МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий

Кафедра Информационные системы и технологии

Специальность 1–40 01 01 Программное обеспечение информационных

технологий

Специализация 1 –40 01 01 Программное обеспечение информационных

технологий (программирование интернет – изданий)

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**К КУРСОВОМУ ПРОЕКТУ НА ТЕМУ:**

«Социальная платформа для программистов»

Выполнил студент Чернявский Алексей Леонидович

(Ф.И.О.)

Руководитель проекта пр.-стаж. Бурмакова А.В.

(учен. степень, звание, должность, Ф.И.О., подпись)

Заведующий кафедрой к.т.н., доц. Смелов В.В.

(учен. степень, звание, должность, Ф.И.О., подпись)

Консультант: пр.-стаж. Бурмакова А.В.

(учен. степень, звание, должность, Ф.И.О., подпись)

Нормоконтролер: пр.-стаж. Бурмакова А.В.

(учен. степень, звание, должность, Ф.И.О., подпись)

Курсовой проект защищен с оценкой

Минск 2020

Реферат

ФИО

Подпись

Дата

Лист

1

КП 00.00.ПЗ

Разраб.

Чернявский А.Л.

Провер.

Бурмакова А.В.

Н. контр.

Бурмакова А.В.

Утверд.

Бурмакова А.В.

Реферат

Лит.

Листов

1

БГТУ 71171014, 2020

Пояснительная записка курсового проекта содержит 37 страниц пояснительной записки, 39 иллюстраций, 8 источников литературы, 2 приложения.

NODE.JS, JAVASCRIPT, MONGODB, EXPRESS.JS, MONGOOSE, MVC

Основная цель курсового проекта: разработка кроссплатформенной социальной платформы для программистов.

В первой главе проводится постановка задачи и обзор аналогов по тематике курсового проекта.

Вторая глава посвящена процессу проектирования системы.

В третьей главе описывается процесс разработки, принципы функционирования и назначение созданных компонент проекта.

В четвертой главе описано руководство пользователя, позволяющее подробно понять интерфейс программного средства.

В пятой главе приведены результаты тестирования проделанной работы.

В заключении приведены результаты проделанной работы.

Abstract

ФИО

Подпись

Дата

Лист

1

КП 00.00.ПЗ

Разраб.

Чернявский А. Л.

Провер.

Бурмакова А. В

Н. контр.

Бурмакова А.В.

Утверд.

Бурмакова А.В.

Abstract

Лит.

Листов

1

БГТУ 71171014, 2020

The explanatory note of the course project contains 30 pages of the explanatory note, 23 illustrations, 8 sources of literature, 3 appendices.

NODE.JS, JAVASCRIPT, MICROSOFT SQL SERVER, EXPRESS.JS, TEDIOUS, SEQUELIZE, MVC

The main goal of the course project: development of cross-platform software for order list management.

The first chapter provides an analytical review of the literature on the subject of the course project and contains an analysis of methods for solving the tasks.

The second chapter is devoted to the system design process.

The third chapter provides the rationale for technical techniques.

The fourth chapter describes testing, which allows you to understand in detail the interface of a software tool.

In conclusion, the results of the work done are presented.

Содержание

ФИО

Подпись

Дата

Лист

1

КП 00.00.ПЗ

Разраб.

Чернявский А.Л.

Провер.

Бурмакова А.В.

Н. контр.

Бурмакова А.В.

Утверд.

Бурмакова А.В.

Содержание

Лит.

Листов

1

БГТУ 71171014, 2020

[Введение 7](#_Toc41605661)

[1 Постановка задачи и обзор аналогов 8](#_Toc41605662)

[1.1 Постановка задачи 8](#_Toc41605663)

[1.2 Обзор аналогов 8](#_Toc41605664)

[2 Проектирование программного средства 11](#_Toc41605665)

[2.1 Проектирование серверной и клиентской частей 11](#_Toc41605666)

[2.2 Проектирование моделей данных 14](#_Toc41605667)

[3 Разработка функциональной модели и модели данных ПС 16](#_Toc41605668)

[3.1 Выполняемые функции 16](#_Toc41605669)

[3.2 Модель данных 17](#_Toc41605670)

[4 Руководство пользователя 18](#_Toc41605671)

[5 Тестирование 31](#_Toc41605672)

[Заключение 34](#_Toc41605673)

[Список литературы 35](#_Toc41605674)

[Приложение А 36](#_Toc41605675)

[Приложение Б 37](#_Toc41605676)

Введение

ФИО

Подпись

Дата

Лист

1

КП 00.00.ПЗ

Разраб.

Чернявский А.Л.

Провер.

Бурмакова А.В.

Н. контр.

Бурмакова А.В.

Утверд.

Бурмакова А.В.

Введение

Лит.

Листов

1

БГТУ 71171014, 2020

Целью данной работы была разработка кроссплатформенного приложения на тему «Социальная платформа для программистов». Приложение должно быть разработано с помощью языка JavaScript и программной платформы Node.js реализуя клиент-серверную архитектуру.

Node.js – программная платформа, основанная на движке Chrome V8. Node.js асинхронен и событийно-ориентирован. Данная платформа позволяет JavaScript взаимодействовать с устройствами ввода-вывода через свой API, подключать другие внешние библиотеки. Node.js предназначен для построения масштабируемых сетевых приложений, преимущественно серверов.

Npm – менеджер пакетов, входящий в состав Node.js, позволяющий расширить возможности приложения.

Для работы с Node.js были выбраны следующий пакеты:

* Express.js – предоставляет собой слой фундаментальных функций для упрощения разработки веб-приложений;
* Body-parser – пакет для парсинга тела запросов;
* Mongoose – ODM-библиотека (Object Data Modelling) для работы с MongoDB;

В качестве СУБД для базы данных была выбрана MongoDB. Для приложения создан кластер базы данных с помощью MongoDB Atlas.

Написанное приложение строится на клиент-серверной архитектуре. Для создания пользовательского интерфейса использовалась библиотека React. Для сервера был выбрана платформа Node.js, в связи с кроссплатформенностью, простотой развертки сервера и гибкости работы за счёт npm.

Постановка задачи и обзор аналогов

ФИО

Подпись

Дата

Лист

1

КП 01.00.ПЗ

Разраб.

Чернявский А.Л.

Провер.

Бурмакова А.В.

Н. контр.

Бурмакова А.В.

Утверд.

Бурмакова А.В.

1 Постановка задачи и обзор аналогов

Лит.

Листов

3

БГТУ 71171014, 2020

1.1 Постановка задачи

Для описания функциональных требований, определяющих функции, которые должно выполнять программное средство, и эксплуатационные требование, определяющие характеристики его функционирования следует сначала дать определение, что есть «социальная платформа» или «социальная веб-платформа».

Социальная веб-платформа – это программный продукт на базе и(возможно) с помощью которого пользователь или группа пользователей могут создать управляемое им (ими) же социальное веб-пространство. Это целевое пространство, в свою очередь, является основой для образования собственной (целевой) социальной сети.

Кратко можно сделать вывод, что приложение должно быть направлено на пользователей и взаимоотношения между ними.

Предполагается, что пользователями этой платформы будут являться программисты, QA-инженеры, менеджеры – люди, которые связаны с работой над IT-продуктами. Исходя из целевой аудитории программного продукта можно сделать вывод: аналогами будут являться различные IT-форумы.

1.2 Обзор аналогов

Одним из аналогов социальной платформы для программистов является «Форум программистов и сисадминов Киберфорум». Главная страница веб-приложения представлена на рисунке 1.1.

Функционал приложения включает в себя следующие действия:

* регистрация и авторизация;
* блоги и форумы программистов и сисадминов;
* посты и комментарии к ним.

Стоит отметить, что форум был создан еще в 2000 году и с тех пор дизайн не менялся. Из-за этого лояльность к приложению у молодого поколения довольно низкая.

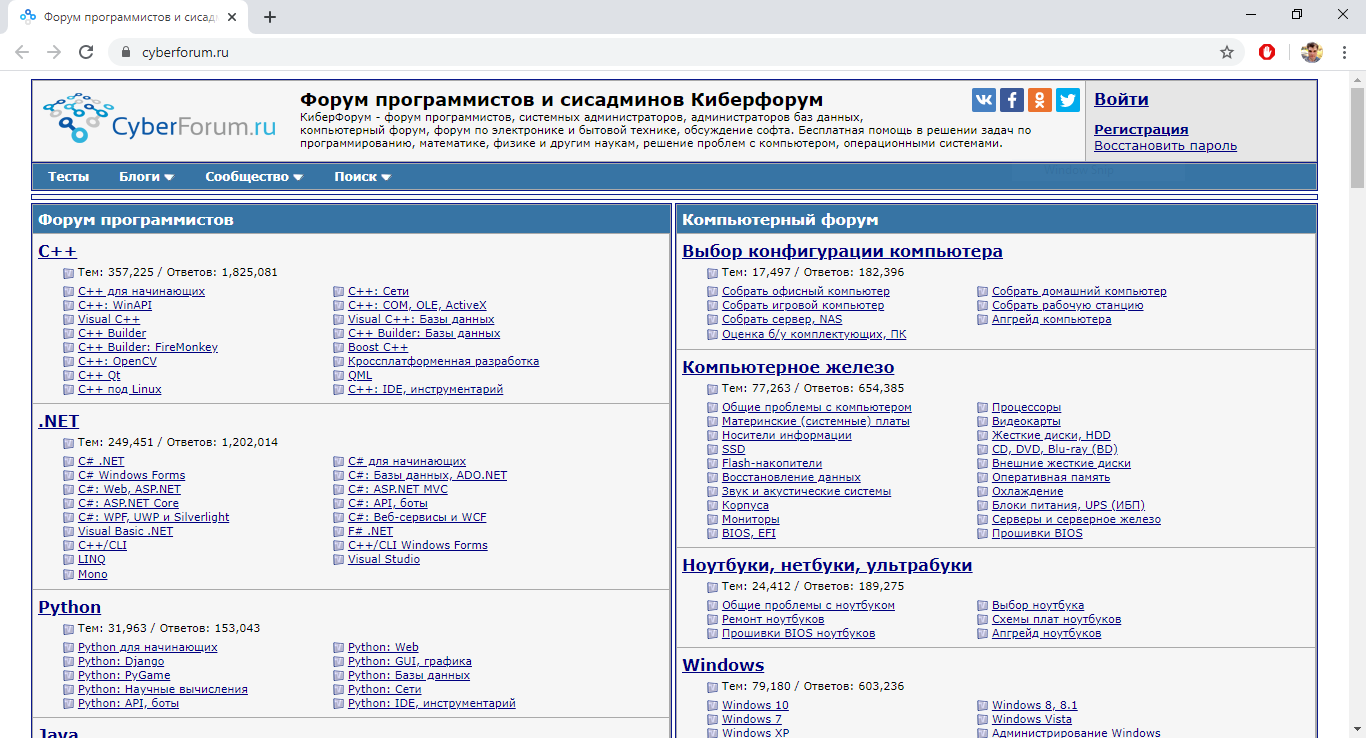


Рисунок 1.1 – «Форум программистов и сисадминов Киберфорум»

Еще один из ресурсов, на которые бы стоило обратить внимание – это «Хабр». Главную страницу можно увидеть на рисунке 1.2.

В самом начале работы это была небольшая новостная лента, но за 12 лет она превратилась в крупную площадку с многомиллионной аудиторией и необычными сервисами.

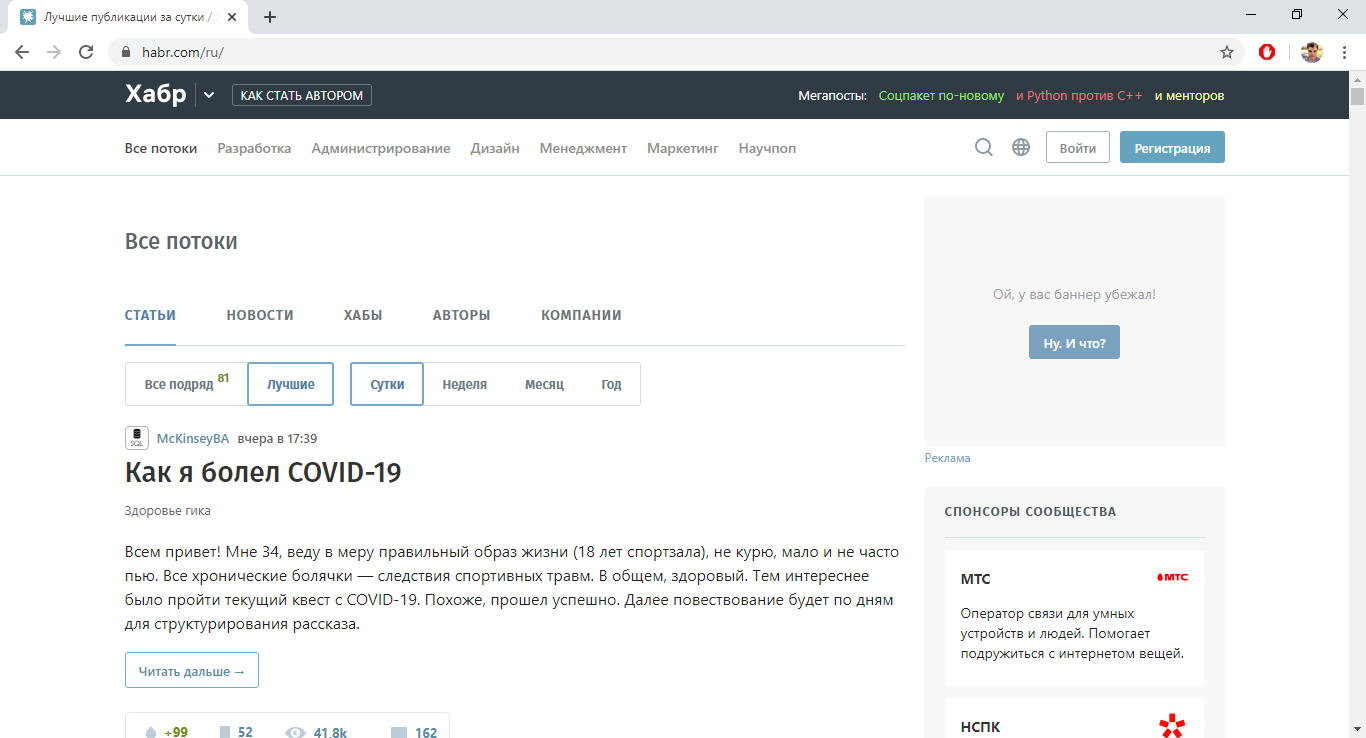


Рисунок 1.2 – «Хабр» - веб-приложение habr.com

Пользователи пишут сюда интересные материалы на разные темы и комментируют их. Тут можно задавать вопросы и получать ответы, обсуждать актуальные темы, искать работу или фриланс и напрямую общаться с крупными компаниями.

Одной из ключевых особенностей является открытость постов и комментариев к ним, быстрая регистрация и авторизация, user-friendly интерфейс.

Отличием нашего приложение от аналогов является сокрытие постов от незарегистрированных пользователей для большей приватности сообщества и тесного сотрудничества именно внутри системы.

Проектирование программного средства

ФИО

Подпись

Дата

Лист

1

КП 02.00.ПЗ

Разраб.

Чернявский А.Л.

Провер.

Бурмакова А.В.

Н. контр.

Бурмакова А.В.

Утверд.

Бурмакова А.В.

2 Проектирование программного средства

Лит.

Листов

5

БГТУ 71171014, 2020

2.1 Проектирование серверной и клиентской частей

Серверная часть написана на Node.js с помощью Express, Mongoose, Socket.io. Node.js – это кроссплатформенная среда исполнения JS, которая выполняет JavaScript вне web-браузера. Node.js позволяет разработчикам использовать JavaScript для написания сценариев на стороне сервера, а также для создания динамического содержимого веб-страницы перед ее отправкой в web-браузер пользователя. Таким образом, мы объединяем разработку web-приложения вокруг одного языка программирования, а не разных языков для серверных и клиентских сценариев.

Express – это минимальная и гибкая среда web-приложений Node.js, которая предоставляет надежный набор функций для приложений.

Mongoose – ODM-библиотека (Object Data Modelling) для работы с MongoDB, которая позволяет сопоставлять объекты классов и документы коллекций из базы данных.

Socket.io – JS-библиотека для web-приложений и обмена данными в реальном времени поверх протокола WebSocket.

Приложение имеет клиент-серверную архитектуру (Рис. 2.1)

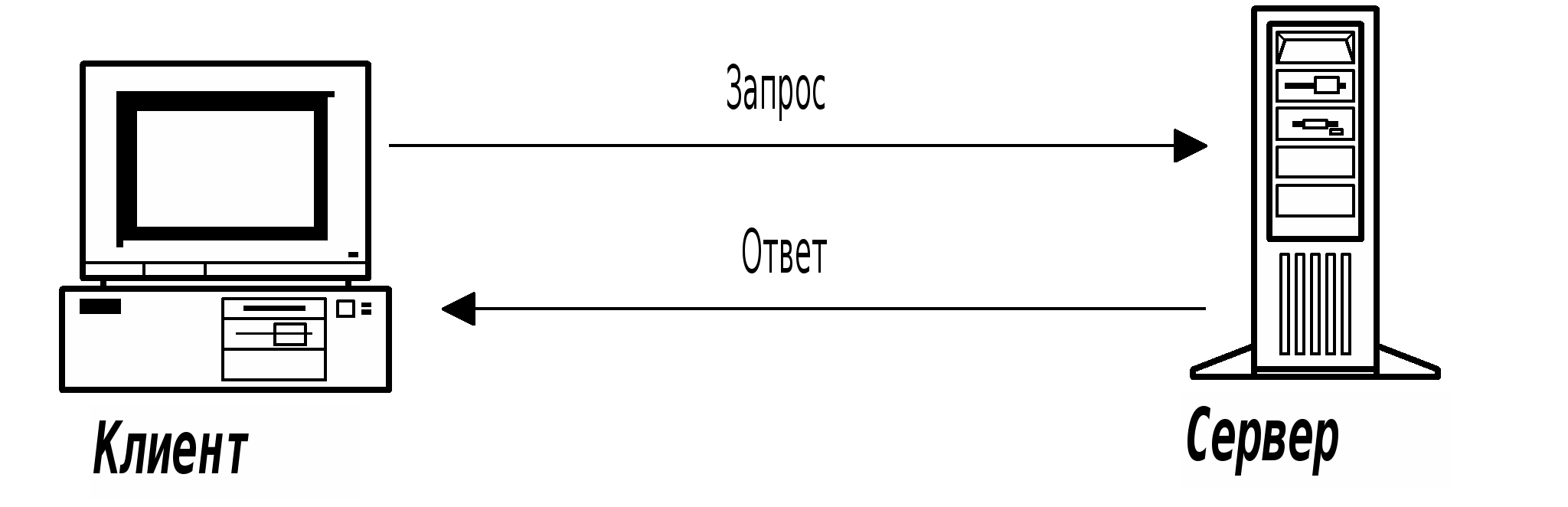


Рисунок 2.1 – Клиент-серверная архитектура

Клиент является браузерным приложением, которое посылает запросы (request) по сети с помощью протокола HTTP, а сервер должен принимать запросы от клиента и возвращать ответ (response). В нашем случае приложение разделено на две части. Это front-end (клиентская) и back-end (серверная).

Клиентская часть разработана с помощью библиотеки React. Следует отметить, что клиентская часть приложения имеет архитектуру Single Page Application (SPA-архитектуру). Отличия от классической Multi Page архитектуры можно рассмотреть на рисунке 2.2.

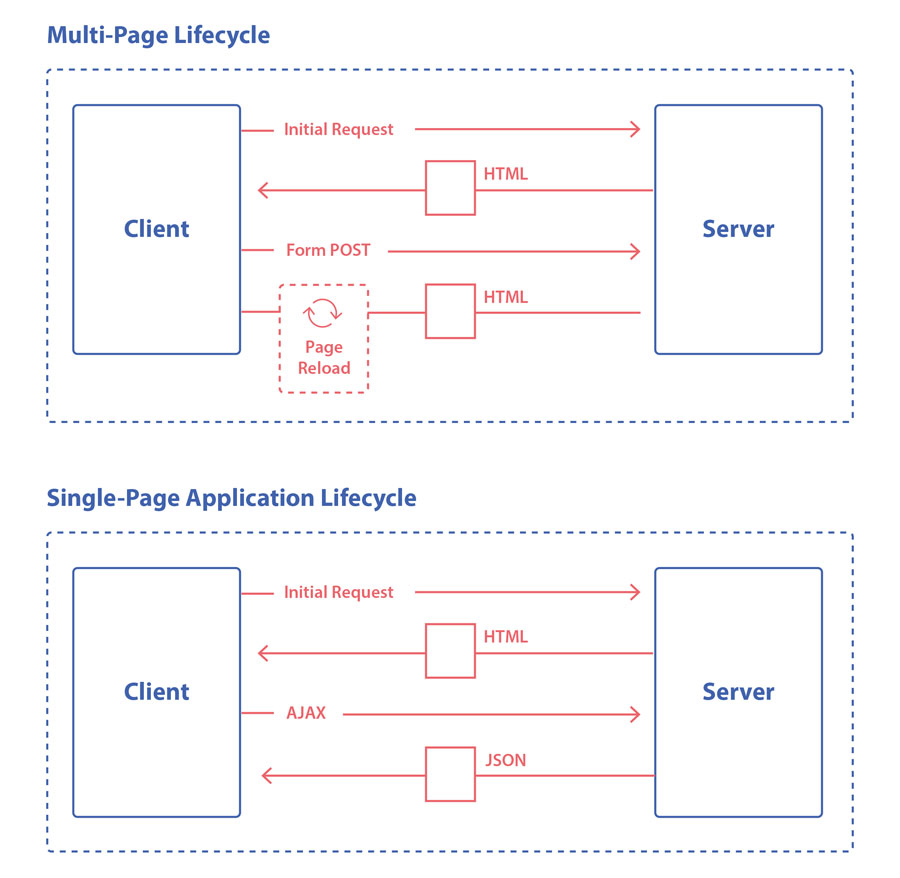


Рисунок 2.2 – Отличия SPA от Multi Page

Это значит, что в SPA архитектуре клиент запрашивает основную HTML-страницу только один раз, а затем общается с сервером с помощью AJAX запросов, при этом перезагрузки страницы не происходит, разметка меняется динамически.

В традиционных приложениях сервер при каждом запросе должен отсылать HTML-страницу.

Стоит отметить, что при разработке приложения учитывался MVC шаблон.

Модель предоставляет остальным компонентам приложения объектно-ориентированное отображение данных (например, данные о пользователе, список постов, список комментариев). Объекты модели осуществляют загрузку и сохранение данных в базе данных и реализуют бизнес-логику.

На рисунке 2.3 можно увидеть взаимодействие между слоями MVC-приложения.

Так как в нашем случае в клиентской части используется SPA, то View будет являться JSON-ответ вместо отдельной HTML-страницы под конкретный action.

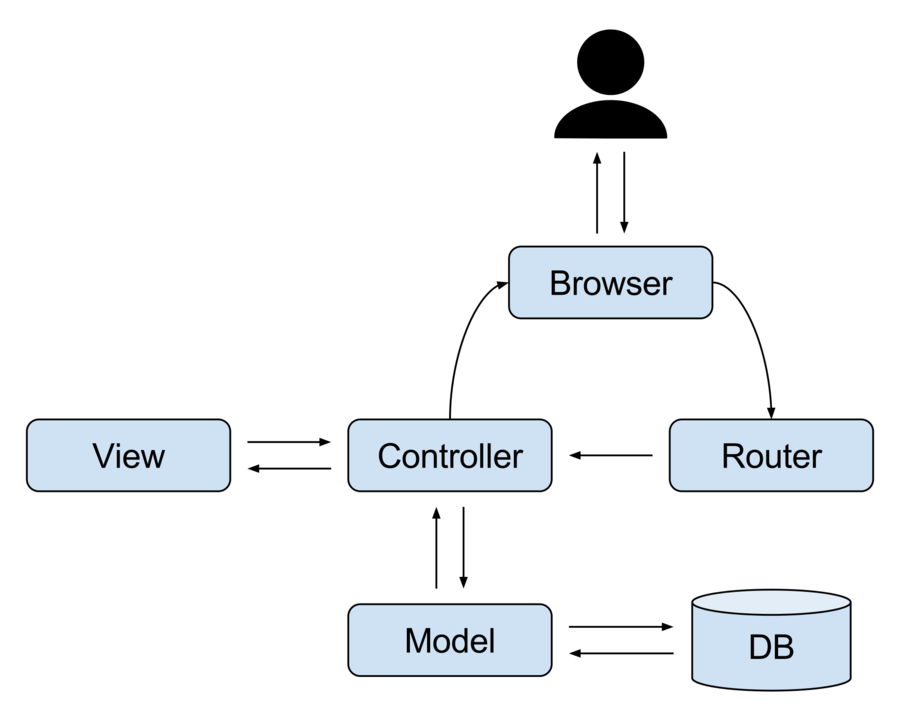


Рисунок 2.3 – Взаимодействие между слоями MVC

Структуру файлов и папок приложения можно увидеть на рисунке 2.4.

В папке «config» находятся файлы конфигурации, а также файл подключения к базе данных, «routes» – маршрутизация и контроллеры, «models» – модели данных, «middleware» – промежуточная обработка некоторых запросов, например тех, которые должны выполняться только для авторизованных пользователей. Файл «server.js» является точкой запуска приложения.

В папке «client» расположены файлы клиентской части. Приложение-клиент представляет собой web-приложение, написанное на языках JavaScript, HTML, CSS. Для рендеринга страниц используется React. React – JavaScript-библиотека для создания пользовательских интерфейсов. Создаются простые представления для каждого состояния вашего приложения, и React будет эффективно обновлять и отрисовывать только нужные компоненты при изменении данных. Компоненты React реализуют метод render(), который принимает входные данные и возвращает данные для отображения. Используется XML-подобный синтаксис JSX. В нашем приложении используется функциональные компоненты. Для отслеживания состояния используется технология Redux.

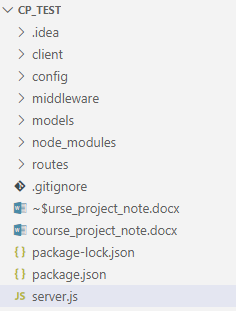


Рисунок 2.4 – Структура файлов и папок приложения

Для отправки запросов с SPA приложения разработаны «end-points». Их можно увидеть на рисунке 2.5.

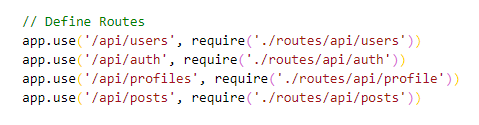


Рисунок 2.5 – API пути для взаимодействия с сервером

2.2 Проектирование моделей данных

MongoDB – это документарная база данных. Вместо хранения данных в таблицах, состоящих из отдельных строк, как в реляционных базах, MongoDB сохраняет данные в коллекциях, состоящих из документов. Документ – это большой JSON объект без заранее определенного формата и схемы.

Для приложения был создан кластер MongoDB в облачном сервисе MongoDB Atlas. Используются три коллекции (Рис. 2.6)

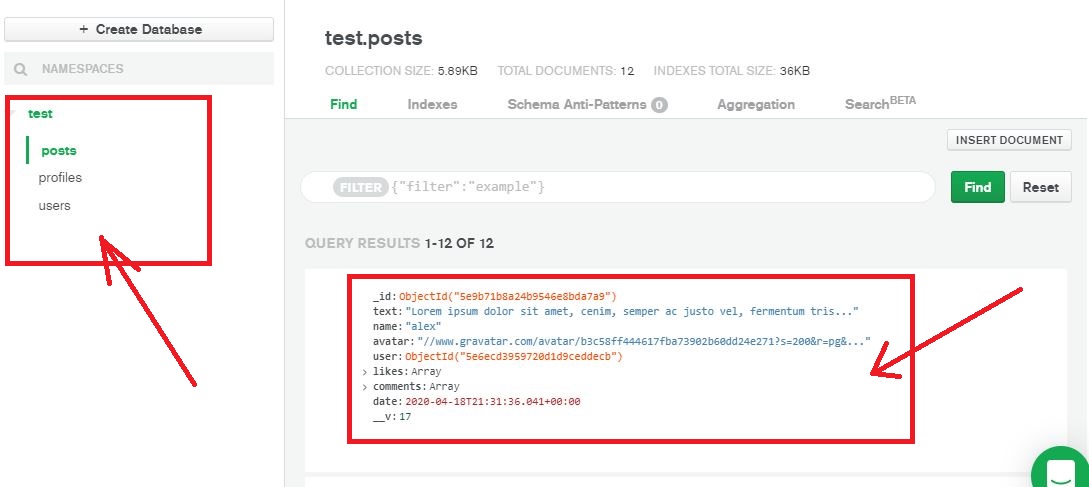


Рисунок 2.6 – Кластер в MongoDB Atlas

Схема в Mongoose определяет метаданные модели – ее свойства, типы данных и ряд другой информации. Созданный кластер содержит коллекции для работы системы.

Разработка функциональной модели и модели данных ПС

ФИО

Подпись

Дата

Лист

1

КП 03.00.ПЗ

Разраб.

Чернявский А.Л.

Провер.

Бурмакова А.В.

Н. контр.

Бурмакова А.В.

Утверд.

Бурмакова А.В.

3 Разработка функциональной модели и модели данных ПС

Лит.

Листов

2

БГТУ 71171014, 2020

3.1 Выполняемые функции

Разрабатываемое приложение представляет собой социальную платформу для программистов, где главная цель – это обсуждение каких-либо тем. В результате были созданы необходимые модели для создания постов и комментариев к ним. У каждого пользователя есть возможность заполнить информацию о себе (о своих предыдущих и текущих проектах, об образовании).

В приложении есть система оценивания постов в виде «лайков».

Также добавлены уведомления при действиях со стороны других пользователей («лайки», комментарии).

Функционально программное средство должно выполнять следующие задачи:

* регистрация и авторизация пользователей;
* добавление и редактирование информации об образовании и проектах пользователя;
* создание, изменения, удаление постов и комментариев к ним;
* возможность оценивать комментарии «лаками»;
* уведомления пользователей о действиях со стороны других пользователей.

При авторизации используется JSON Web Token (JWT), при любом запросе клиент должен его посылать на сервер. JWT имеет некоторые преимущества – он самодостаточен, все необходимые данные для аутентификации можно хранить прямо в самом токене.

Мы можем рассмотреть устройство токена. JWT состоит из трёх основных частей: заголовка (header), нагрузки (payload) и подписи (signature). Заголовок и нагрузка формируются отдельно, а затем на их основе вычисляется подпись с помощью npm-пакета «jsonwebtoken». После первого входа в веб-приложение клиенту возвращается сгенерированный сервером токен. На стороне клиента этот токен кладется в LocalStorage браузера. При каждом последующем запросе токен извлекается из LocalStorage и посылается в HTTP-заголовке ‘x-auth-token’. На стороне сервера пришедший токен проверяется с помощью метода «verify» библиотеки. В Приложении А находится middleware по проверке авторизации пользователя.

3.2 Модель данных

В рамках курсового проекта было разработано три схемы моделей данных.

Схема User хранит в себе информацию о пользователях для входа в систему:

* name – имя пользователя;
* email – электронная почта пользователя – логин для входа в систему;
* password – пароль для входа в систему;
* avatar – аватар профиля;
* date – дата регистрации пользователя.

Схема Profile содержит подробную личную информацию о пользователях, биографию, ссылки на социальные сети, опыт:

* user – ссылка на объект из коллекции Users, т.е. каждый профиль привязан к определенному пользователю;
* company – компания;
* website – веб-сайт;
* location – расположение (место проживания);
* status – степень seniority программиста;
* skills – навыки и умения пользователя;
* bio – краткая биография;
* githubusername – имя пользователя в Github;
* experience – прошлый и текущий опыт в проектах;
* education – образование;
* social – ссылки на популярные социальные сети;
* date – дата создания профиля.

Схема Post хранит данные о постах и комментариях:

* user – ссылка на объект из коллекции User;
* text – текст поста;
* name – имя пользователя, создавшего пост;
* avatar – аватар профиля;
* likes – «лайки» пользователей;
* comments – комментарии пользователей;
* date – дата создания;

Стоит отметить, что на каждое поле модели существует ограничение целостности в серверной части. Листинги моделей представлены в Приложении Б

Руководство пользователя

ФИО

Подпись

Дата

Лист

1

КП 04.00.ПЗ

Разраб.

Чернявский А.Л.

Провер.

Бурмакова А.В.

Н. контр.

Бурмакова А.В.

Утверд.

Бурмакова А.В.

4 Руководство пользователя

Лит.

Листов

12

БГТУ 71171014, 2020

Если пользователь не зарегистрирован или не авторизован, главная страница выглядит следующим образом (Рис. 4.1).

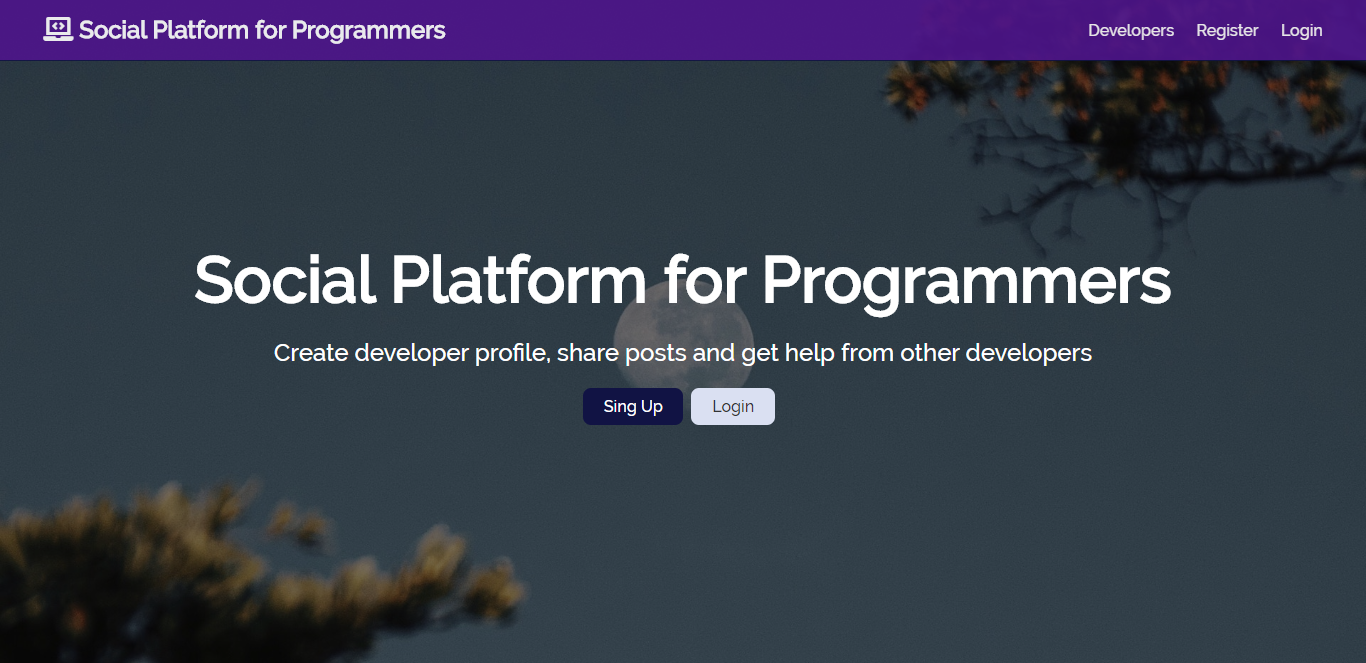


Рисунок 4.1 – Главная страница

Пользователю предлагается перейти на страницы входа или регистрации нажав на кнопки на главной странице, либо же можно просмотреть список пользователей, нажав на кнопку в верхней части приложения.

Форма входа представлена на рисунке 4.2.

Для авторизации пользователю необходимо ввести свой e-mail и пароль.

Если же пользователь не зарегистрирован в приложении, ниже формы ему предлагается перейти на страницу регистрации.

Форма регистрации представлена на рисунке 4.3.

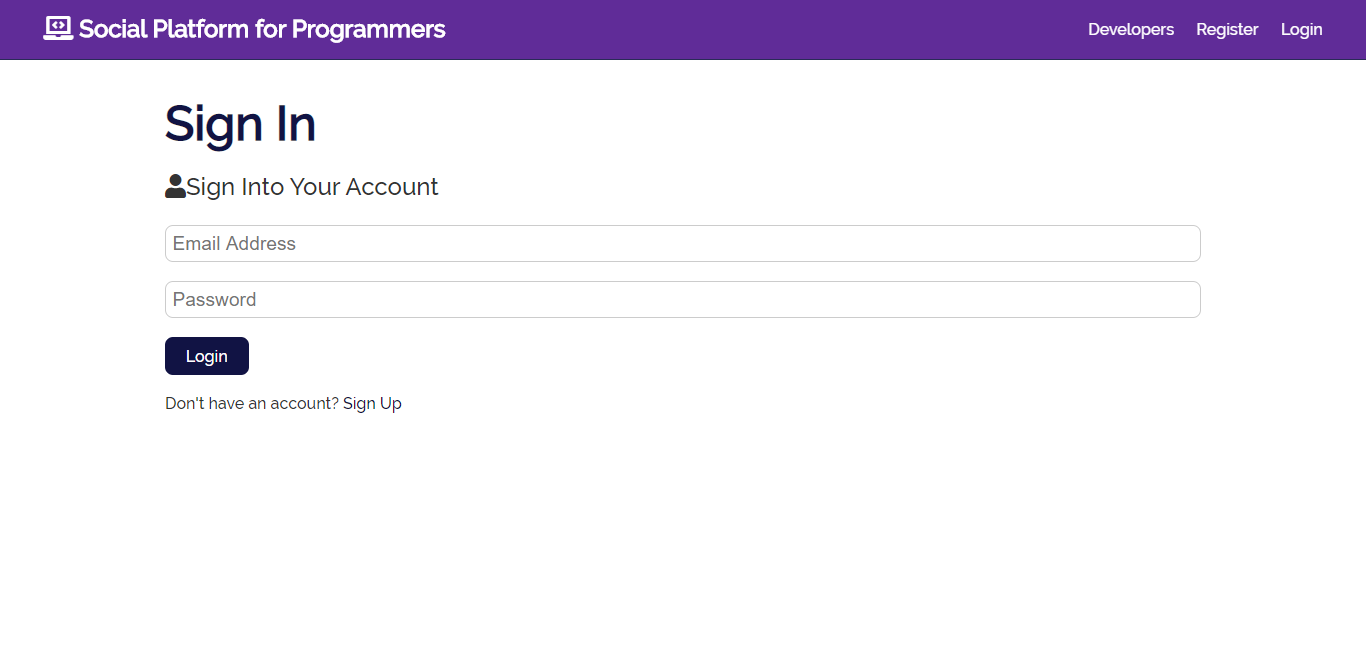


Рисунок 4.2 – Форма входа в приложение

При регистрации необходимо указать имя, e-mail и пароль. Пароль нужно подтвердить.

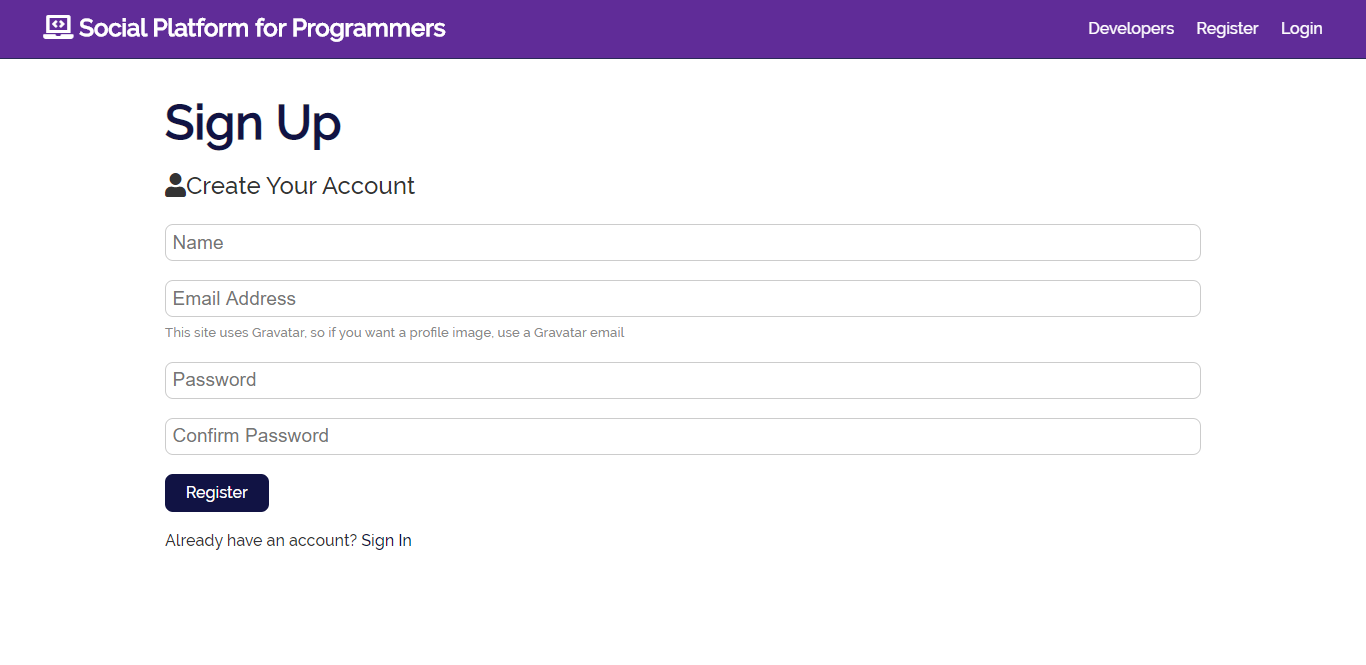


Рисунок 4.3 – Форма регистрации в системе

После входа система перенаправляет пользователя на Dashboard страницу, где указано приветствие и предлагается создать профиль пользователя (Рис. 4.4).

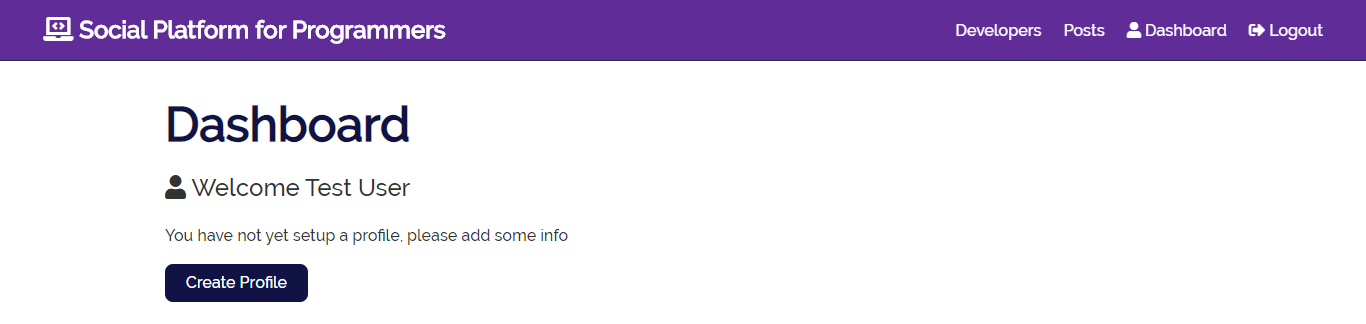


Рисунок 4.4 – Dashboard страница

Также для авторизированного пользователя меняется шапка сайта – добавляются ссылки на посты, Dashboard, Logout – для выхода из системы.

При нажатии пользователем на кнопку Create Profile система перенаправляет на страницу создания профиля. Форму создания профиля можно увидеть на рисунке 4.5

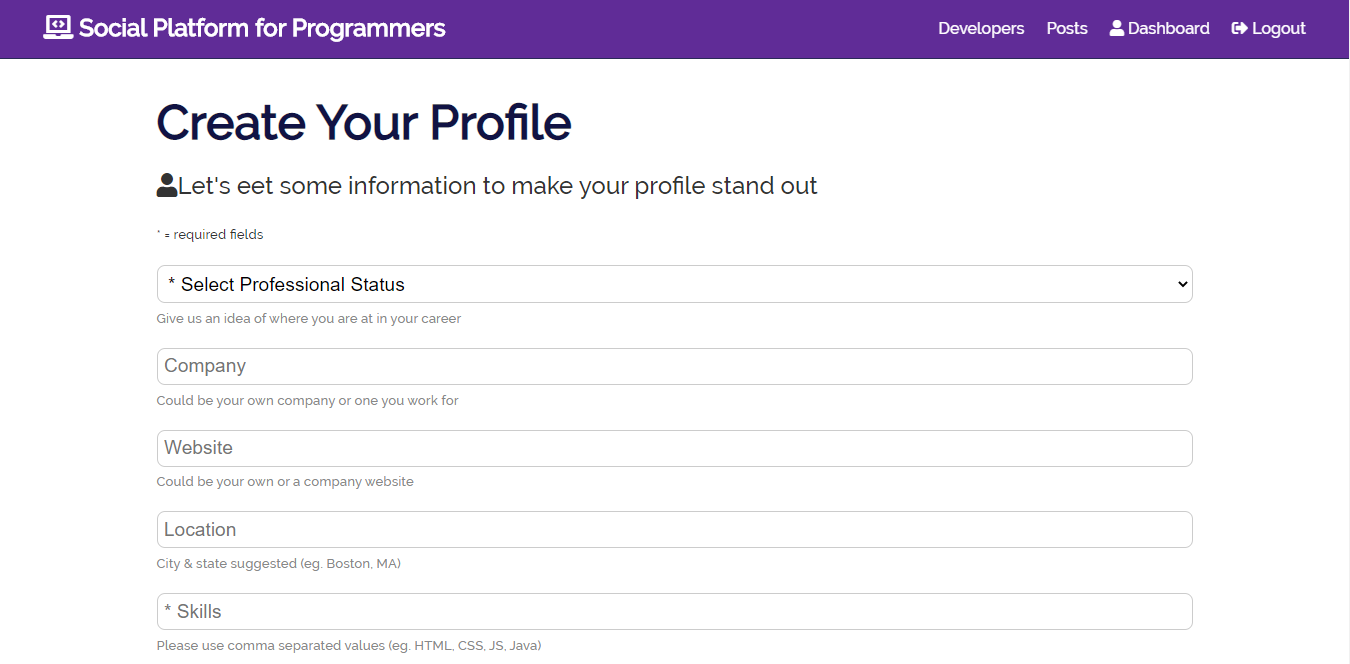


Рисунок 4.5 – Форма создания профиля

Звёздочкой указаны обязательные поля (статус, навыки). Также есть необязательные поля (компания, вебсайт, Github имя пользователя, локация, биография) (Рис. 4.6)

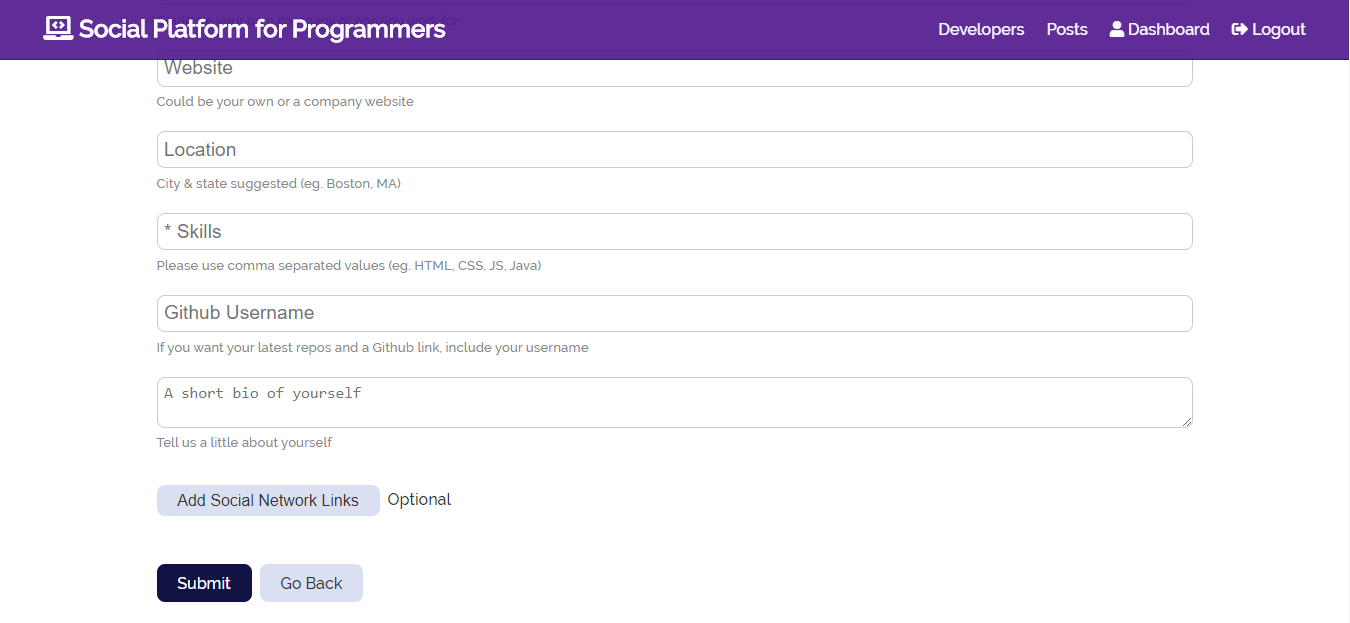


Рисунок 4.6 – Необязательные поля формы создания профиля

Есть возможность добавить ссылки на социальные сети нажав на кнопку Add Social Network Links. При нажатии на кнопку раскрываются поля для ввода ссылок (Рис. 4.7).

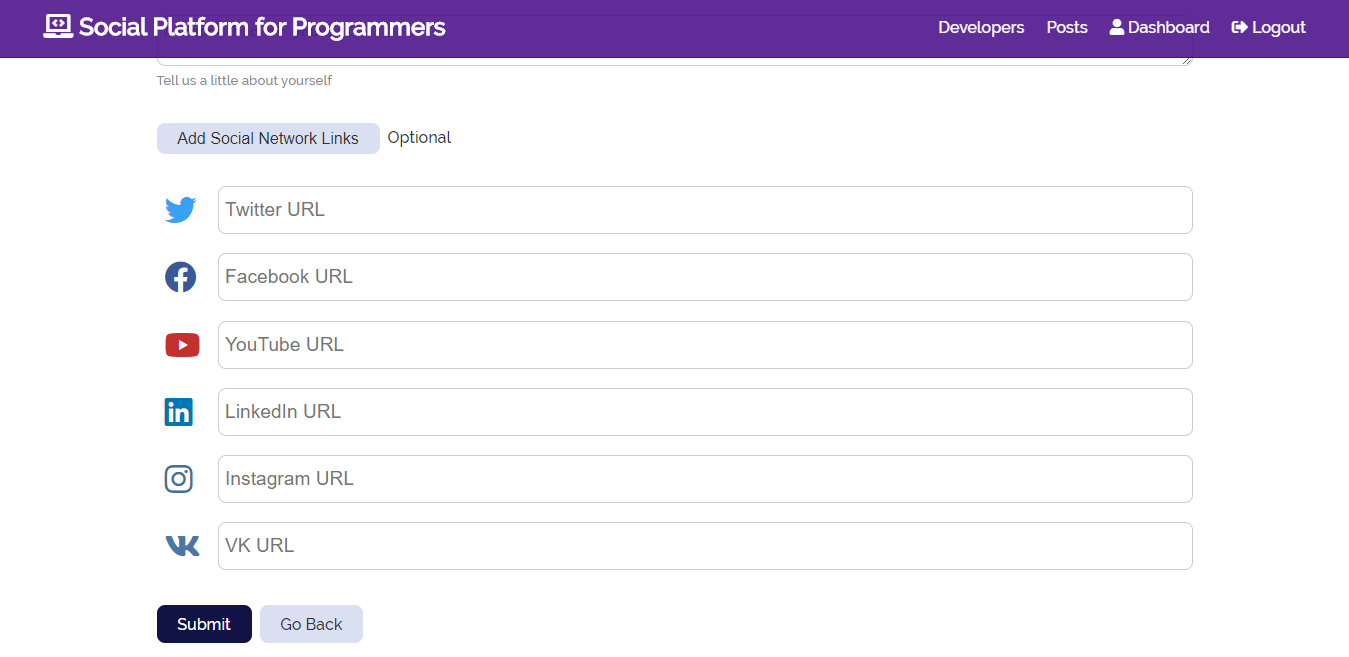


Рисунок 4.7 – Поля для ввода ссылок на социальные сети

Для отправки данных на сервер необходимо нажать кнопку Submit либо же нажать на кнопку Go Back, чтобы вернуться.

После создания профиля изменяется Dashboard страница (Рис 4.8).

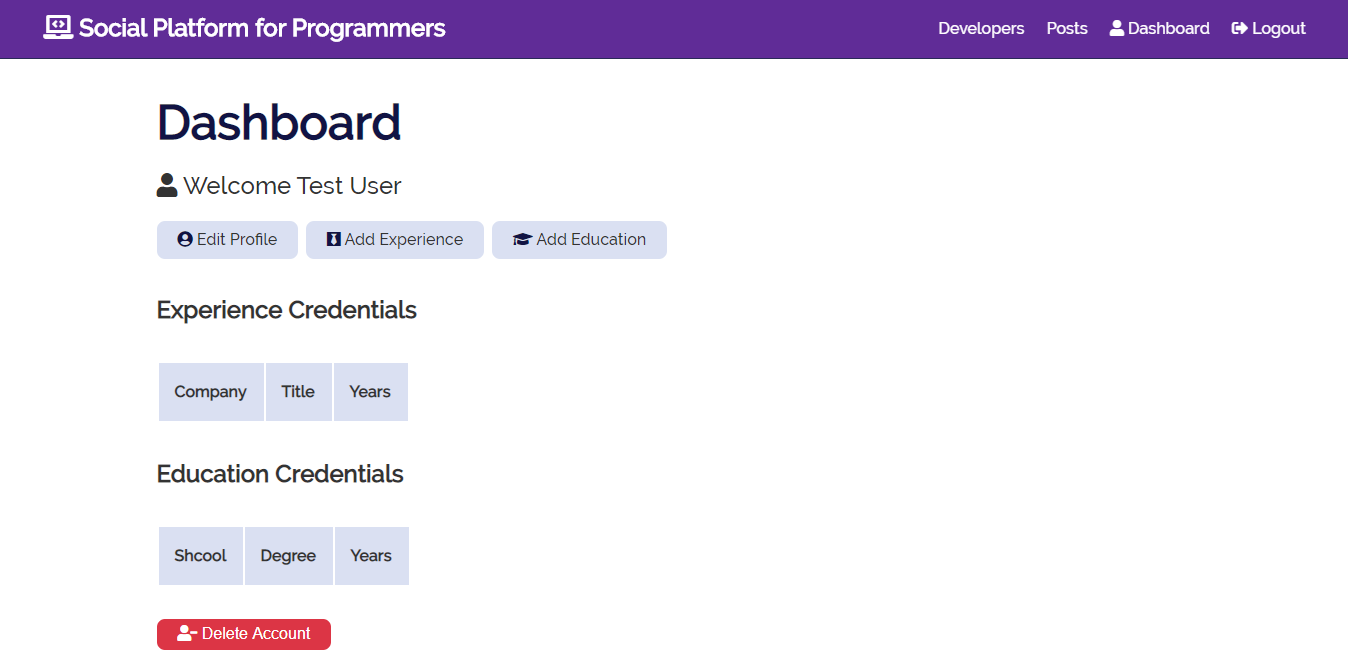


Рисунок 4.8 – Страница Dashboard после создания профиля

На ней появляются кнопки для изменения профиля, добавления опыта работы и информации об образовании. Также есть кнопка для удаления профиля.

Нажмем на кнопку Add Experience. Система перенаправляет пользователя на страницу с формой (Рис. 4.9).

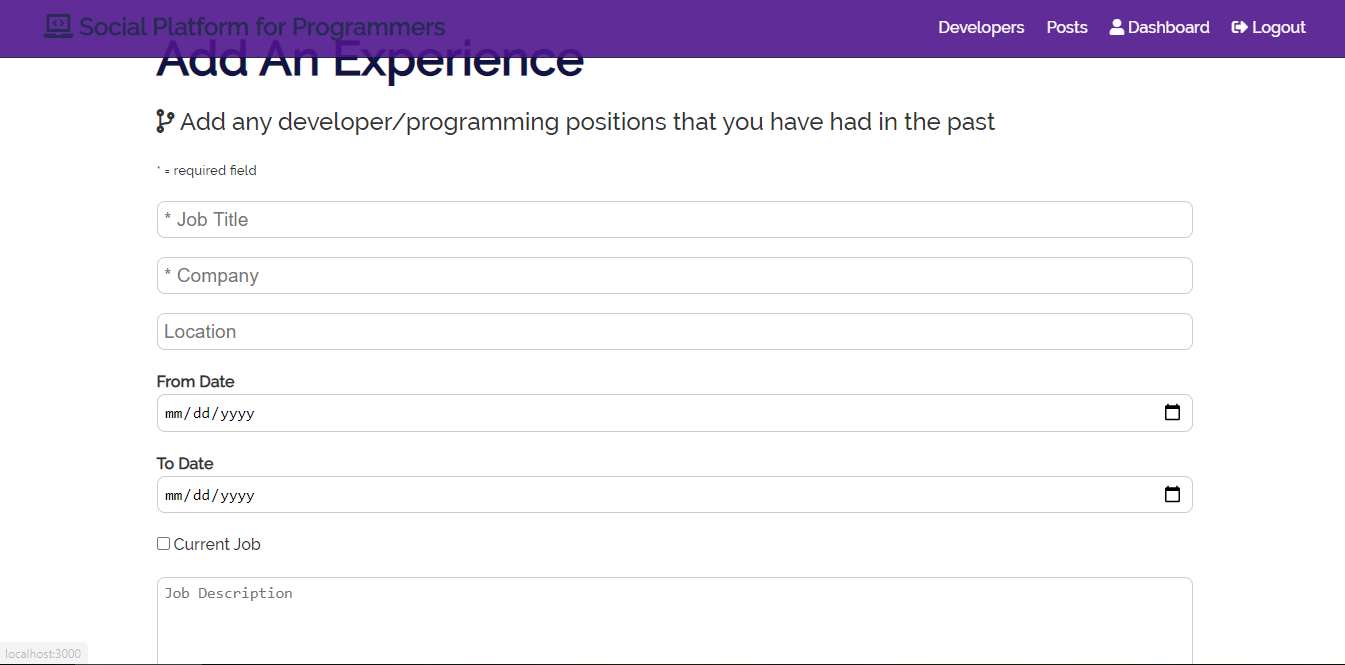


Рисунок 4.9 – Форма добавления данных об опыте работы

Обязательными полями являются поле Должность, Компания, Дата начала работы. Необязательные поля: Расположение, Дата окончания работы, Описание работы.

Заполним данные и нажмем на кнопку Submit. Система перенаправит пользователя на страницу Dashboard с добавленными данными о работе (Рис. 4.10).

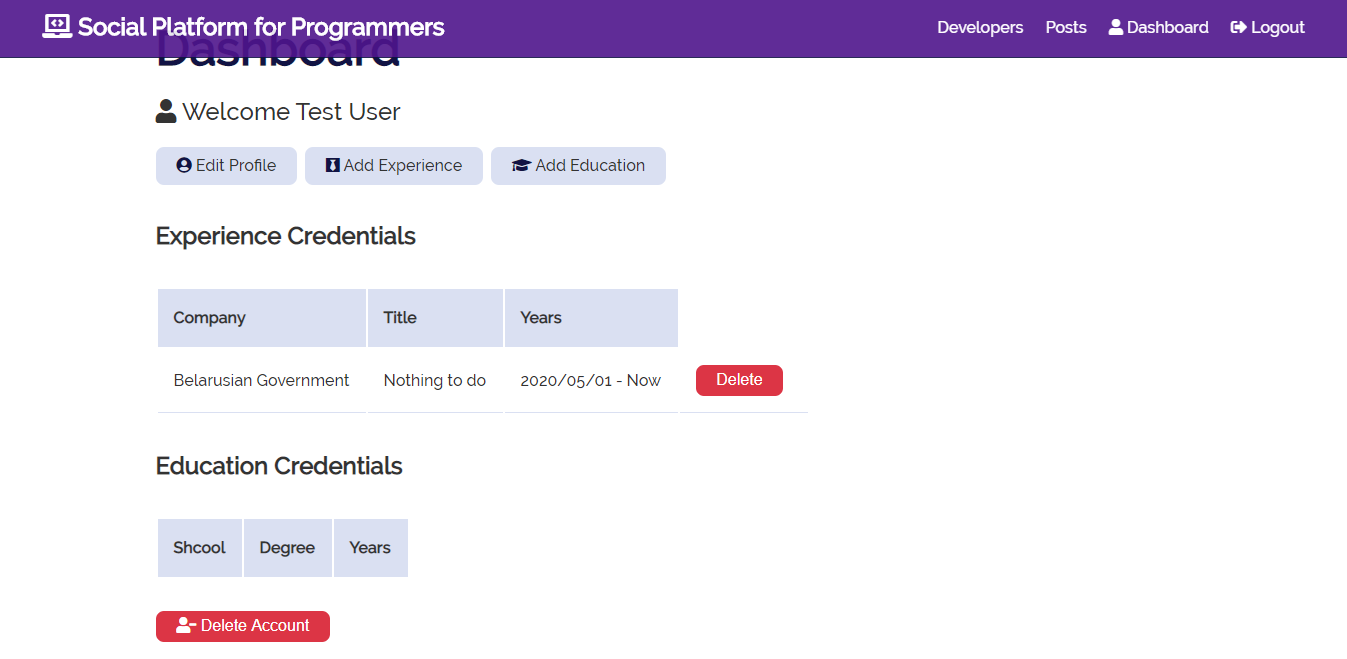


Рисунок 4.10 – Dashboard с данными о работе

В приложении есть возможность добавить несколько мест работы.

Кликнем на кнопку Add Education. Система перенаправляет пользователя на страницу добавления данных об образовании. Форма идентична с формой добавления данных о работе, за исключением некоторых обязательных полей

Нужно обязательно указать школу или университет, ученую степень и дату начала обучения (Рис. 4.11).

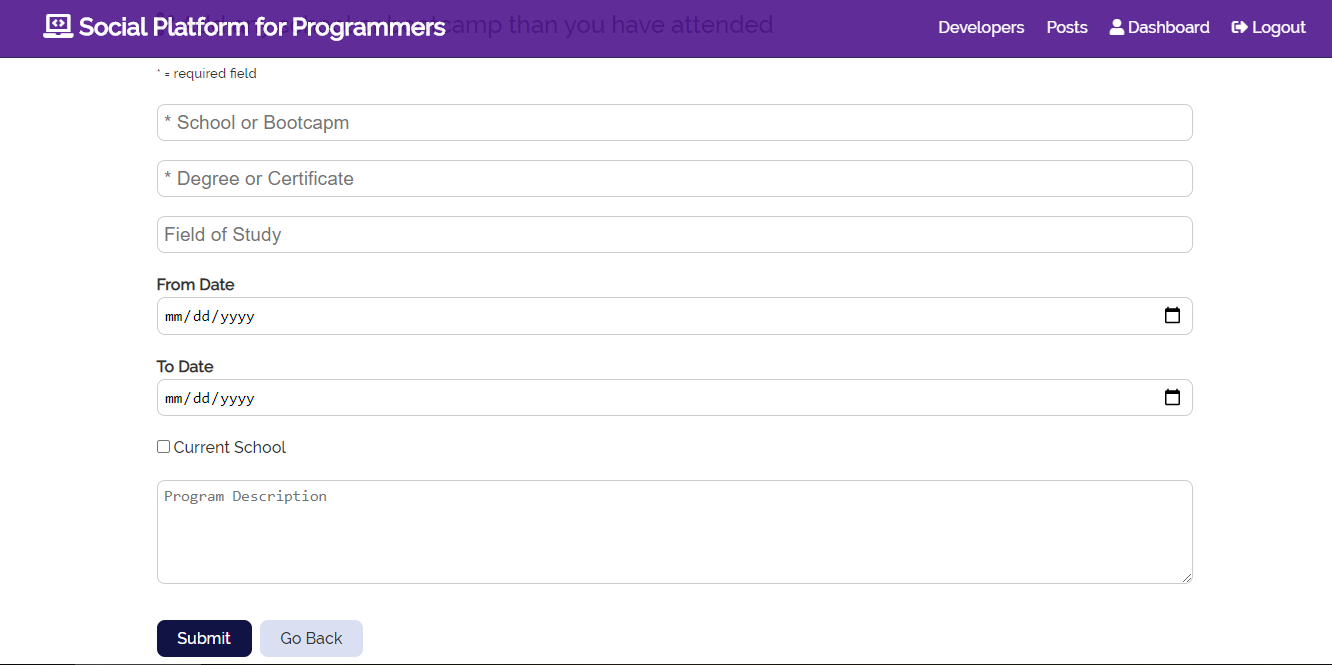


Рисунок 4.11 – Форма добавления данных об образовании

После добавления данных система перенаправляет пользователя на страницу Dashboard. Как видим – данные добавились в профиль (Рис. 4.12).

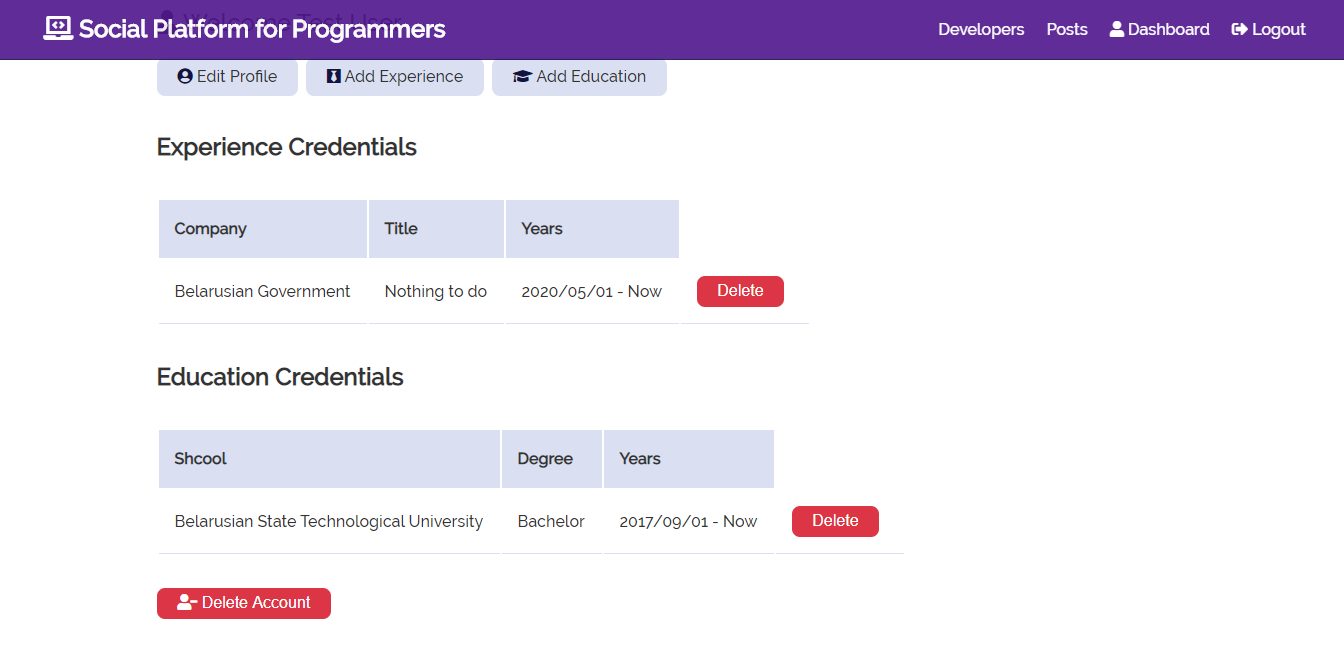


Рисунок 4.12 – Dashboard после добавления данных об образовании

При нажатии на кнопку Developers в шапке сайта система перенаправит на страницу со списком всех разработчиков. (Рис 4.13)

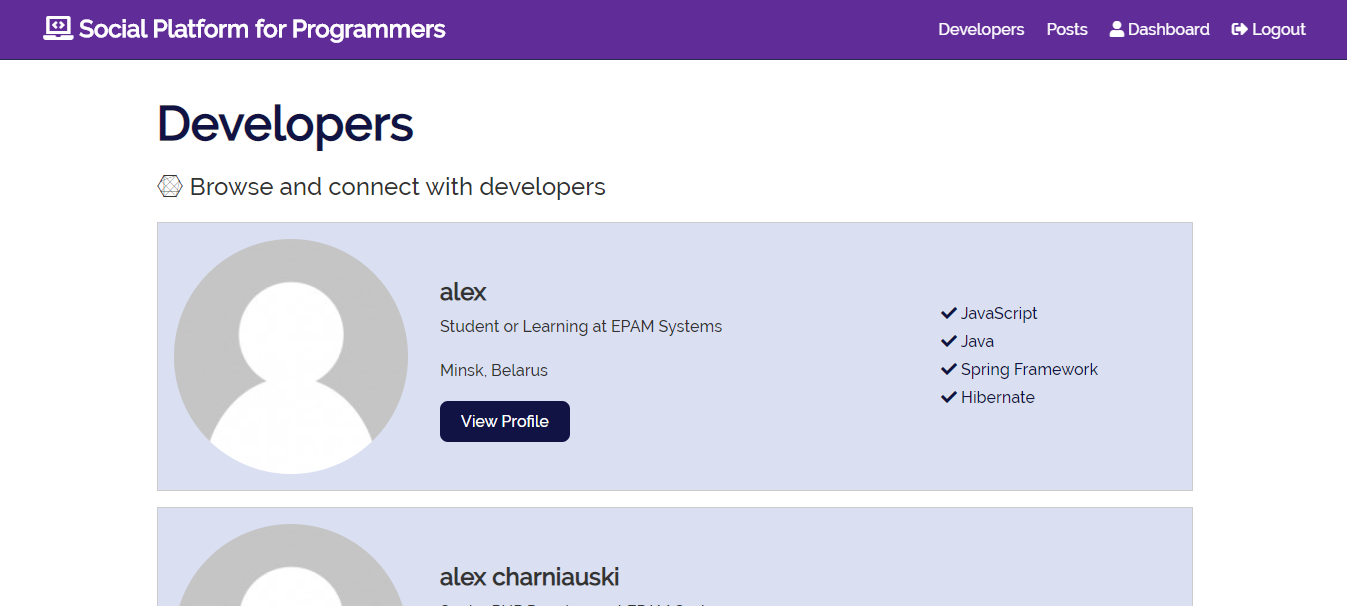


Рисунок 4.13 – Страница с разработчиками

На плитках отображается краткая информация о пользователях (статус, расположение, навыки). Есть кнопка, при нажатии на которую можно перейти на страницу профиля конкретного пользователя.

На рисунке 4.14 можно увидеть шапку страницы пользователя, в которой отображается аватар, имя, статус, расположение и указанные ссылки на социальные сети.

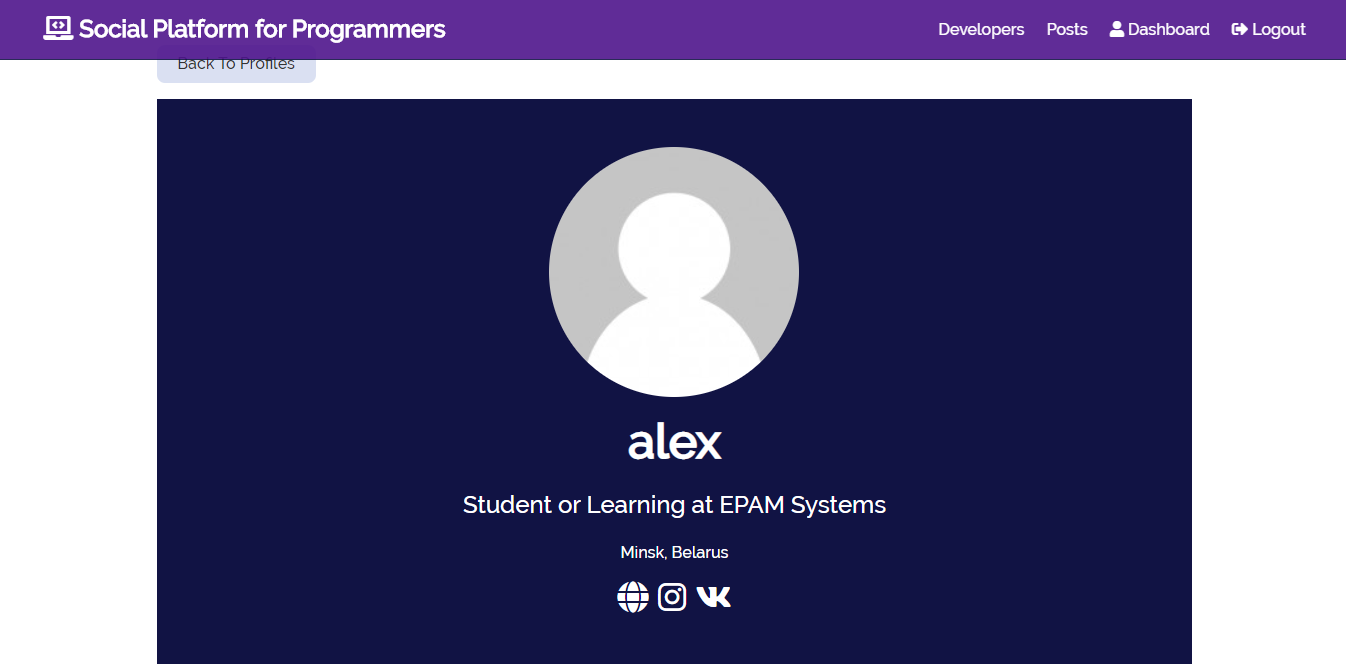


Рисунок 4.14 – Шапка страницы пользователя

На рисунке 4.15 можно увидеть продолжение страницы пользователя. Отображается биография, список навыков. Слева разделены опыт работы, справа информация об образовании.

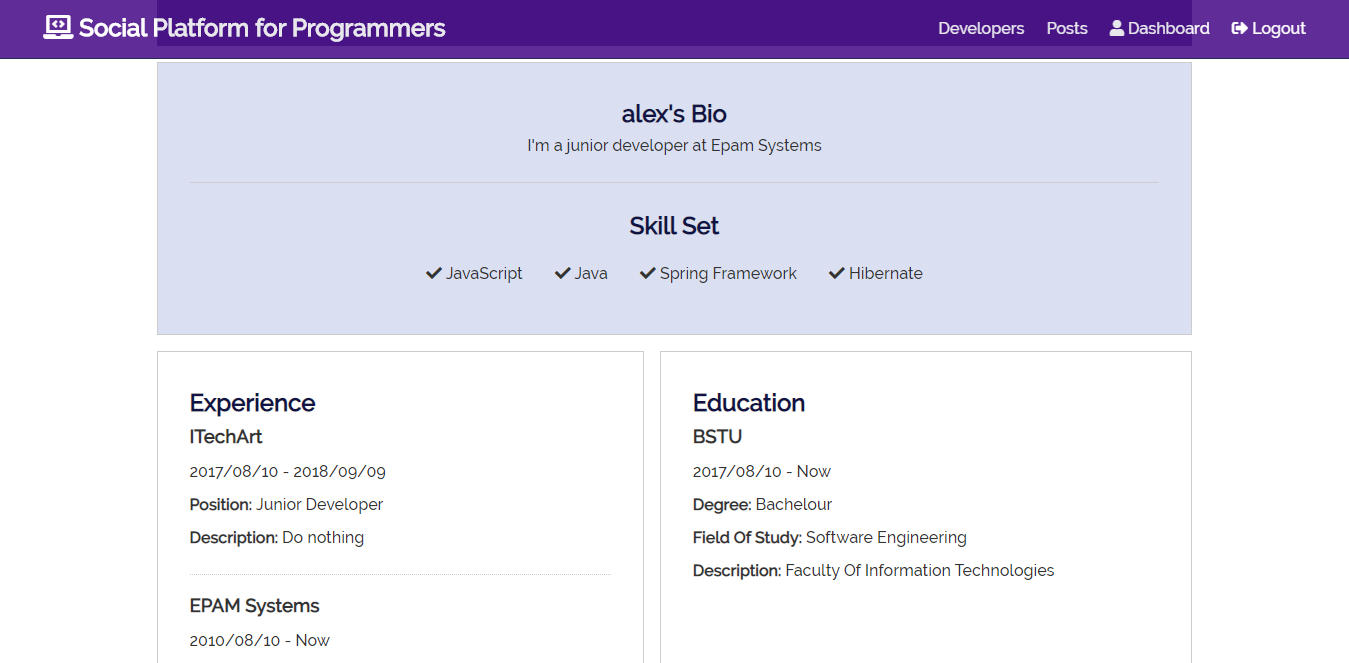


Рисунок 4.15 –Продолжение страницы пользователя

В самом низу страницы расположен список репозиториев, подтянутый с Github (Рис. 4.16).

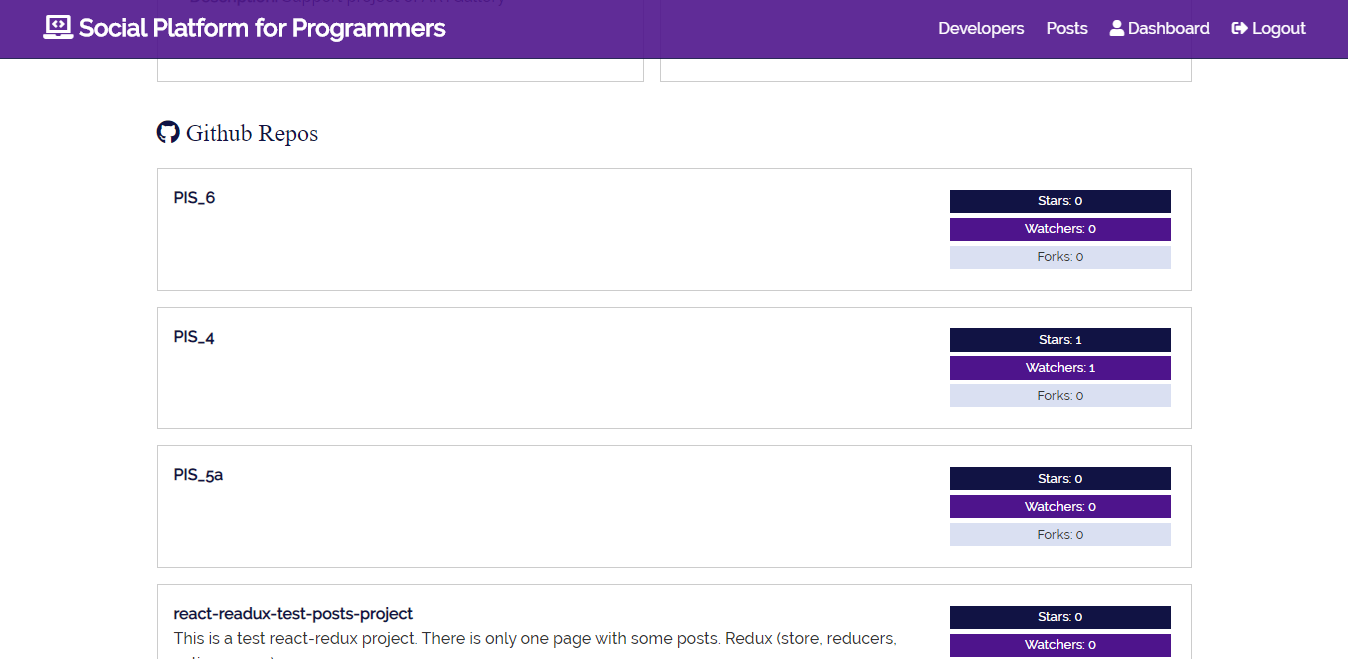


Рисунок 4.16 – Список Github репозиториев на странице пользователя

При нажатии на название репозитория можно перейти на Github страницу этого репозитория.

Теперь перейдем на страницу постов кликнув в шапке сайта на кнопку Posts (Рис. 4.17).

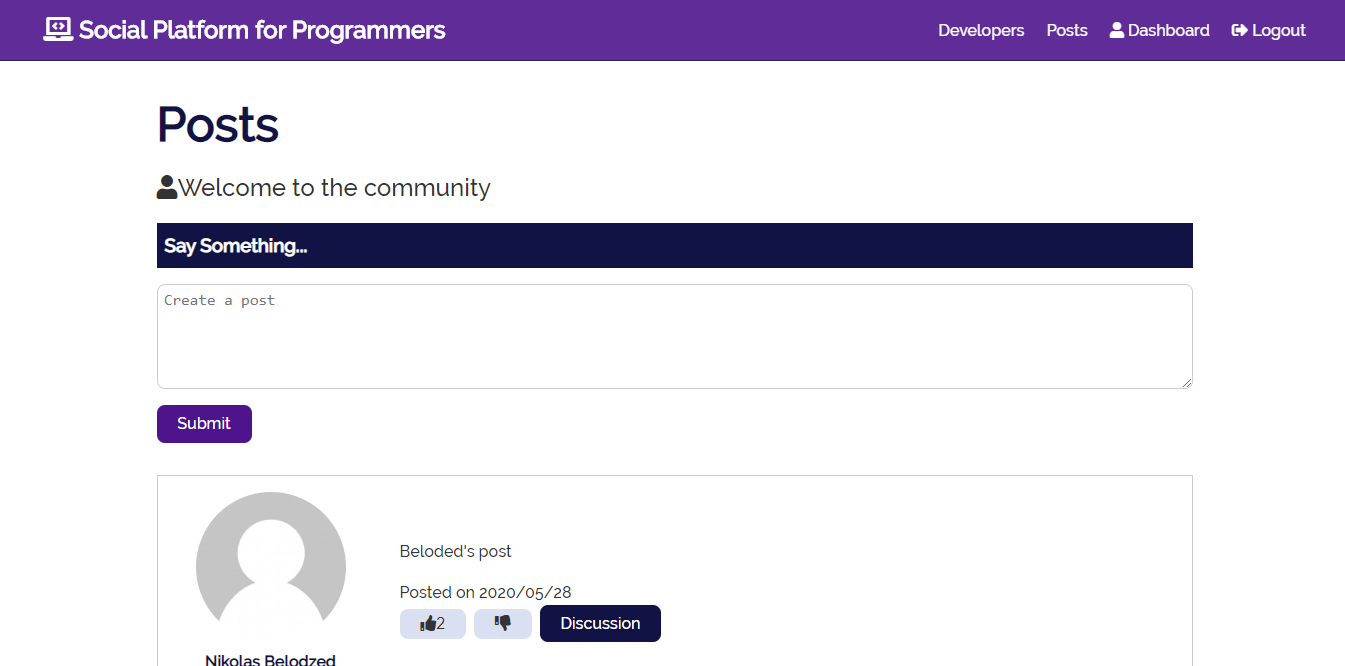


Рисунок 4.17 – Страница постов

На странице можно видеть поле для создание нового поста. При нажатии кнопки Submit пост создастся. При создании поста также записывается время создания. Это можно видеть на рисунке 4.18.

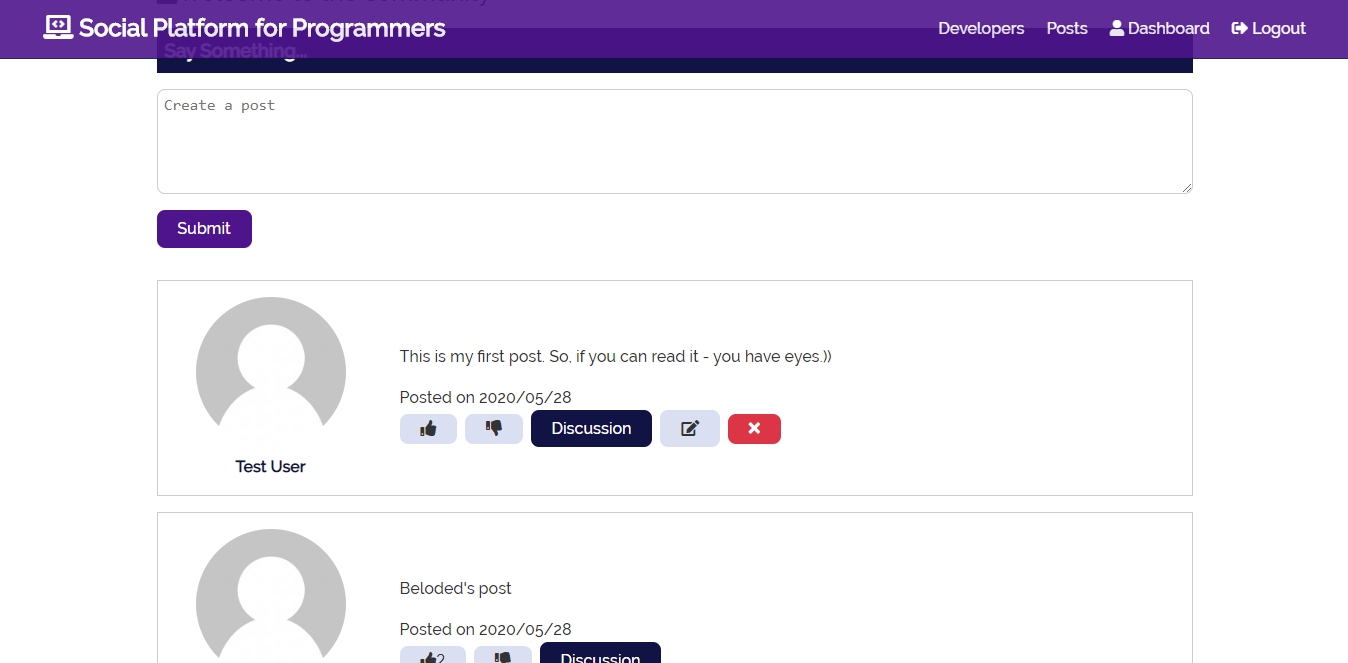


Рисунок 4.18 – Созданный пользователем пост добавлен

Пользователь может редактировать и удалять только свои посты. Для этого на плитках отображаются кнопка редактирования и удаления. Также пользователь может лайкать любой пост, либо убирать лайк. Для этого есть кнопки «палец вверх» и «палец вниз». Есть возможность лайкнуть только один раз.

Для добавления комментария нажмем на кнопку Discussion. Система перенаправит нас на страницу выбранного поста (Рис. 4.19).

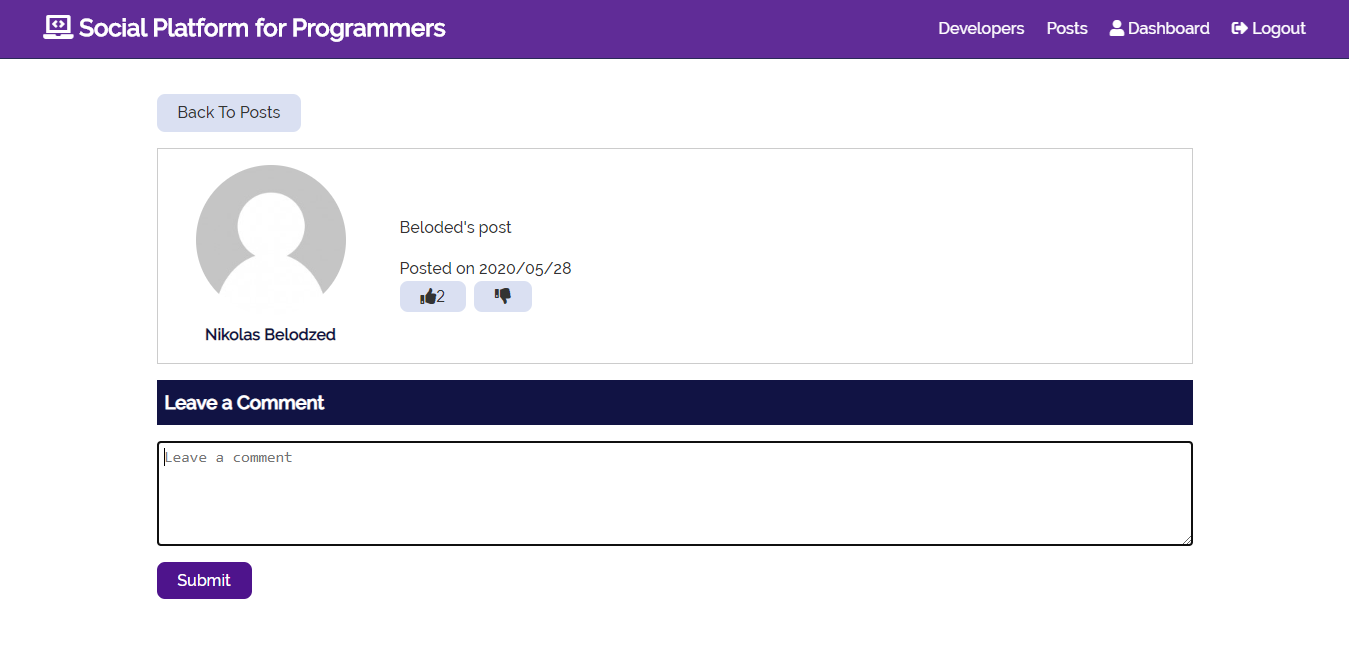


Рисунок 4.19 – Страница поста

На странице видна вся информация о посте, а также есть возможность добавить комментарий. Поле для ввода комментария расположено снизу. Добавим комментарий, нажав на кнопку Submit. И лайкнем пост. Результат представлен на рисунке 4.20.

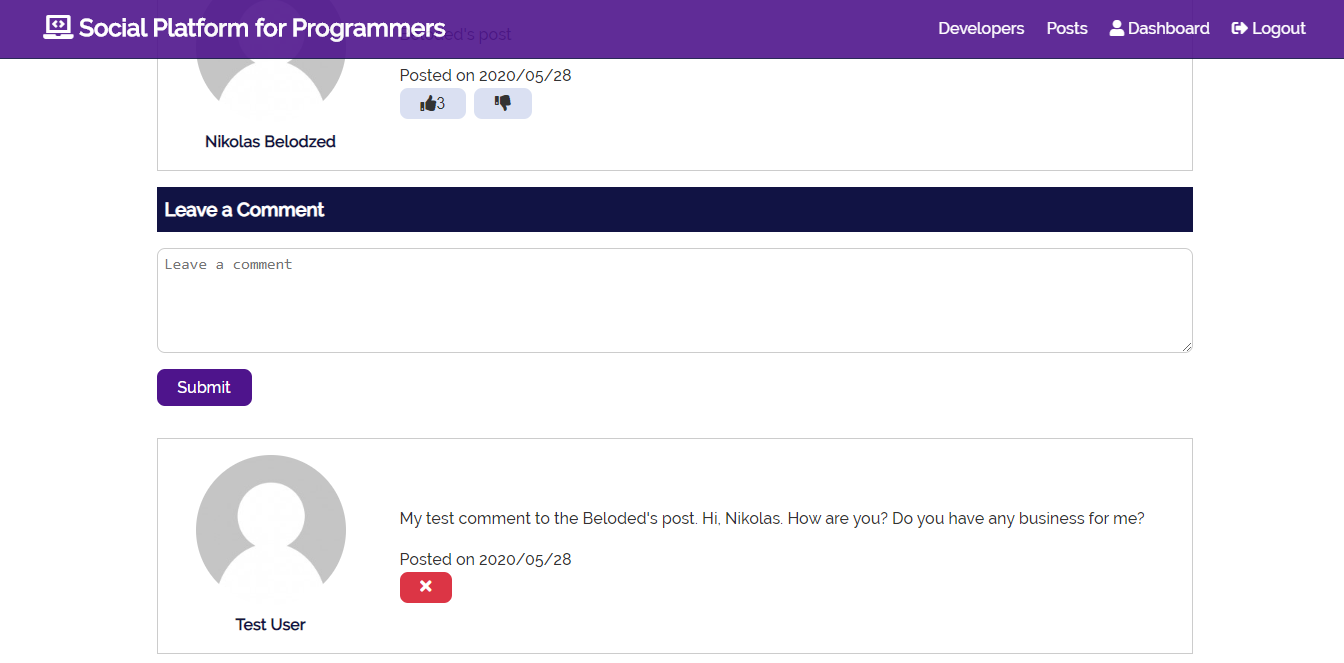


Рисунок 4.20 – Результаты добавления комментария и лайка к посту

Можно увидеть, что комментарий добавился, его можно удалить, нажав на красную кнопку крестика.

Следует отметить, что при успешных или неуспешных действиях пользователя вверху страницы показываются уведомления. Их цвет описывает наличие ошибки или успешность (красный, зеленый). Пример уведомления при создании поста можно увидеть на рисунке 4.21

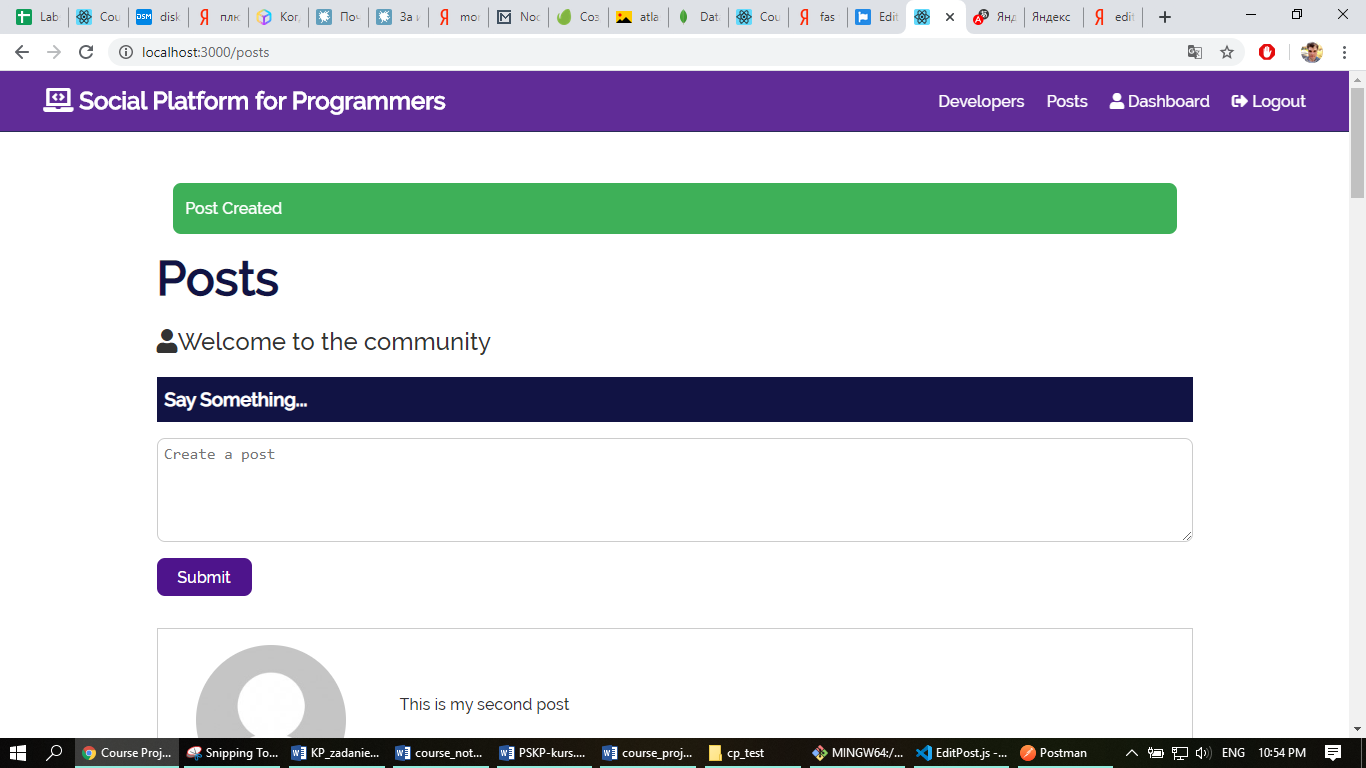


Рисунок 4.21 – Пример уведомления при успешном создании поста

Также существуют уведомления со стороны других пользователей (лайк, комментарий). Результат написания комментария на пост текущего пользователя можно видеть на рисунке 4.22

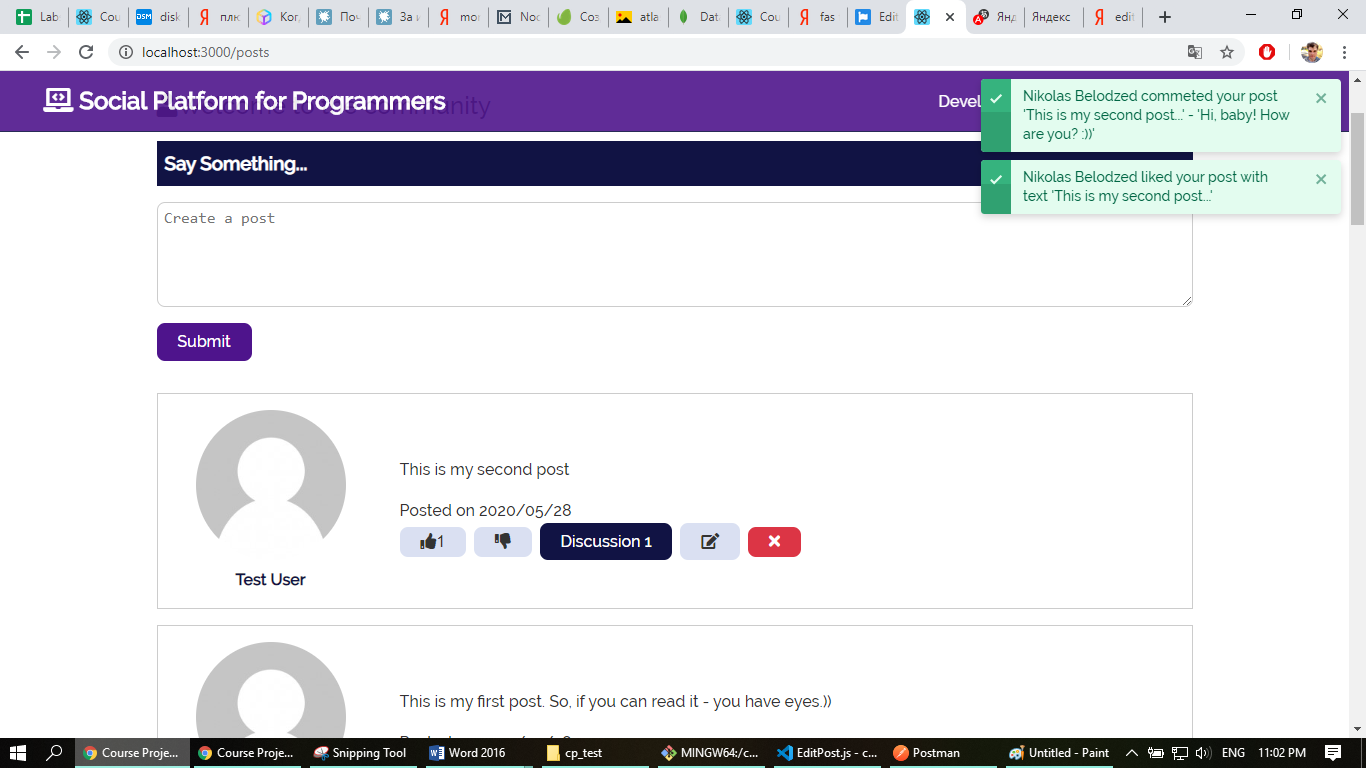


Рисунок 4.22 – Уведомление о действиях от другого пользователя

Видно, что возле слова Discussion появилась единица – это означает, что у поста один комментарий.

При нажатии на кнопку редактирования система перенаправляет пользователя на страницу редактирования поста (Рис. 4.23).

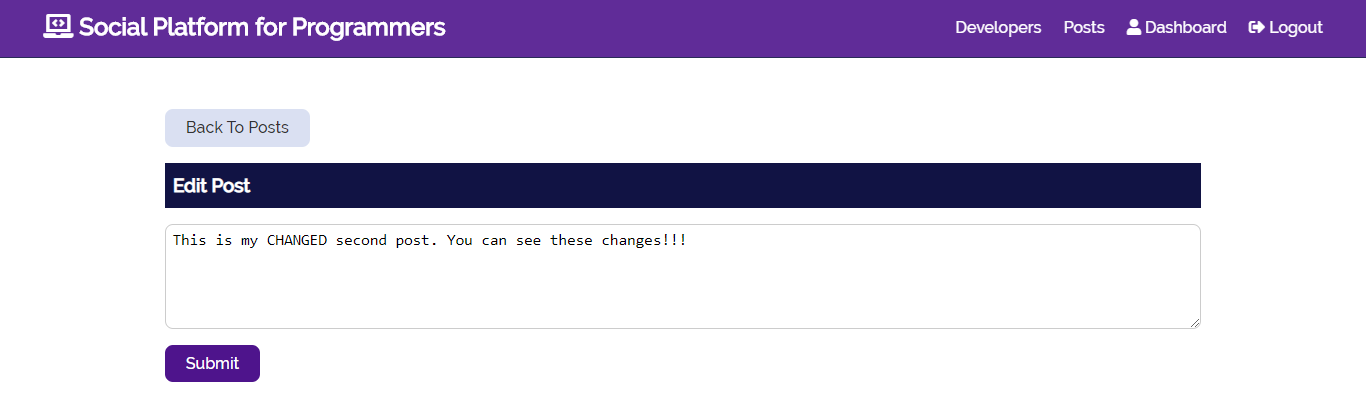


Рисунок 4.23 – Страница редактирования поста

После нажатия пользователем на кнопку Submit система перенаправляет на страницу поста и уведомляет об успешном изменении данных (Рис. 4.24).

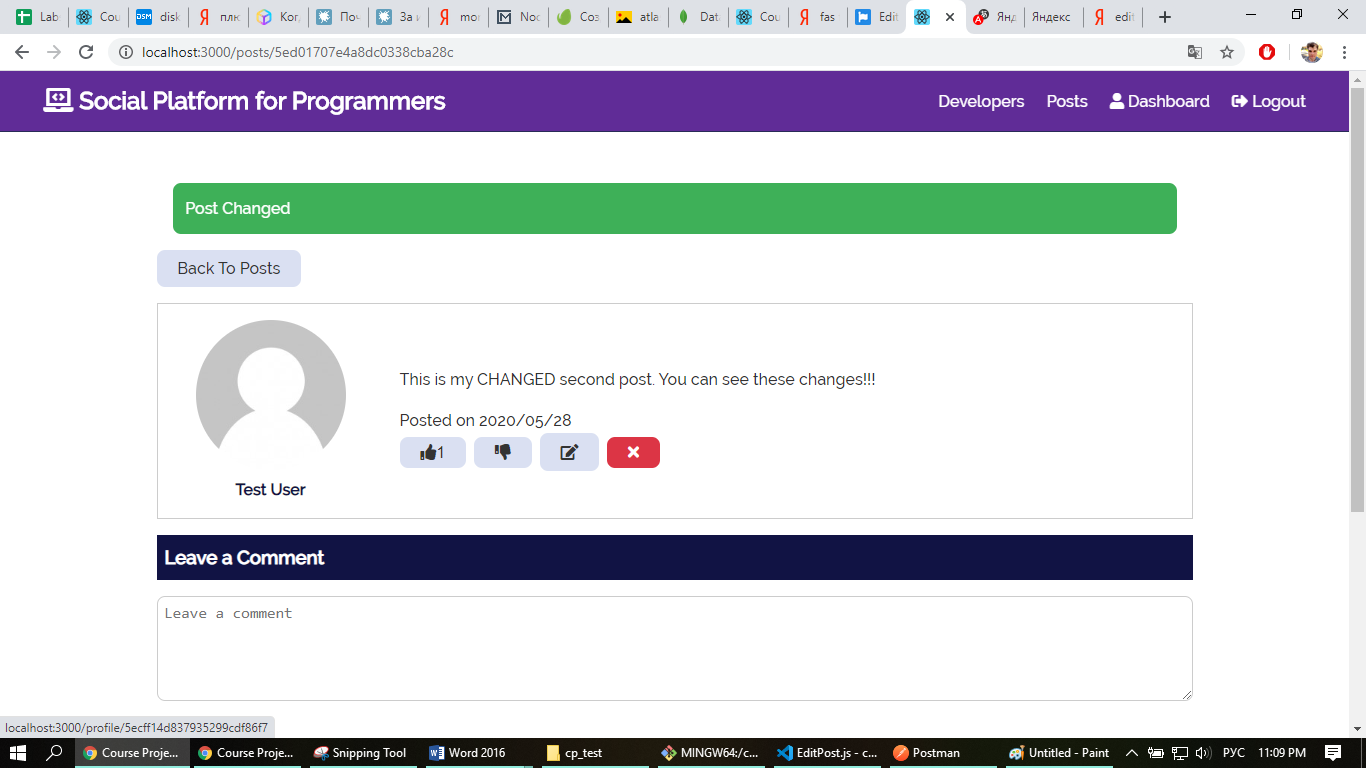


Рисунок 4.24 – Страница поста после изменения

Если удалить пост, то удаляются и комментарии, оставленные к нему. Попробуем это сделать. Для этого вновь перейдем на страницу постов и нажмем красную кнопку. Результат можно увидеть на рисунке 4.25.

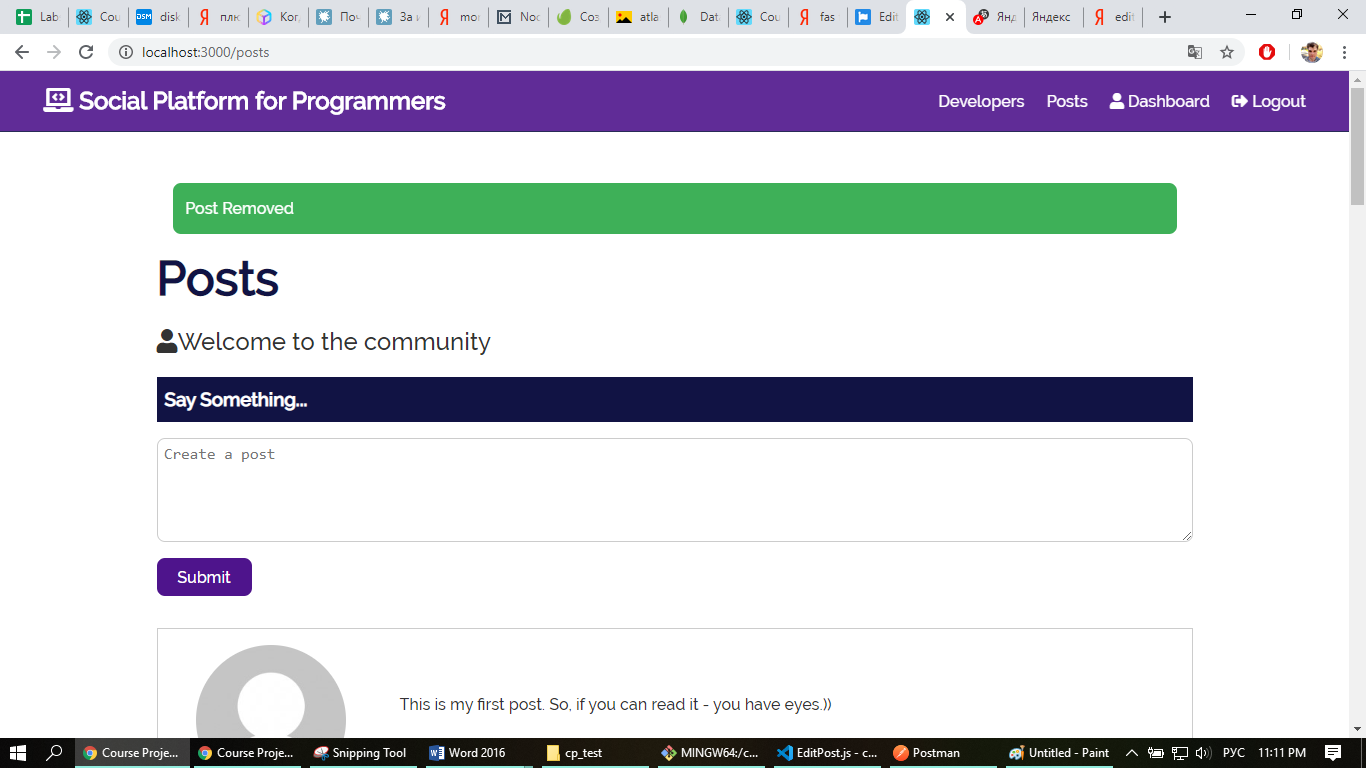


Рисунок 4.25 – Результат удаления поста

Тестирование

ФИО

Подпись

Дата

Лист

1

КП 05.00.ПЗ

Разраб.

Чернявский А.Л.

Провер.

Бурмакова А.В.

Н. контр.

Бурмакова А.В.

Утверд.

Бурмакова А.В.

5 Тестирование

Лит.

Листов

11

БГТУ 71171014, 2020

В этом разделе будут протестированы компоненты системы, которых потенциально быть введены некорректные данные. Такие компоненты как авторизация, регистрация, добавление данных и др.

При регистрации должны быть заполнены все поля (Рис. 5.1).

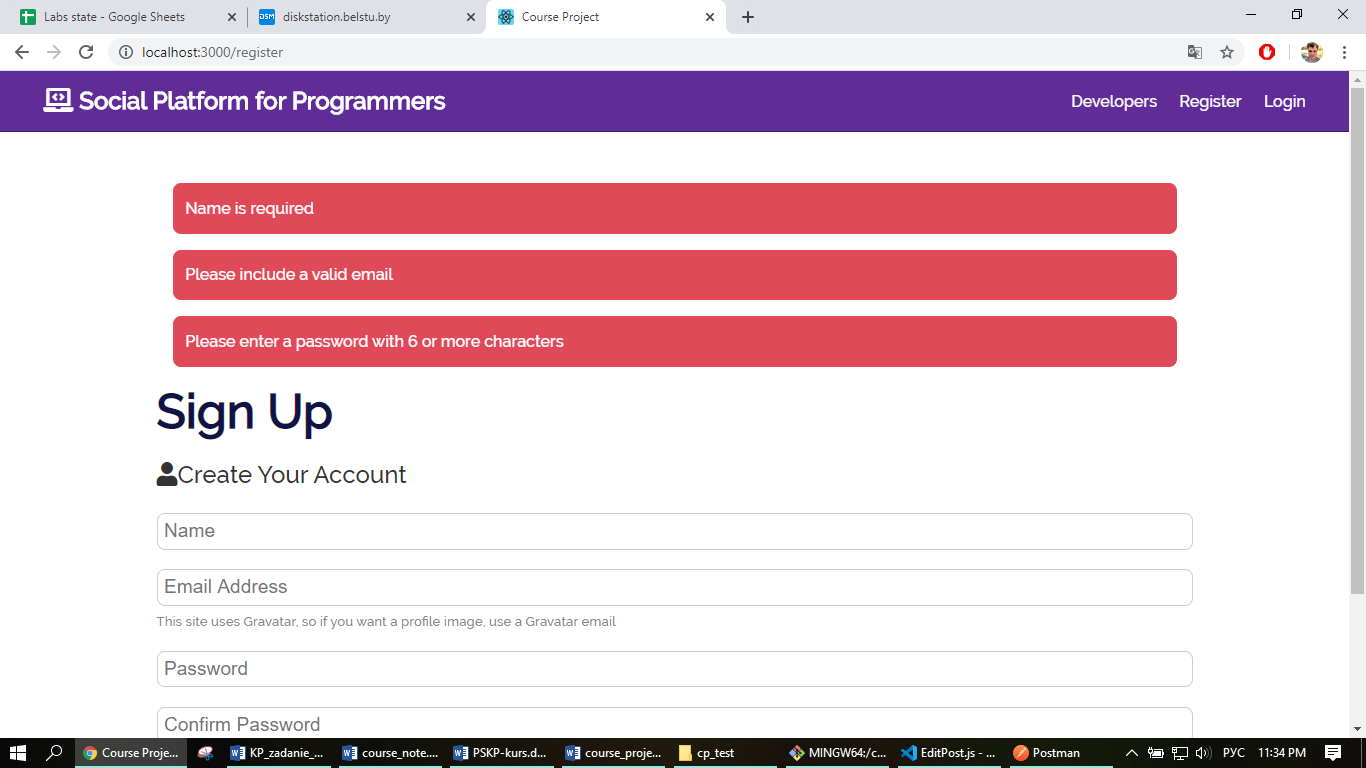


Рисунок 5.1 – Проверка ограничений в форме регистрации

Также при регистрации пароли должны совпадать. Уведомление об этом можно увидеть на рисунке 5.2.

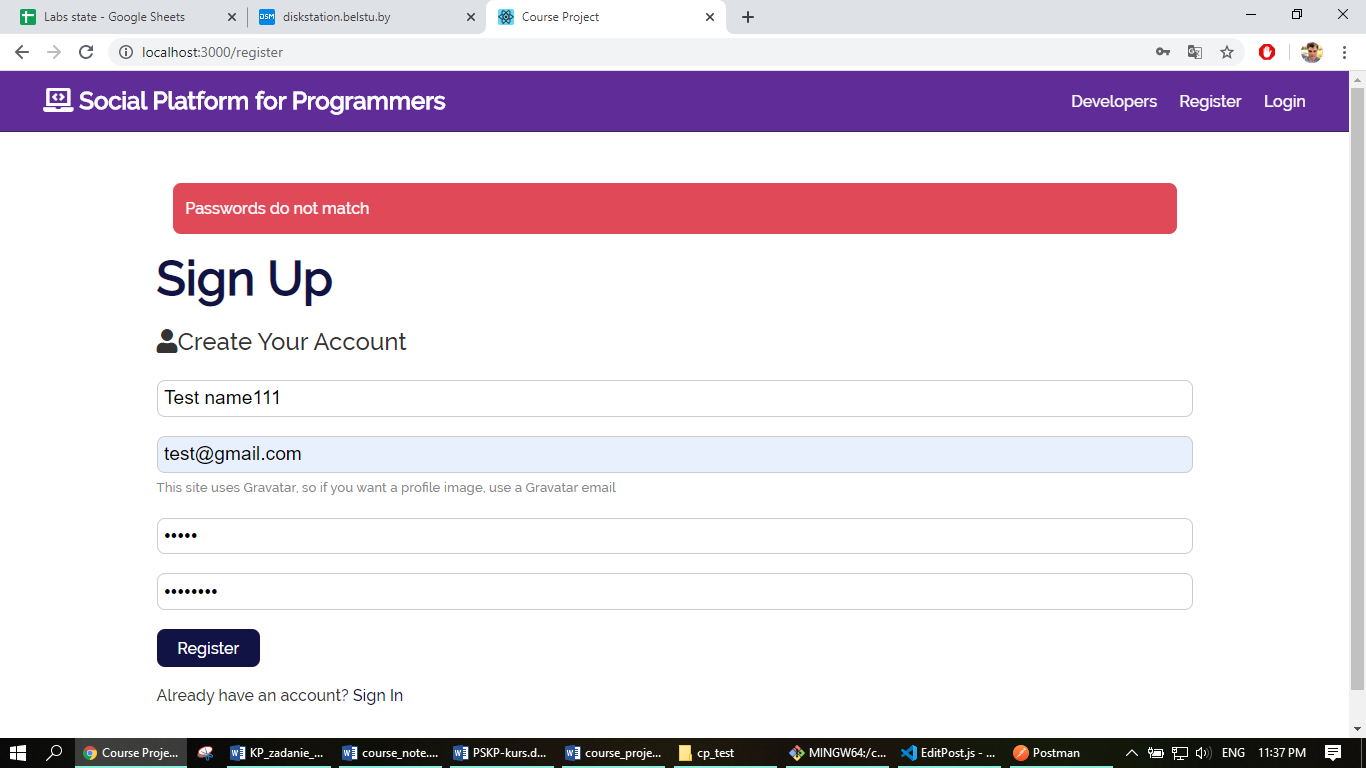


Рисунок 5.2 – Проверка на совпадение паролей

При добавлении опыта работы также есть ограничения. Результат можно увидеть на рисунке 5.3

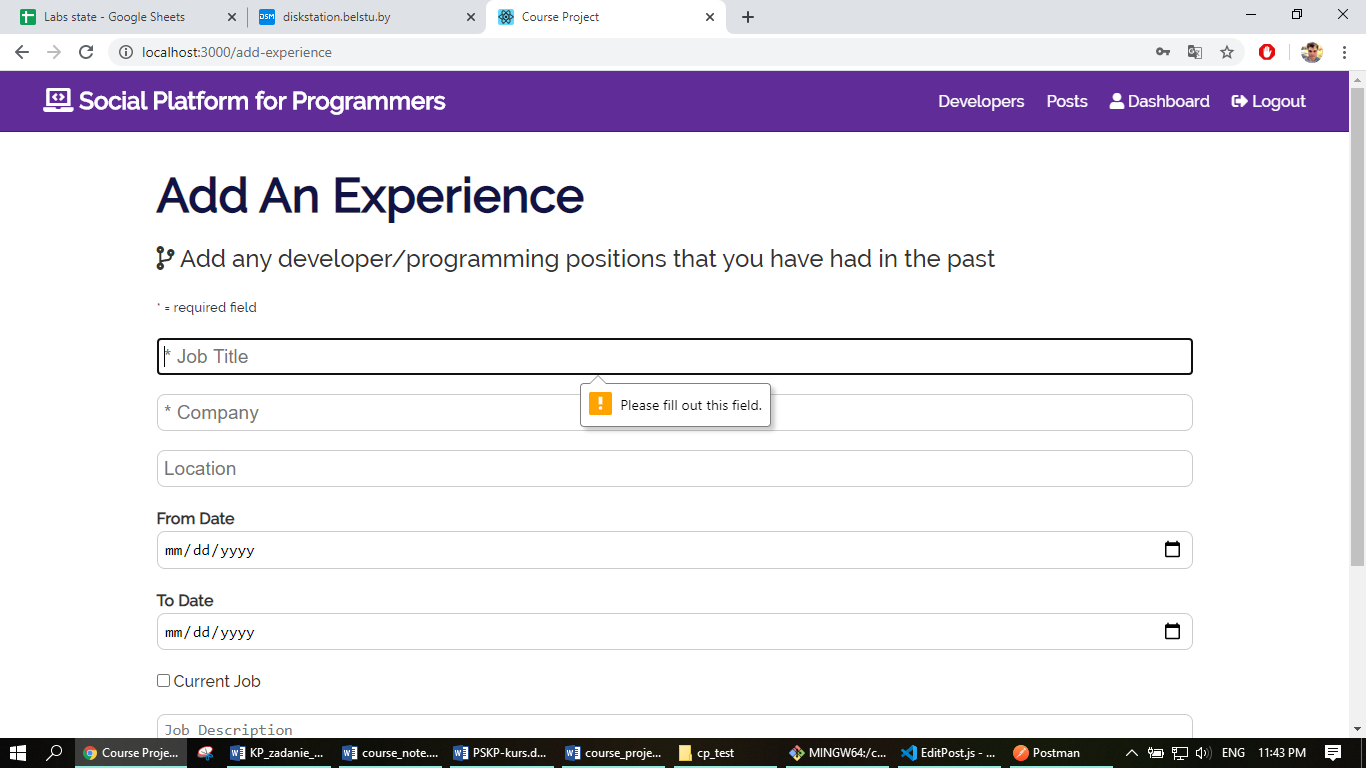


Рисунок 5.3 – Обязательные поля при добавлении опыта

Также есть ограничение на добавление только одного лайка (Рис 5.4).

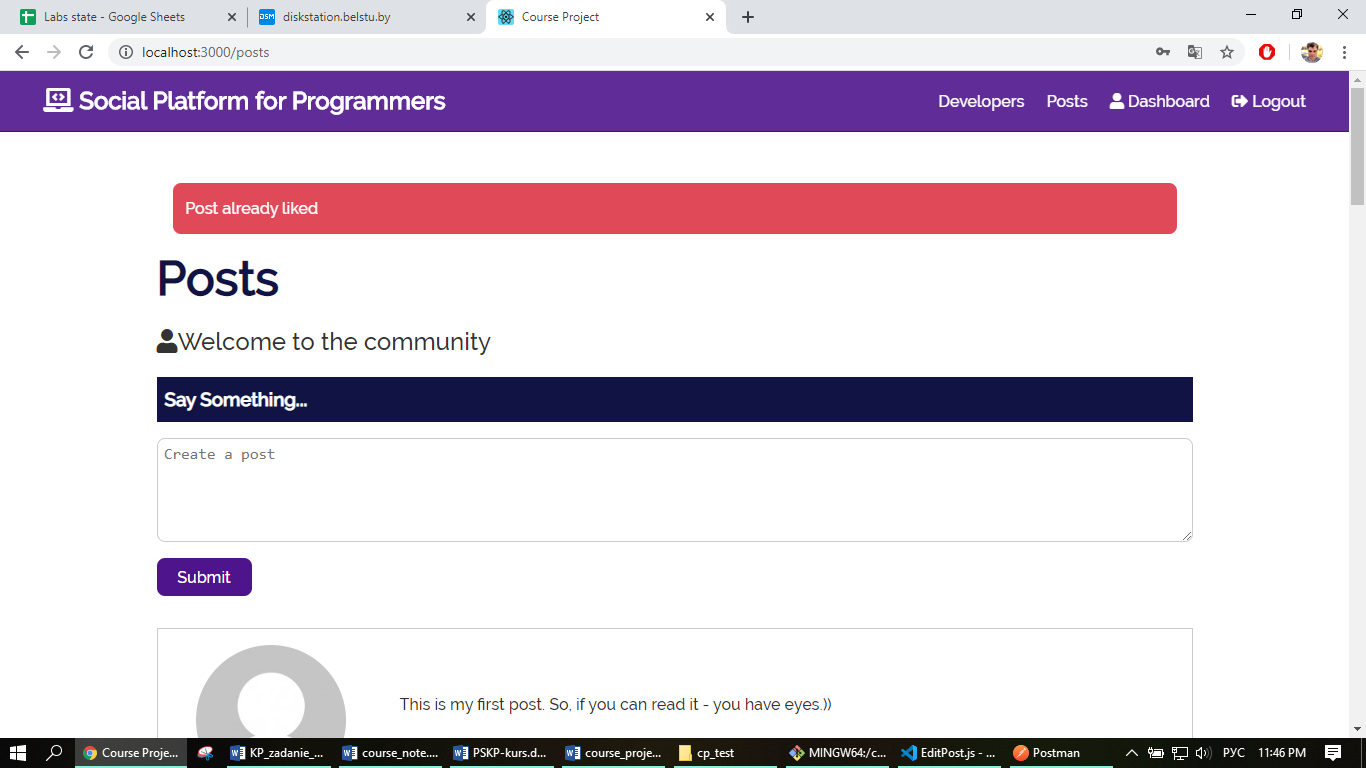


Рисунок 5.4 – Проверка на возможность лайкнуть пост один раз

Таким же образом нельзя убрать лайк, если его еще нет от текущего пользователя (Рис. 5.5).

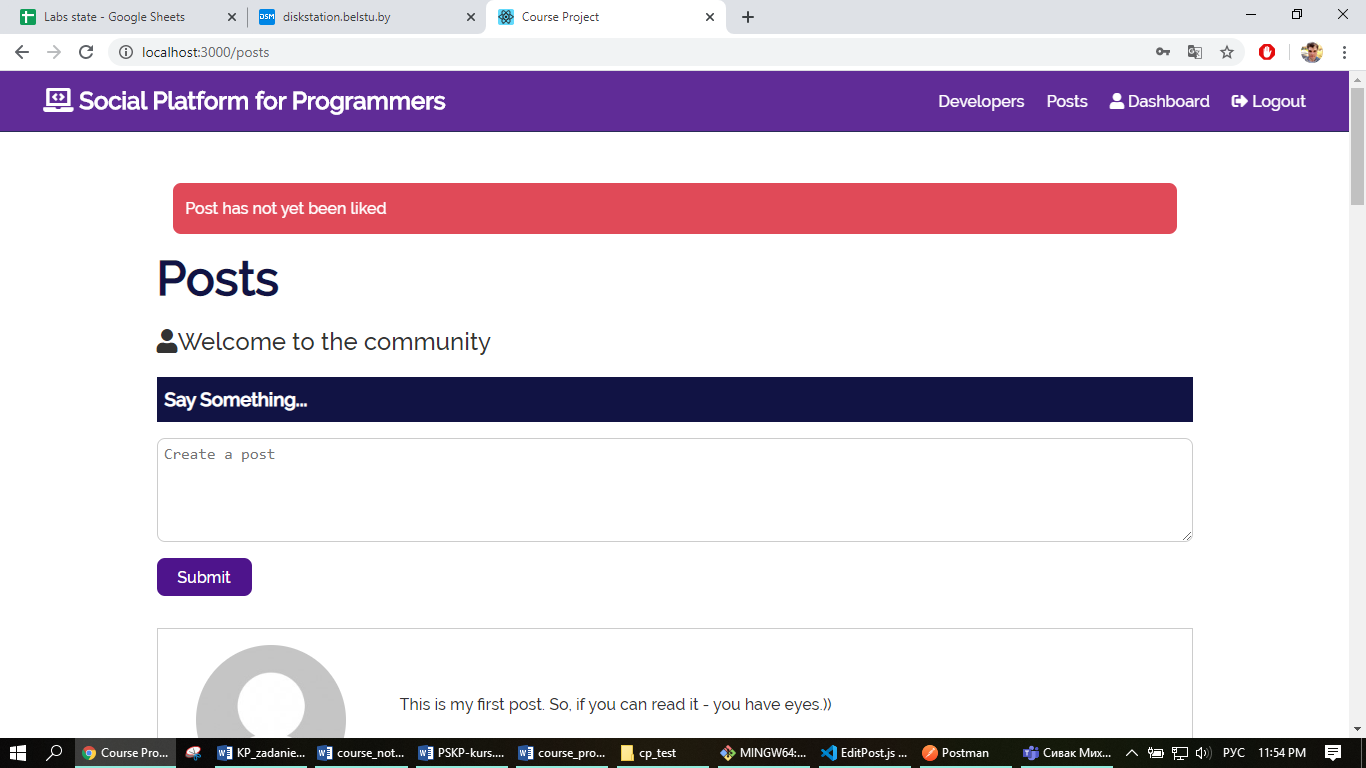


Рисунок 5.5 – Проверка на возможность убрать лайк, которого нет

Заключение

ФИО

Подпись

Дата

Лист

1

КП 00.00.ПЗ

Разраб.

Чернявский А.Л.

Провер.

Бурмакова А.В.

Н. контр.

Бурмакова А.В.

Утверд.

Бурмакова А.В.

Заключение

Лит.

Листов

1

БГТУ 71171014, 2020

В процессе решения поставленной задачи была достигнута поставленная цель по созданию программного средства «Социальная платформа для программистов». Основой целью курсового проекта стало проектирование кроссплатформенного приложение, которое помогло облегчить взаимодействие с пользователем. Это было достигнуто за счёт гибкой программной платформы сервера на Node.js и адаптивности сайта, в качестве клиентской части.

Отображение, бизнес логика и хранилище данных максимально независимы друг от друга для возможности расширения.

При составлении руководства пользователя была подробно описана работа с программным средством. Также было проведено тестирование.

В соответствии с полученным результатом работы программы можно сделать вывод, что разработанная программа работает верно, а требования технического задания выполнены в полном объёме.

Список литературы

ФИО

Подпись

Дата

Лист

1

КП 00.00.ПЗ

Разраб.

Чернявский А.Л.

Провер.

Бурмакова А.В.

Н. контр.

Бурмакова А.В.

Утверд.

Бурмакова А.В.

Список литературы

Лит.

Листов

1

БГТУ 71171014, 2020

1. Express – фреймворк веб-приложений Node.js [Электронный ресурс] – Режим доступа: <https://expressjs.com/ru/>. – Дата доступа: 10.04.2020.
2. React – A JavaScript library for building user interfaces [Электронный ресурс] – Режим доступа: <https://reactjs.org/>. – Дата доступа: 20.04.2020.
3. Mongoose ODM [Электронный ресурс] – Режим доступа: <https://mongoosejs.com/>. – Дата доступа: 22.03.2020.
4. MongoDB Atlas [Электронный ресурс] – Режим доступа: <https://docs.atlas.mongodb.com/>. – Дата доступа: 22.03.2020.
5. Request – npm [Электронный ресурс] – Режим доступа: <https://www.npmjs.com/package/request>. – Дата доступа: 22.04.2020.
6. Socket.IO – Docs [Электронный ресурс] – Режим доступа: <https://socket.io/docs>. – Дата доступа: 24.04.2020.
7. Redux – A predictable state container for JavaScript apps [Электронный ресурс] – Режим доступа: <https://redux.js.org/>. – Дата доступа: 24.04.2020.
8. JSON Web Tokens [Электронный ресурс] – Режим доступа: <https://jwt.io/>. – Дата доступа: 28.04.2020.

Приложение А

|  |
| --- |
| const jwt = require('jsonwebtoken');  const config = require('config');  module.exports = (req, res, next) => {    // Get token from header    const token = req.header('x-auth-token');    // Check if not token    if (!token) {      return res.status(401).json({        msg: 'No token, authorization denied'      });    }    // Verify token    try {      const decoded = jwt.verify(token, config.get('jwtSecret'));      req.user = decoded.user;      next();    } catch (err) {      res.status(401).json({ msg: 'Token is not valid' });    }  }; |

Приложение Б

|  |
| --- |
| const UserSchema = new mongoose.Schema({      name: {          type: String,          required: true      },      email: {          type: String,          required: true,          unique: true      },      password: {          type: String,          required: true      },      avatar: {          type: String      },      date: {          type: Date,          default: Date.now      }  })  const ProfileSchema = new mongoose.Schema({      user: {          type: mongoose.Schema.Types.ObjectId,          ref: 'user'      },      company: {          type: String      },      website: {          type: String      },      location: {          type: String    // city      },      status: {          type: String,   // developer, junior, senior          required: true      },      skills: {          type: [String], // JS, Java, php, ...          required: true      },      bio: {          type: String      },      githubusername: {          type: String      },      experience: [          {              title: {                  type: String,                  required: true              },              company: {                  type: String,                  required: true              },              location: {                  type: String              },              from: {                  type: Date,                  required: true              },              to: {                  type: Date              },              current: {                  type: Boolean,                  default: false              },              description: {                  type: String              }          }      ],      education: [          {              school: {                  type: String,                  required: true              },              degree: {                  type: String,                  required: true              },              fieldofstudy: {                  type: String,                  required: true              },              from: {                  type: Date,                  required: true              },              to: {                  type: Date              },              current: {                  type: Boolean,                  default: false              },              description: {                  type: String              }          }      ],      social: {          youtube: {              type: String          },          twitter: {              type: String          },          facebook: {              type: String          },          linkedin: {              type: String          },          instagram: {              type: String          },          vk: {              type: String          }      },      date: {          type: Date,          default: Date.now      }  })  const PostSchema = new Schema({      user: {          type: Schema.Types.ObjectId,          ref: 'users'      },      text: {          type: String,          required: true      },      name: {          type: String      },      avatar: { // if user delete account - not delete post          type: String      },      likes: [          {              user: {                  type: Schema.Types.ObjectId,                  ref: 'users'              }          }      ],      comments: [          {              user: {                  type: Schema.Types.ObjectId,                  ref: 'users'              },              text: {                  type: String,                  required: true              },              name: {                  type: String              },              avatar: { // if user delete account - not delete post                  type: String              },              date: {                  type: Date,                  default: Date.now              }          }      ],      date: {          type: Date,          default: Date.now      }  }) |