке веб-приложений на языке Java. По статистике JetBrains[2] для решения задачи написания сервера какого-либо приложения на языке Java воспользоваться именно этим инструментом предпочитает 61 % разработчиков.

Неудивительно, что на многих вакансиях на позицию Java-разработчика требуют знания данного фреймворка. Именно поэтому было принято решение создать собственное веб-2 с использованием технологий Spring.

Форматом приложения был выбран блог. Мотивация к выбору данного формата заключалась в том, что именно в этом формате мы сможем задействовать как можно больше технологий, а также в дальнейшем мы можем использовать данное 2 для личного использования.

[1] https://spring.io/

[2] https://www.jetbrains.com/lp/devecosystem-2019/java/

Постановка задачи

Целью данной работы является освоение технолгии Spring посредством написания веб-приложения формата блога с авторизацией, редактированием профиля и возможностью добавлять сообщения. Проект коллективный: в нем участвуют два человека.

Для достижения поставленной цели были поставлены следующие задачи:

1. Разработать архитектуру приложения.

2. Разработать дизайн приложения.

3. Реализовать авторизацию пользователей.

4. Реализовать страницу с редактированием и добавлением постов.

5. Реализовать профиль пользователя.

6. Подключить микросервисы.

Для достижения поставленной цели от автора курсовой работы требуется:

1. Разработать архитектуру приложения.

2. Реализовать профиль пользователя и администратора.

3. Реализовать лайки.

4. Реализовать добавление постов.

5. Реализовать активацию аккаунта с помощью почты.

1. Обзор инструментов, используемых в приложении

1.1. FreeMarker

На данный момент наиболее распространенными шаблонизаторами для языка Java являются FreeMarker[3] и Thymeleaf[4]. При использовании шаблонизаторов верстка сводится к написанию html-шаблонов, не требующих глубоких знаний языка, а так же к написанию макросов для переиспользования кода. Было решено использовать именно FreeMarker по причине того, что он гораздо проще Thymeleaf, но его функциональности нам вполне достаточно.

1.2. Bootstrap

Разработка хорошего графического интерфейса достаточно сложный процесс, требующий опыта и знания языков для веб-разработки. В связи с тем, что мы хотели сосредоточиться на разработке логики приложения и меньше уделять времени веб-разработке, мы стали искать технологии, позволяющие сделать красивый интерфейс.
Мы решили использовать Bootstrap[5], потому что это один из самых популярных фреймворков, подходящих для новичков,[6], и его графические компоненты понравились нам больше, чем компоненты других фреймворков.

[3]https://freemarker.apache.org
[4]https://en.wikipedia.org/wiki/Thymeleaf
[5]https://getbootstrap.com/
[6]https://www.mockplus.com/blog/post/css-framework

2. Реализация

2.1. Архитектура

Перед началом работы над проектом было решено создать архитектуру, позволяющую легко добавлять новую функциональность проекту. Этого можно добиться строгим разделением логики между классами, а так же выносом общего кода в отдельные классы.

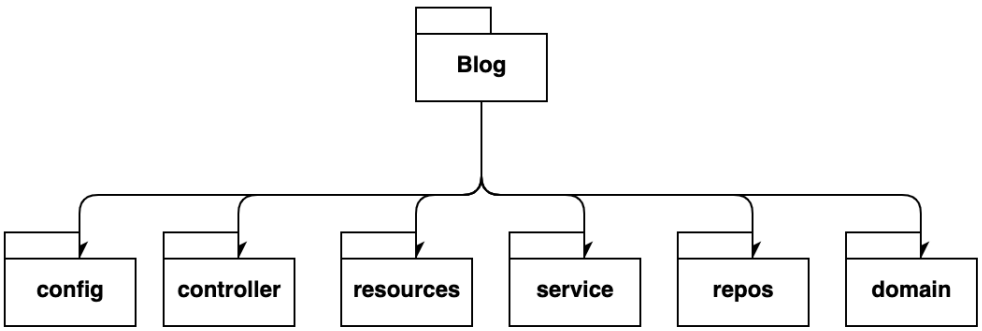

Рис. 1: Структура приложения

Архитектура приложения состоит из несколько пакетов:

1. config – в нем хранятся классы, отвечающие за настройку конфигураций.

2. controller – в нем хранятся классы, отвечающие за логику приложения.

3. domain – в нем хранятся классы, представляющие сущности приложения.

4. repos – в нем хранятся интерфейсы, отвечающие за взаимодействие с базами данным.

5. service – в нем хранятся классы со вспомогательными методами для сущностей.

6. resources – в нем хранятся файлы, отвечающие за дизайн приложения.

2.2. База данных

В качестве базы данных было решено использовать PostgreSQL[7]. Эта база данных была выбрана по нескольким причинам: во-первых, у нас уже был опыт использования реляционных СУБД, во-вторых, она является одной из самых востребованных СУБД на позицию Junior Java-разработчик. На стадии проектировки базы данных, было решено создать несколько таблиц (Рисунок):

1. users – хранит информацию о пользователях.

2. messages – хранит список сообщений.

3. likes – хранит информацию о том, какой пост лайкнул пользователь.

4. user_subscribers – хранит список подписчиков.

5. user_subscriptions – хранит список пользователей, на которых пользователь подписан.

Так как главной целью нашего проекта является изучение технологий Spring, для более удобного взаимодействия с базой данных было решено использовать фреймворк Spring Data JPA[8]. С его помощью мы можем заполнять таблицы при помощи создания сущностей, так же данная технология позволяет не писать самому стандартные запросы по типу «findByUsername», так как такие методы уже реализованы в библиотеке.

Между пользователям и сообщение было решено установить связь «OneToMany», а между пользователем и подписчиками связь «ManyToMany», так как пользователь может быть подписан на нескольких других пользователей.

2.3. Активация аккаунта

Активацию аккаунта можно описать следующим образом: на введенную пользователем почту пользователя отправляется активационная ссылка. Когда он переходит по ссылке, его аккаунт активируется

[7]https://www.postgresql.org/

[8]https://spring.io/projects/spring-data-jpa

и вносится в базу данных. При повторном переходе по ссылке пользователю сообщается о том, что она недействительна (Рисунок 11).

За отправку писем отвечает метод send у класса MailSender, в качестве аргументов ему приходит почта, на которую необходимо отправить письмо, текст и его тему.

Регистрация пользователя происходит в методе addUser у класса UserService (Рисунок 2), оттуда мы и вызываем метод send у класса MailSender.

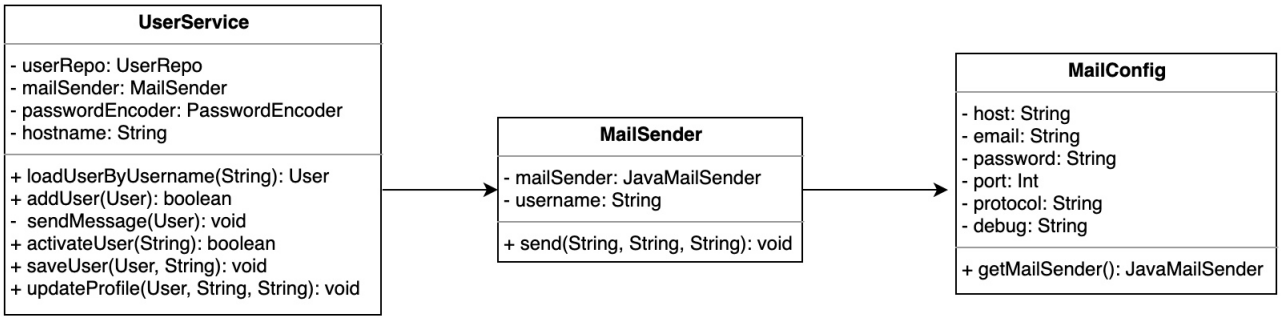

Рис. 2: Представление классов, осуществляющих отправку писем

2.4. Страница с сообщениями

Эта страница является основной в приложении, поэтому большую часть времени мы занимались именно ей. Она отвечает за отображение и добавление постов. На ней также можно фильтровать сообщения по тегам (Рисунок 8).

Отображать все сообщения на одной странице неудобно, поэтому было решено разделить их по страницам. Благодаря этому пользователь может удобно перемещаться между сообщениями. Также была добавлена возможность пользователю самому выбирать количество сообщений на странице (Рисунок 3).

Рис. 3: Навигационный блок

В приложении сообщения представляются в виде класса Message (Рисунок 4). У каждого сообщения (Рисунок 5) отображается:

1. Автор
2. Тег
3. Текст сообщения
4. Количество лайков
5. Приложенный файл
6. Кнопка редактирования сообщения

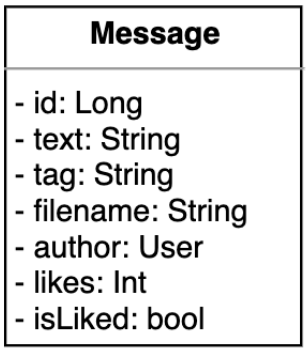

Рис. 4: Класс, представляющий сообщение

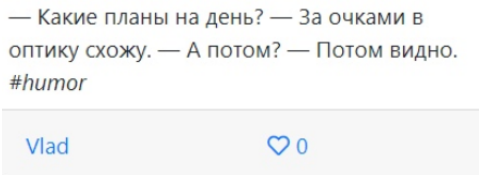

Рис. 5: Пример сообщения

2.4.1. Добавление новых сообщений

Добавление новых сообщений можно описать следующей диаграммой (Рисунок 6):

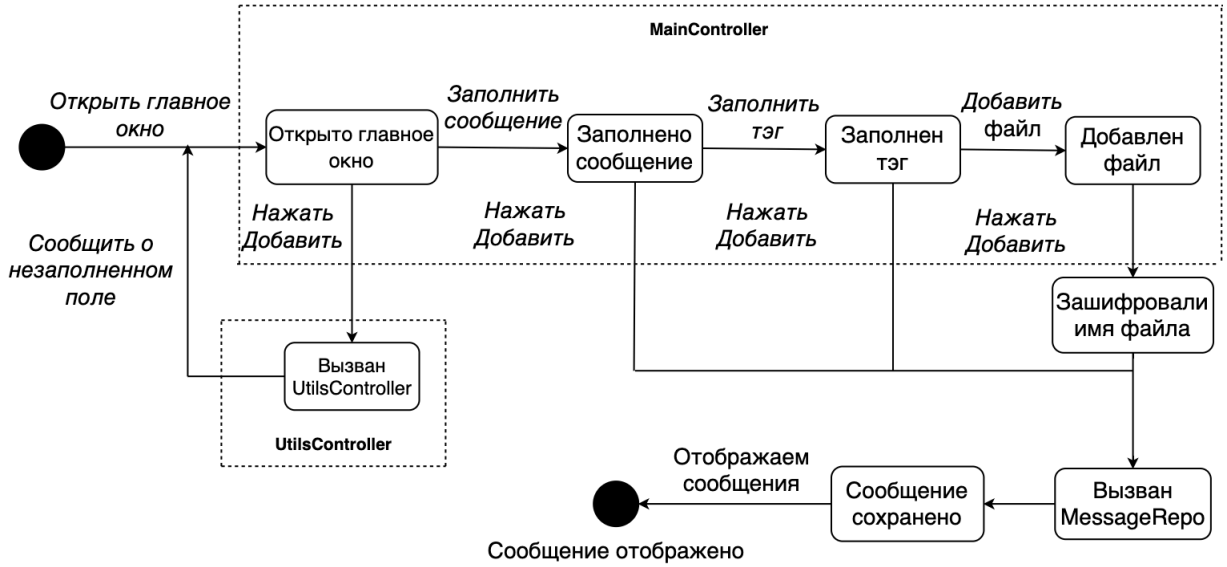

Рис. 6: Сценарий добавления сообщения

Если пользователь не заполнит графу сообщения, в таком случае он будет оповещен о том, что графу сообщений необходимо заполнить (Рисунок 9).

Помимо обычного текста есть возможность добавить тег и прикрепить файл. При загрузке файла на сервер мы в его название добавляем случайно сгенерированный ключ. Это сделано для того, чтобы не было проблем с загрузкой файлов с одинаковыми именами.

Добавление нового сообщение происходит с помощью метода add у класса MainController (Рисунок 7). Как только мы заходим в данный метод, мы проверяем переменную bindingResult на наличие ошибок, если они есть, тогда вызываем обработчик ошибок и говорим об ошибках пользователю, если никаких ошибок не было, в таком случае сохраняем сообщение и в качестве результата возвращаем строку, которая перенаправит нас на 1 страницу всех сообщений.

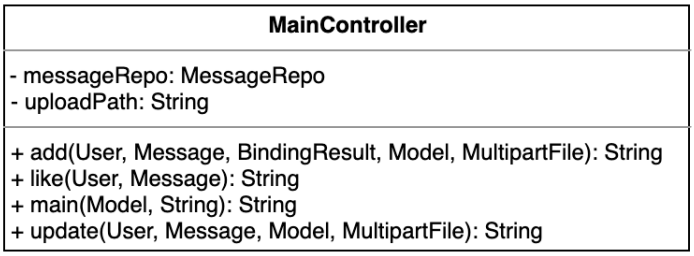

Рис. 7: Класс, представляющий окно с сообщениями

2.4.2. Реализвация лайков

Для того, чтобы реализовать лайки, необходимо хранить дополнительную информацию о том, какие посты лайкал пользователь. Это необходимо сделать для того, чтобы пользователю, лайкнувшему пост, отображалось закрашенное сердечко. Поэтому было решено в базу данных добавить таблицу, в которой будет 2 атрибута: id пользователя, который лайкнул пост и id поста, который он лайкнул.

Также к классу Message нам пришлось добавить 2 новых переменных: переменная, которая показывает, лайкнул ли пользователь данное сообщение или нет, а также переменная, отвечающая за общее количество лайков

2.5. Окно администратора

На данный момент в приложении есть 2 типа пользователей это обычный пользователь и администратор.

Было принято решение сделать окно администратора для того, чтобы в дальнейшем при появлении новых ролей администраторам было более комфортно назначать пользователям новые роли. Доступ к окну администратора есть только у пользователей с типом "ADMIN". В этом окне администратор видит список всех пользователей, а также имеет возможность назначать им новые роли и изменять их имена (Картинка 10).

2.6. Профиль пользователя

Также нами было принято решение добавить возможность пользователю изменить свой пароль и почту, к которой привязан аккаунт.

Смена почты происходит аналогичным способом как и при регистрации.

На введенную пользователем почту пользователя отправляется активационная ссылка, при переходе по которой, почта и пароль изменяются на введенные. Если по ссылке не перешли, то пароль и почта останутся прежними.

Результаты

В ходе выполнения данной работы были достигнуты следующие результаты:

1. Получен опыт работы с технологией Spring.

2. Реализована главная страница с сообщениями.

3. Разработана архитектура приложения.

4. Реализован профиль пользователя и администратора.

5. Реализованы лайки .

6. Реализована активация аккаунта с помощью почты.

Приложения

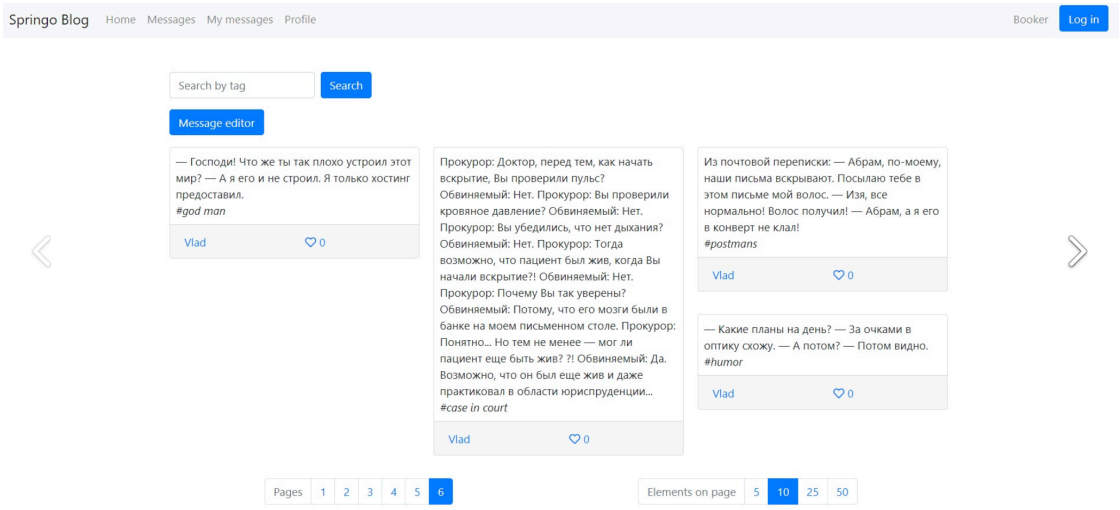

Рис. 8: Главная страница

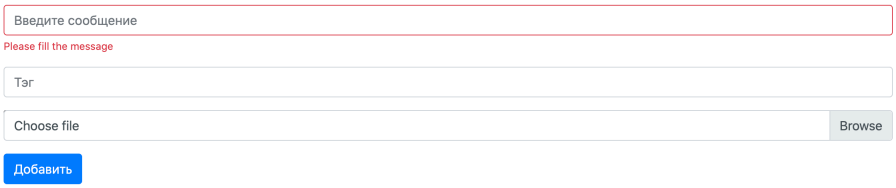

Рис. 9: Предупреждение о незаполненной графе

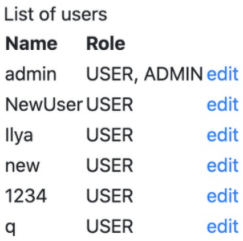

Рис. 10: Список пользователей

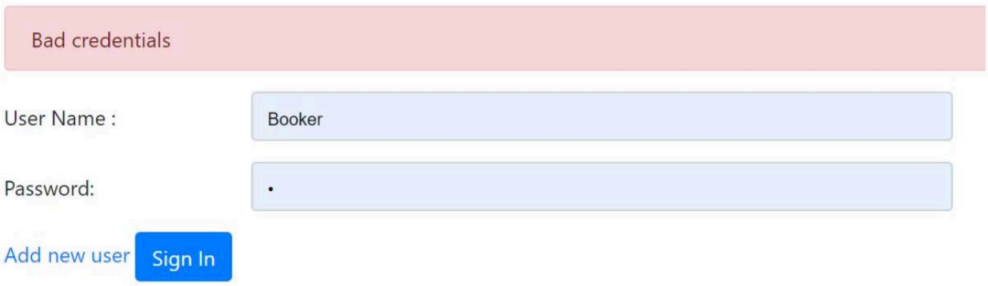

Рис. 11: Ошибка при активации аккаунта

Список литературы

[1] Apache. FreeMarker documantation (Дата обращения: 19.03.2020). — URL: `https://freemarker.apache.org/docs/index.html`.

[2] Apache. What is Apache FreeMarker? (Дата обращения: 18.03.2020). — URL: `https://freemarker.apache.org/`.

[3] Bootstrap. Bootstrap documantation (Дата обращения: 12.04.2020). — URL: `https://getbootstrap.com/`.

[4] JetBrains. Spring framework usage (Дата обращения: 02.06.2020). — 2019. — URL: `https://www.jetbrains.com/lp/devecosystem-2019/java/`.

[5] SpringCentral. Spring | Guides (Дата обращения: 26.02.2020). — URL: `https://spring.io/guides`.