Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего профессионального образования

«Владимирский государственный университет

имени Александра Григорьевича и Николая Григорьевича Столетовых»

(ВлГУ)

Кафедра информационных систем и программной инженерии

Лабораторная работа №1

по дисциплине "Технологии разработки веб приложений"

ТЕМА РАБОТЫ:

Разработка клиентских приложений

Выполнил:

студент гр. ПРИ-120

Парахин К.В.

Принял:

преподаватель кафедры ИСПИ

Шамышев А.А

Владимир 2024 г

Цель работы:

Изучить различные библиотеки и фреймворки создания клиентской части веб-приложений.

Задание:

В ходе выполнения работы вам необходимо 3 раза с использованием различных технологий создать клиентскую часть веб приложения.

Требования к клиентской части:

1. Целостность (интерфейс должен предназначаться для выполнения какой-либо одной функции);
2. Наличие минимум двух страниц с различными URL.
3. Наличие дизайн-концепции веб интерфейса (нельзя просто прикрутить стандартные кнопочки и успокоиться!);
4. Адаптивный веб-дизайн (правильное отображение на мобильных устройствах);
5. Наличие форм ввода данных. Обработка форм ввода данных.
6. Наличие изображений (картинок);

Выполнение работы:

Были выбраны 3 технологии: React, Flutter и Vue.js

Тематика веб-приложения: приложение для абитуриентов ВУЗа. Представляет собой интерфейс, позволяющий новым абитуриентам регистрироваться/авторизоваться в системе, с помощью администратора (отдельная роль в системе) – добавлять список направлений на поступление, с указанием приоритета, а также отслеживать наличие оригинала диплома. Каждое направление также добавляется администраторов, по умолчанию, для каждого абитуриента в списке проставляется нулевой балл. По получению результатов письменных испытаний, администратор может загрузить .csv файл с оценками – и они обновят старые значения по определенному направлению. Каждое направление имеет свои настройки, на основе которых по финализации результатов – рассчитываются списки поступивших  
Свои результаты абитуриенты могут отслеживать в своем личном кабинете, все действия по изменению их контента производят администраторы в своих личных кабинетах, соответственно.

До разработки клиентских приложений было разработано серверное приложение с REST API, при открытых CORS заголовках для каждого из приложений фронта  
  
Список запросов можно посмотреть в рамках данного окна Postman, все данные приходят через HTTP POST запросы, авторизация происходит с помощью токена, генерирующегося на сервере при регистрации/авторизации пользователя (и сбрасывающегося после его logout действия, соответственно)

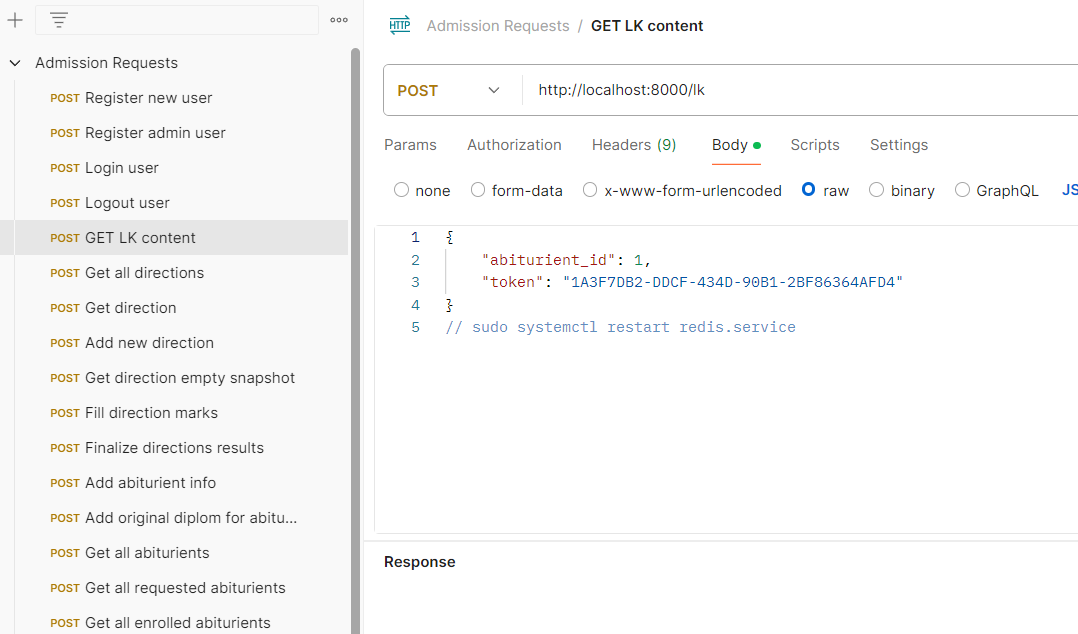


Рисунок 1. Визуализация запросов к API сервера

1. Разработка приложения на React

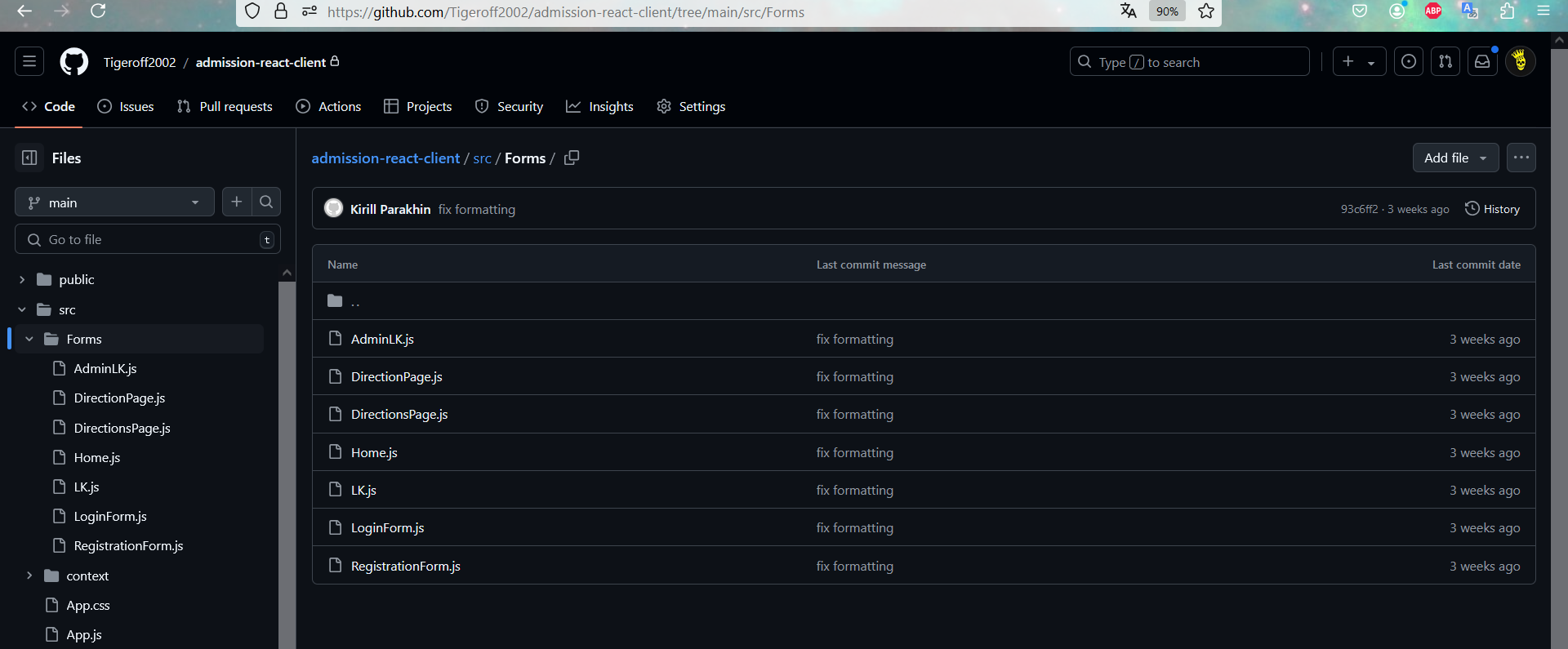


Рисунок 2.1. Репозиторий с приложением

Были разработаны следующие формы:  
Home.js – домашний экран (загружается для неавторизованного пользователя)

LoginForm.js – страница с логином

RegisterForm.js – страница с регистрацией

LK.js – страница, стартовая для аутентифицированного в системе пользователя

Стоит отметить, что была разработана система аутентификации и авторизации пользователя – она работает благодаря использованию хуков, позволяющих работать с AuthContext.js – отдельный класс – позволяющий обращаться к сессионным хранимым данным пользователя (в данном случае это токен и его идентификатор)

AdminLK.js – страница, стартовая для пользователя, имеющего роль администратора в системе

Администратор в целом обладает каким то реальным функционалом, по сравнению с обычным пользователем, который даже не может без его действий менять какую то свою личную информацию

Администратор же имеет возможность просматривать всех зарегистрированных пользователей, добавлять для них приоритеты и направления. На отдельной странице он может также создавать новые направления.

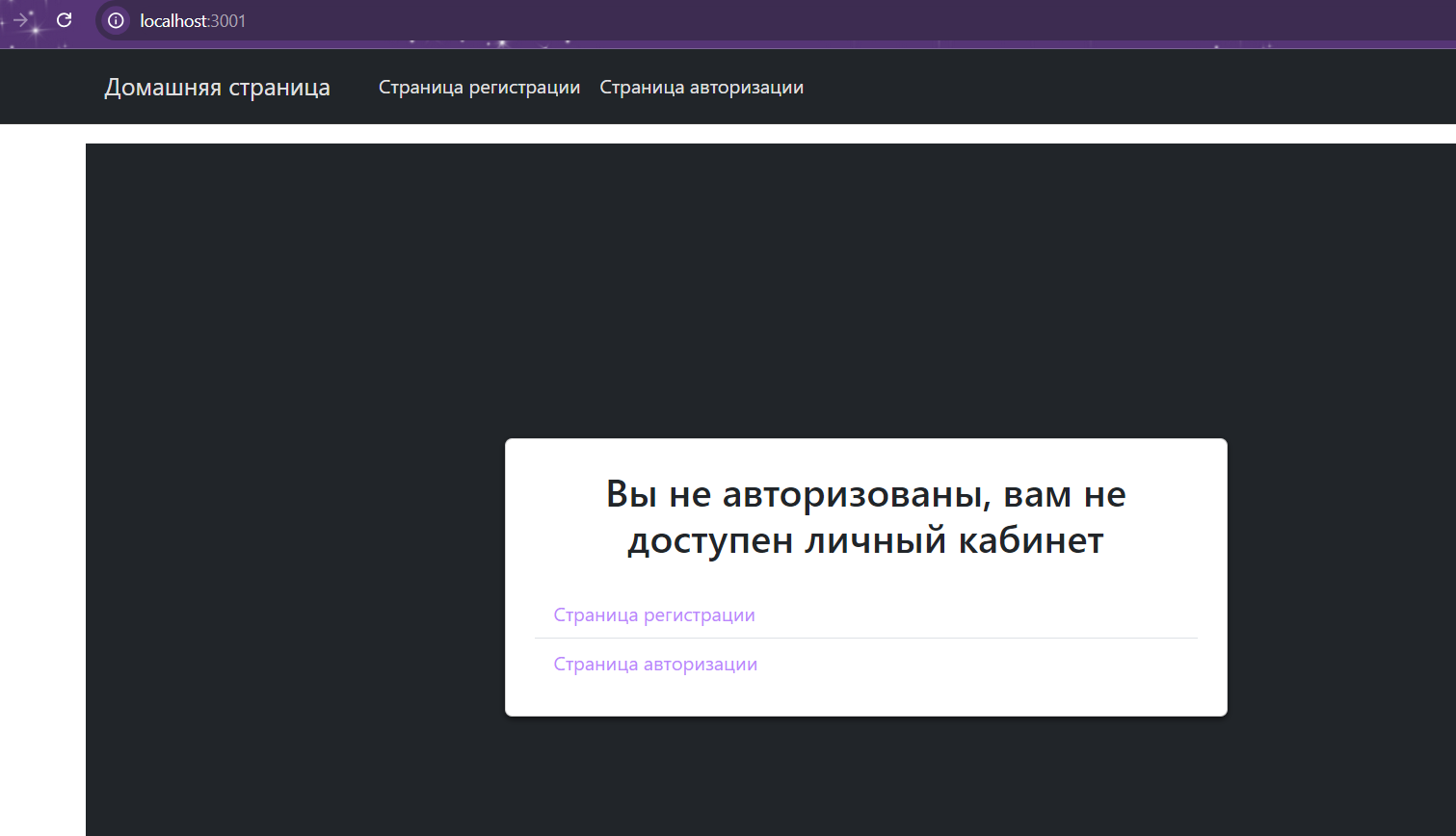
Ну и система имеет базовый листинг направлений – форма DirectionsPage.js – через который можно переходить на страницу каждого из направлений по его айдишнику – страница DirectionPage.js

На этой странице администратор может загружать файл с оценками абитуриентов и выполнять финализацию направления

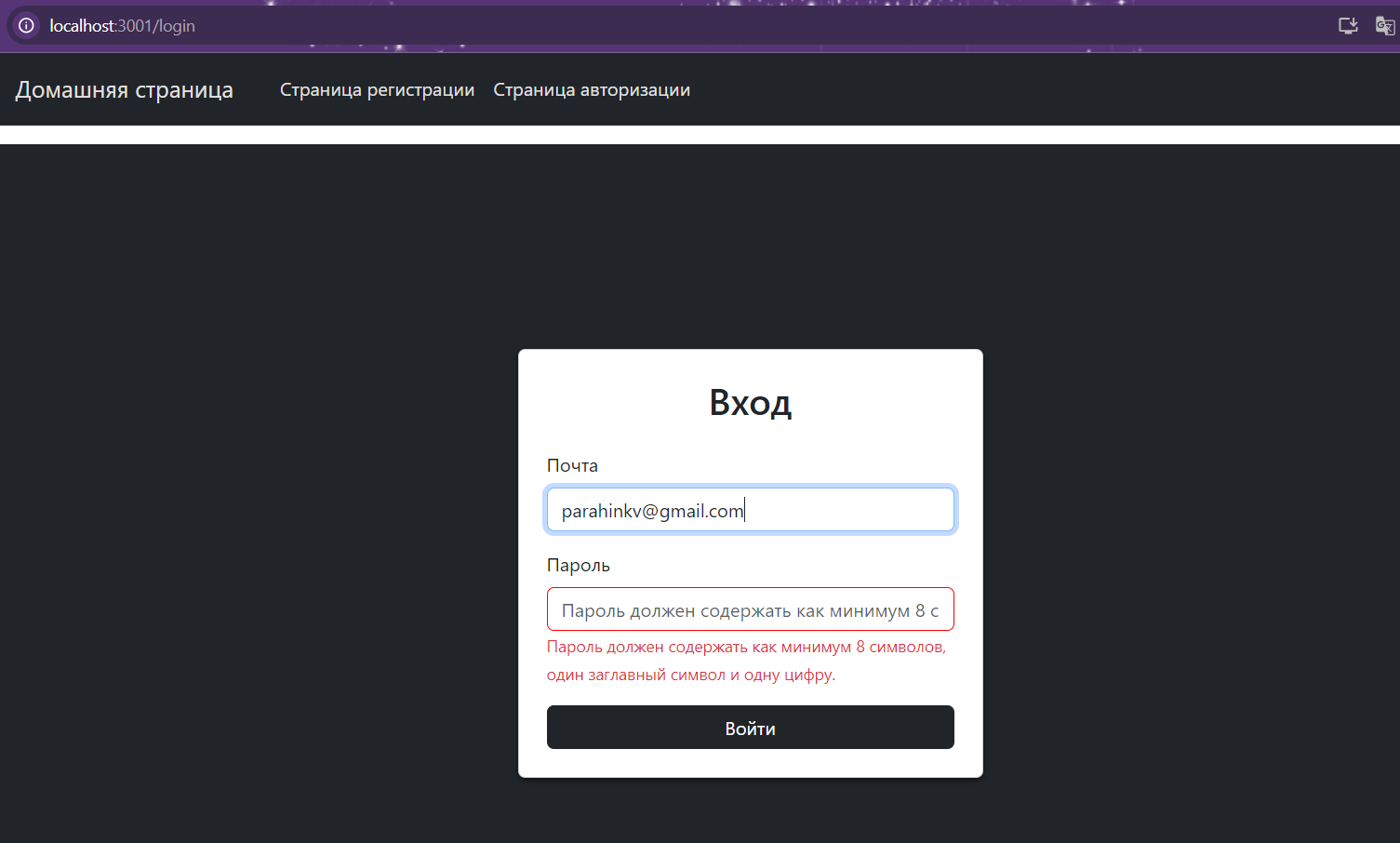
То есть система действительно направлена на решение важной бизнес функции – предоставлением для абитуриентов возможность вести учет их поступлений в направления ВУЗа.

Ниже скриншоты основных страниц приложения.Использовалась примерно одна и та же цветовая схема (с темными тонами, не сказал бы что именно темная тема). Для стилей в основном использовался Boostrap CSS. Сами странички мапятся на свои URLы при помощи React Router – его настройка происходит в файле App.js:

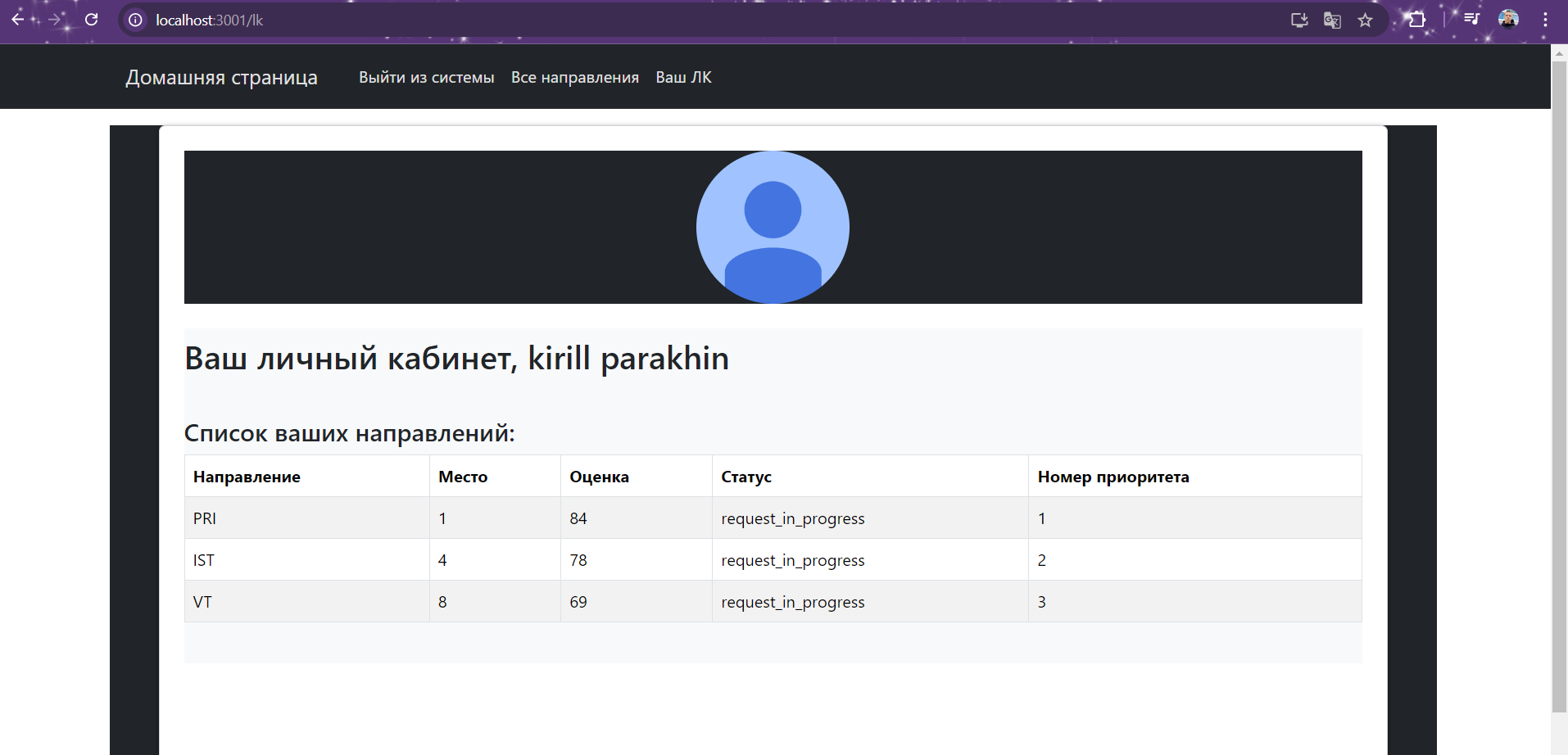
Страница стартовая:



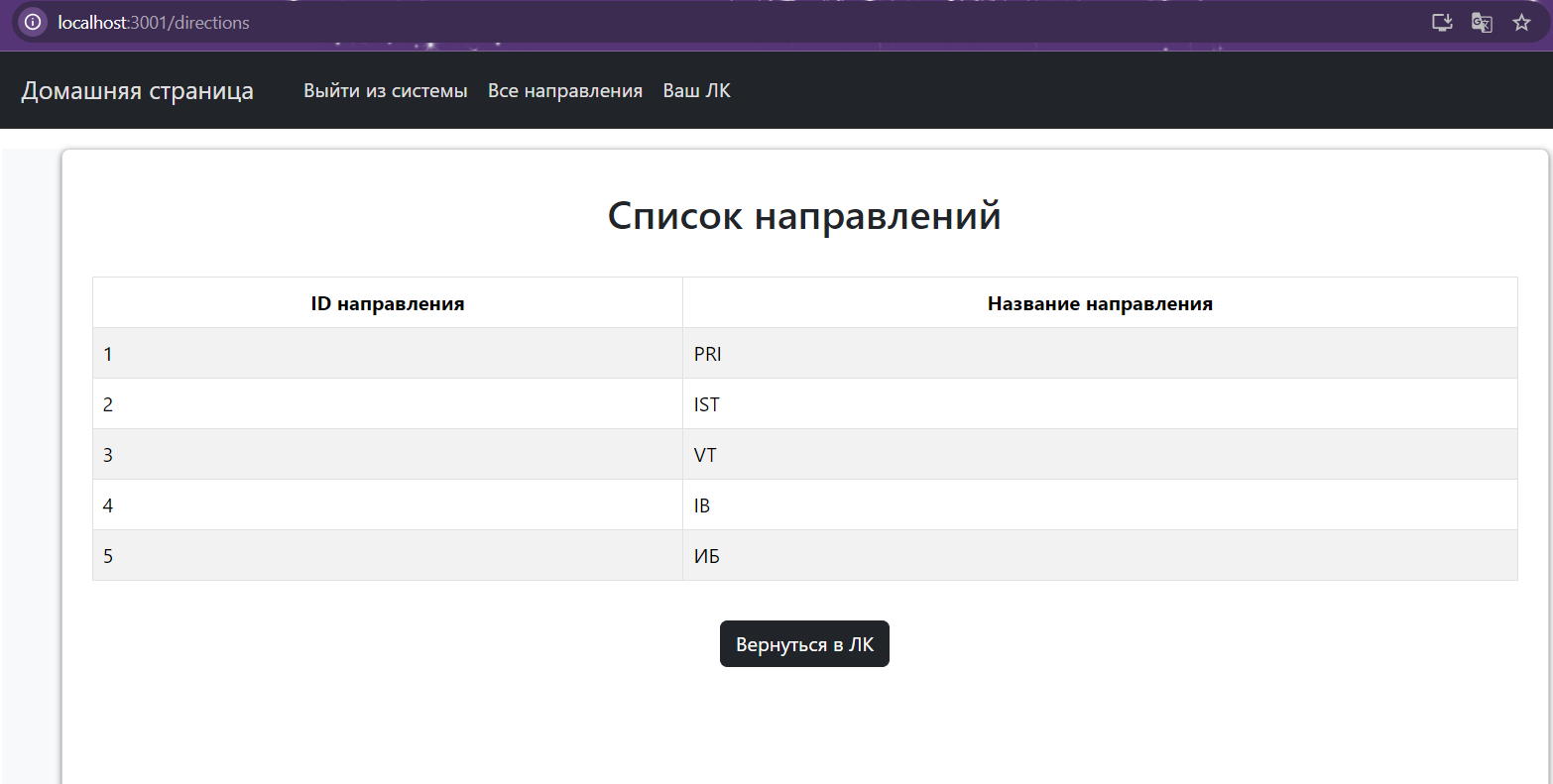
Страница авторизации (с валидацией email по regex и валидацией пароля)



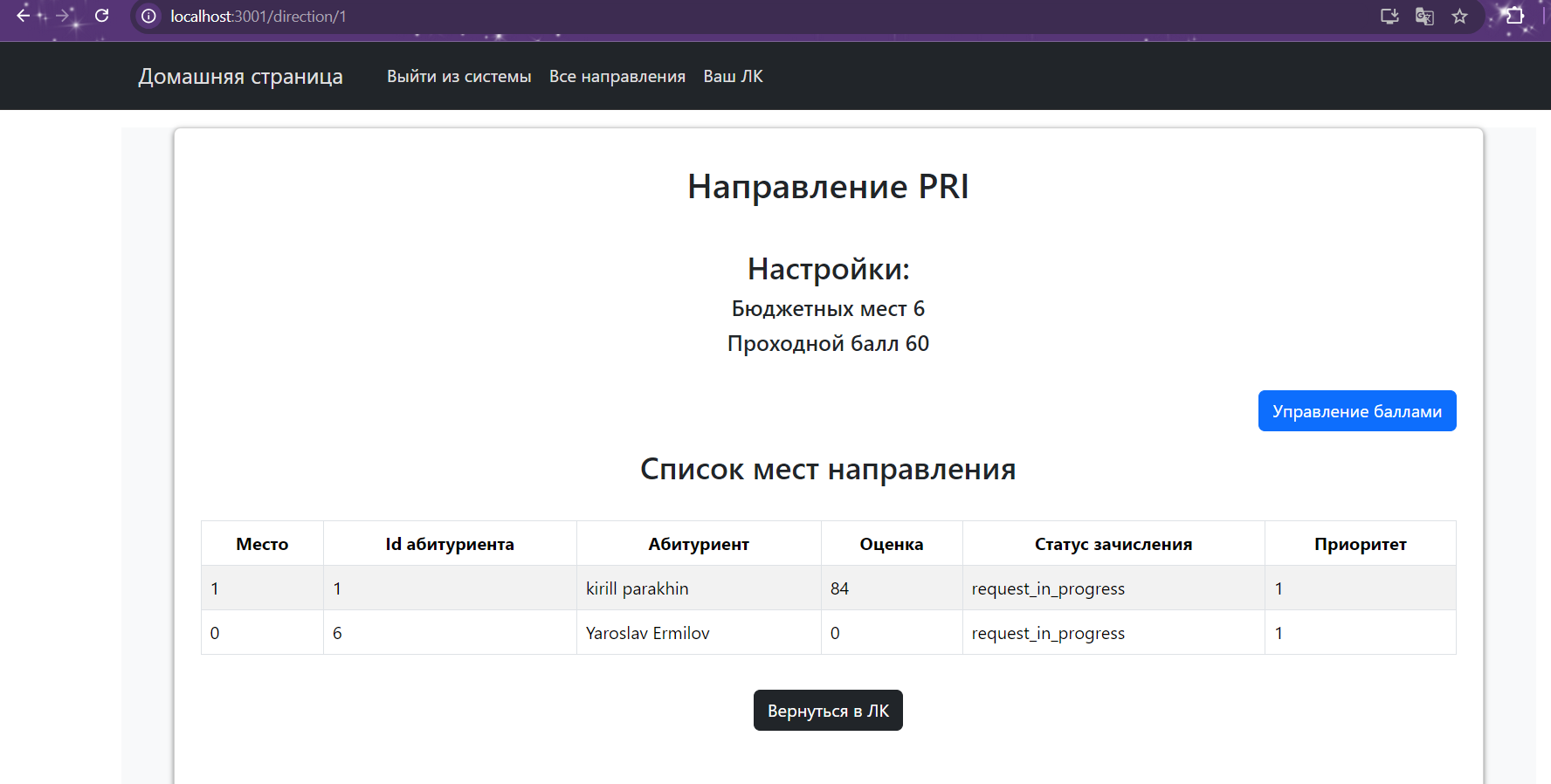
Страница личного кабинета обычного пользователя (есть использование статического изображения)

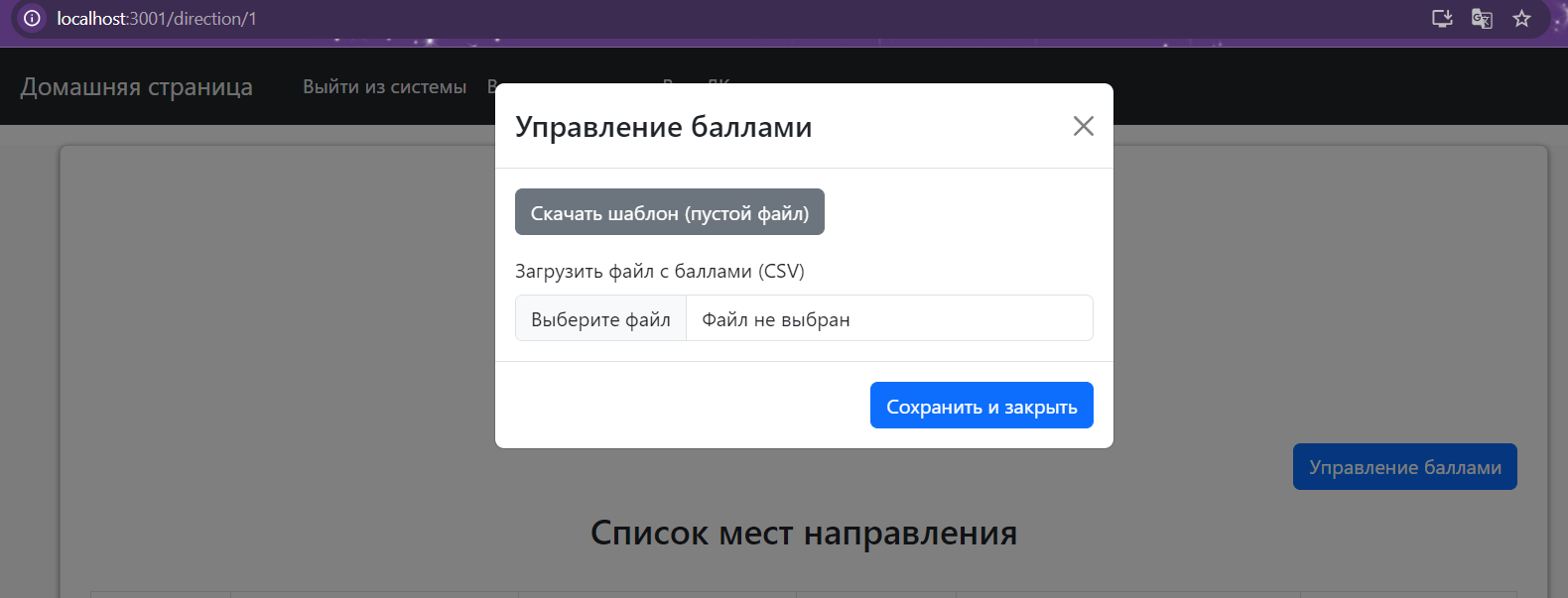


Страница со списком направлений – через нее можно переходить на каждое направление по клику на строку таблицы.

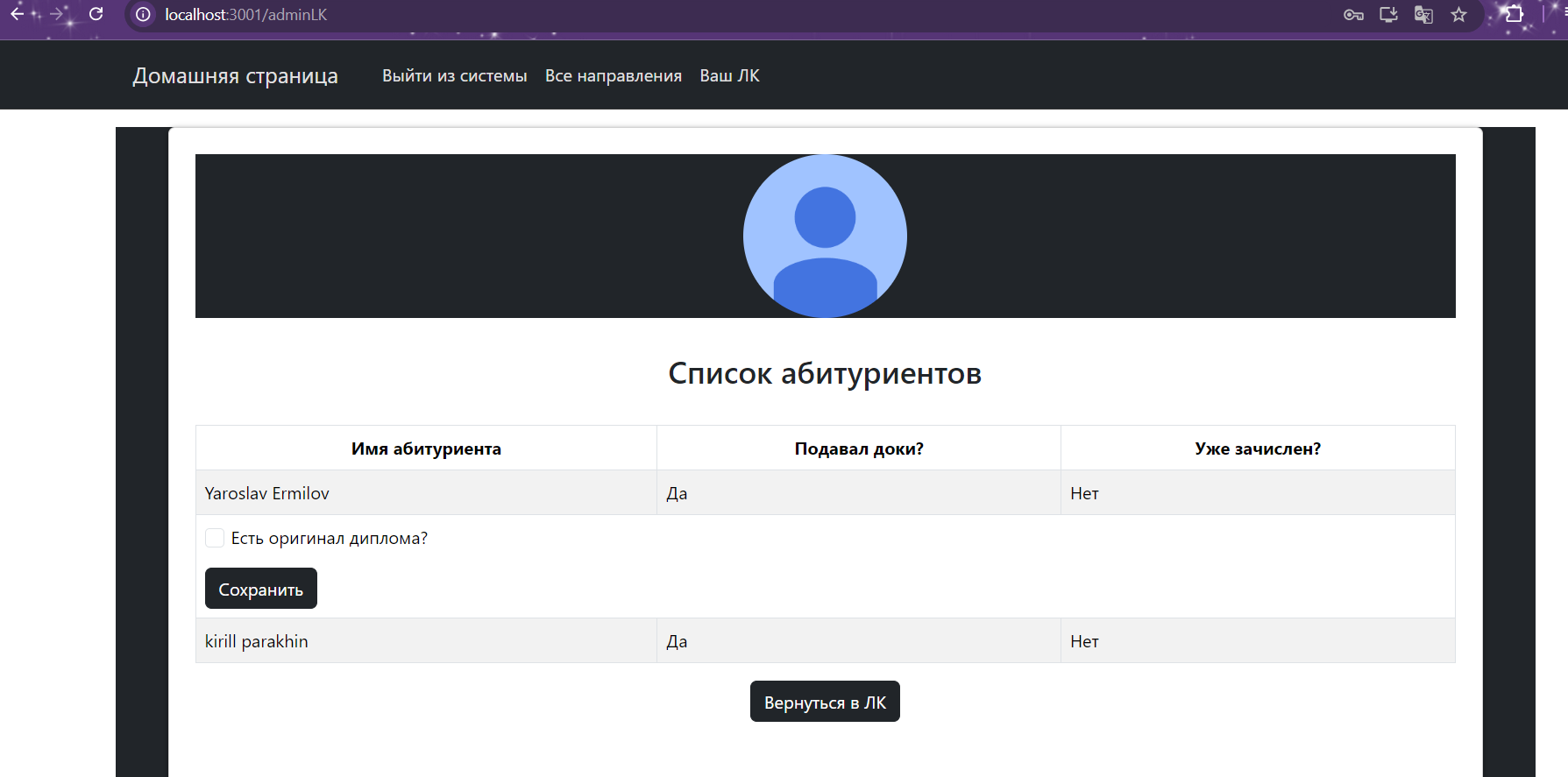


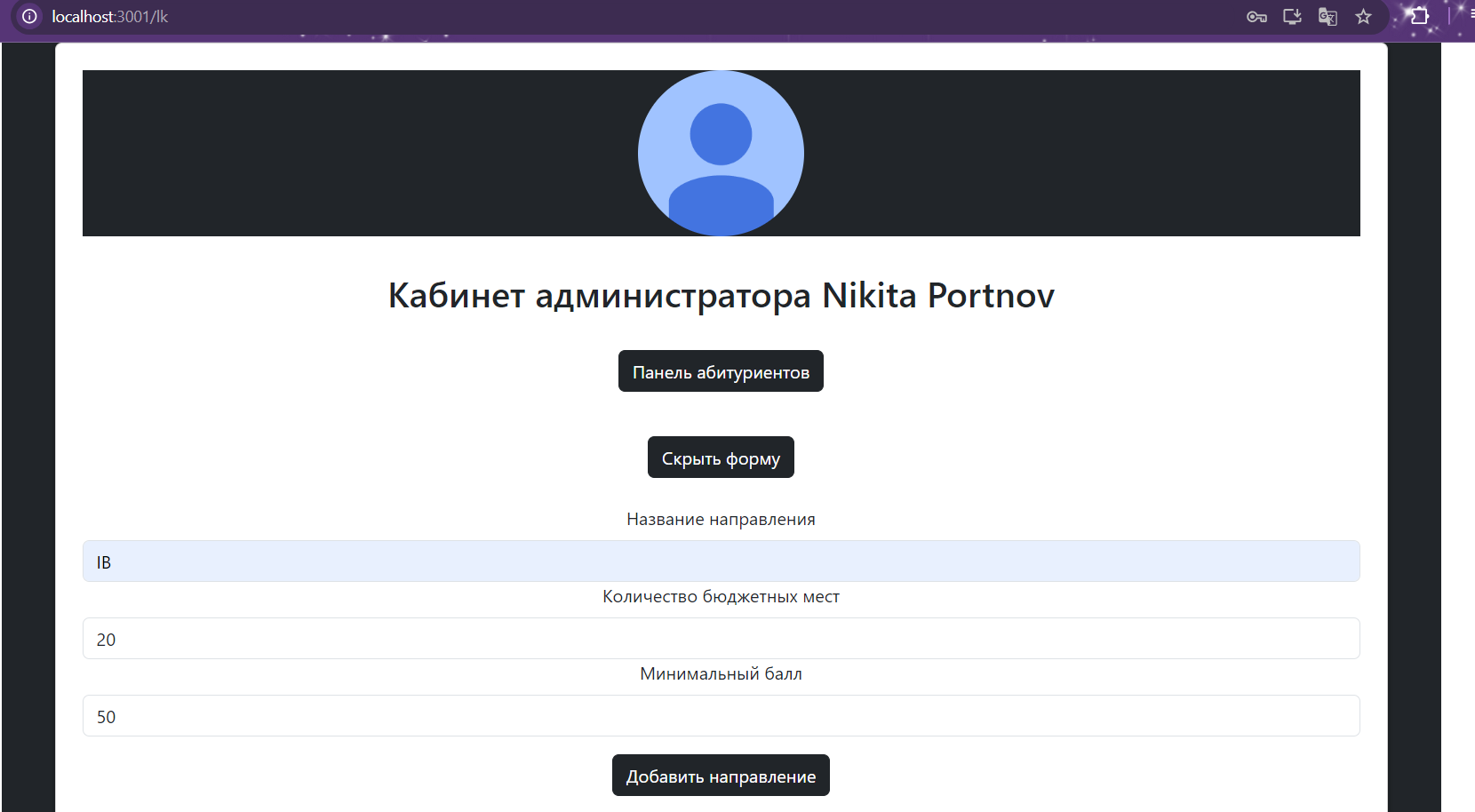
Страничка определенного направления – на ней указываются текущие позиции, настройки направления и кнопочка для управления баллами – там можно загружать файл





Есть отдельный функционал администратора – страничка с абитуриентами и их доками (здесь уже все подали свои доки)





1. Разработка приложения на Flutter

Flutter – он в целом является больше кросс платформенным фрейм ворком – но на нем можно также создавать веб-приложения, использовав адаптации виджетов под то, что бы они нормально выглядели в браузере

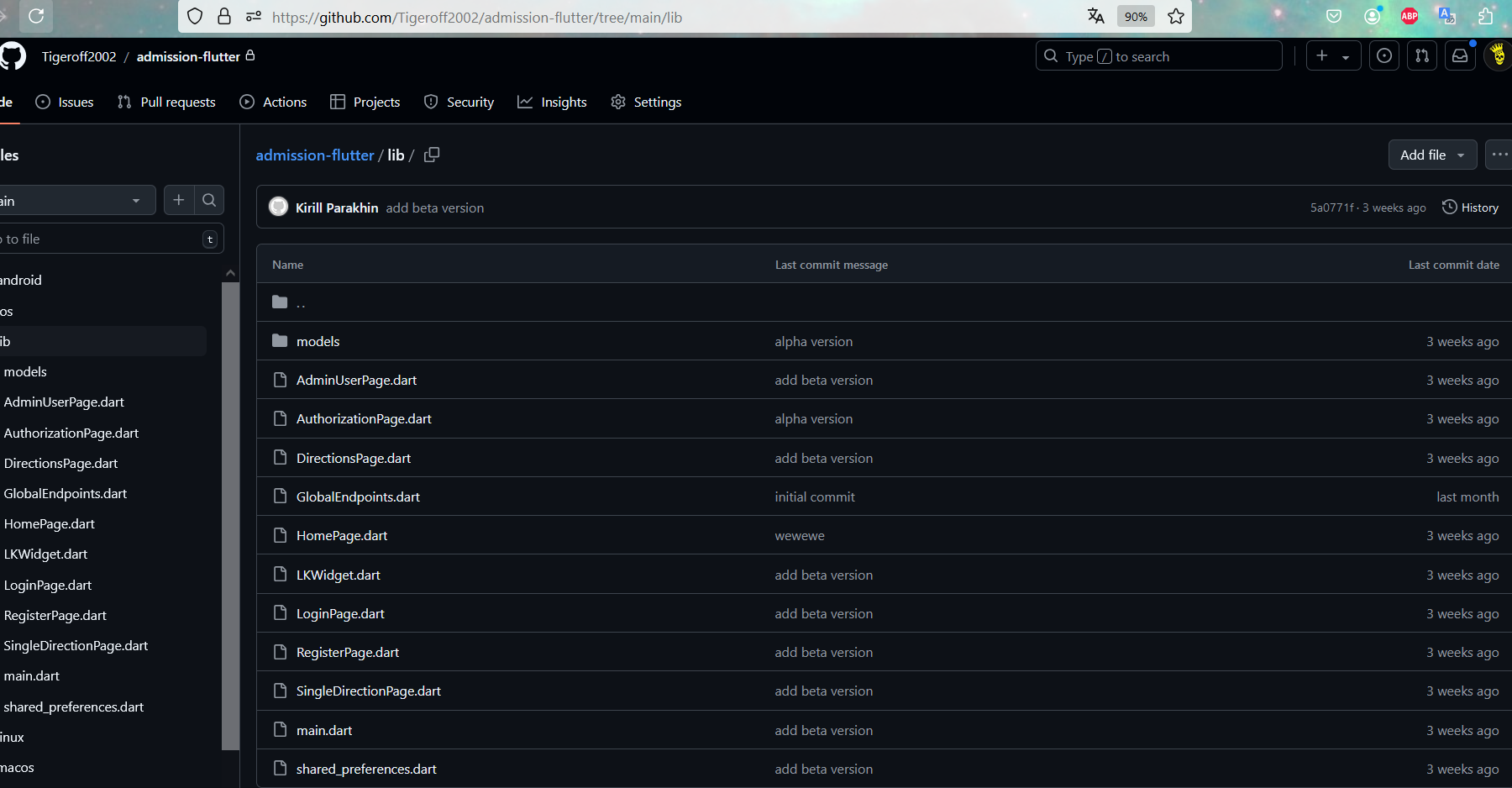


Рисунок 3.1. Репозиторий Flutter - приложения

Собственно, как в задании и просилось – на Flutter я создал точно такие же страницы и старался использовать те же принципы работы.

Собственно, получаются такие аналогии. StatelessWidgets – функциональные компоненты React, StatefulWidget – классовые компоненты.

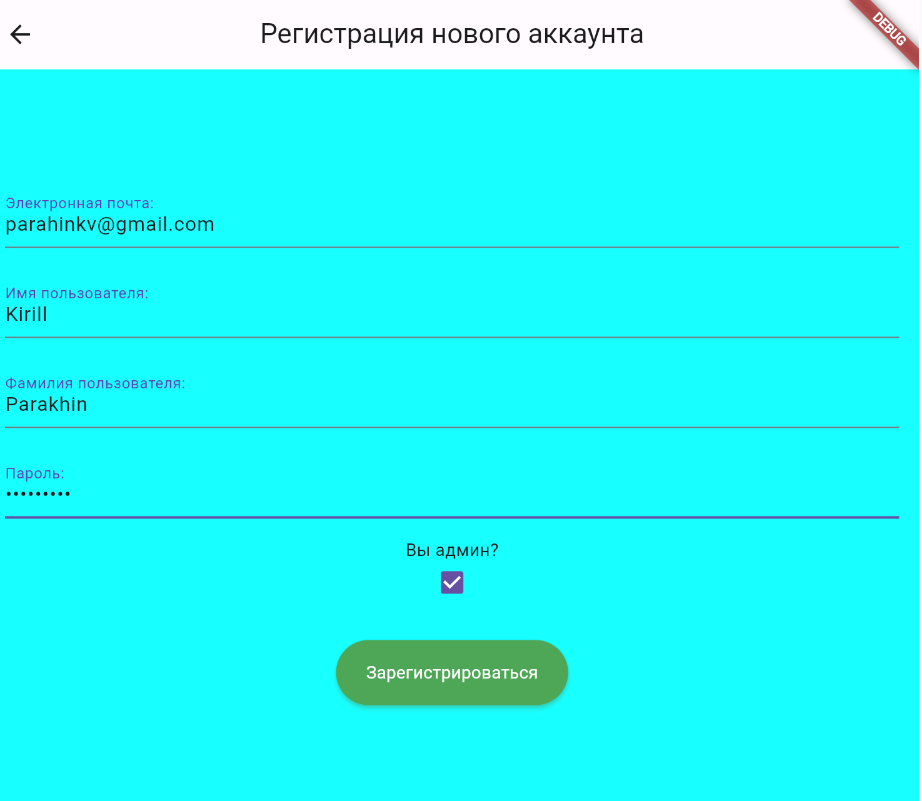
Вместо хуков – тут явно используются виджеты и вызываются их методы. Для работы с контекстом аутентификации пользователя я пользовался кэшированием – для этого есть отдельный файл.

Маппинг по страницам и URLам настраивается в файле main.js

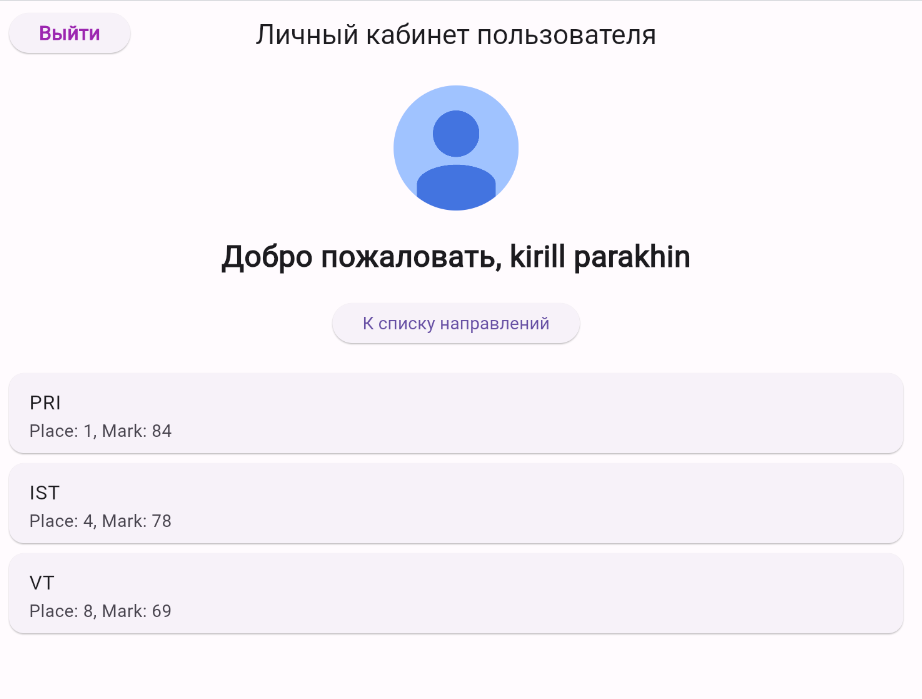
Одним из требований было создание адаптивных страниц приложения – под разные дисплеи. На React я этого не демонстрировал – хотя там это тоже работает

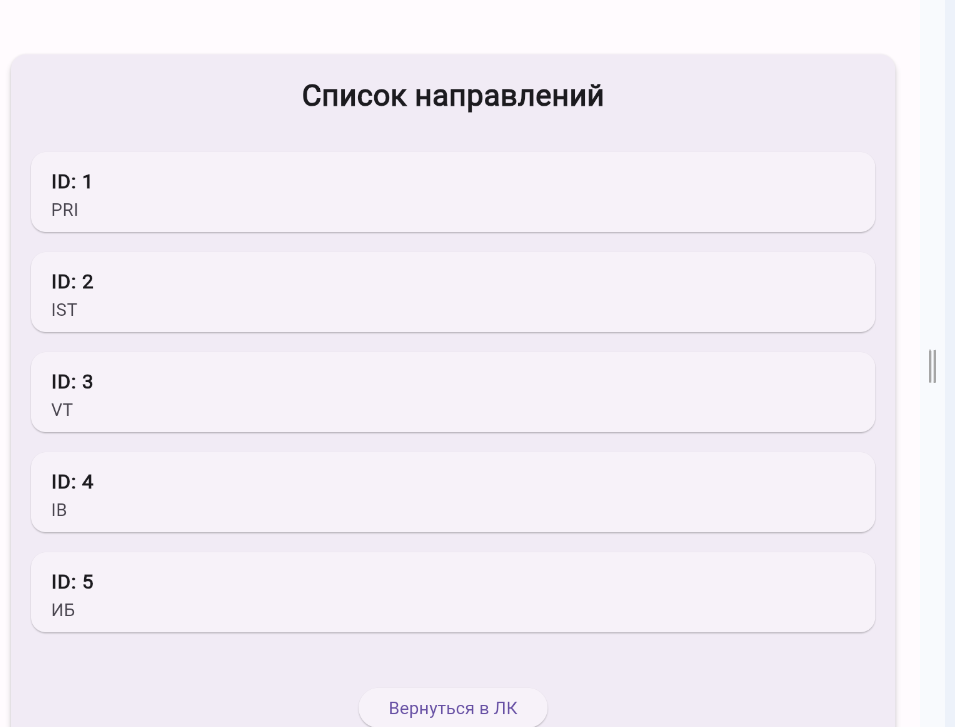
Поэтому покажу в скриншотах для Flutter-приложения

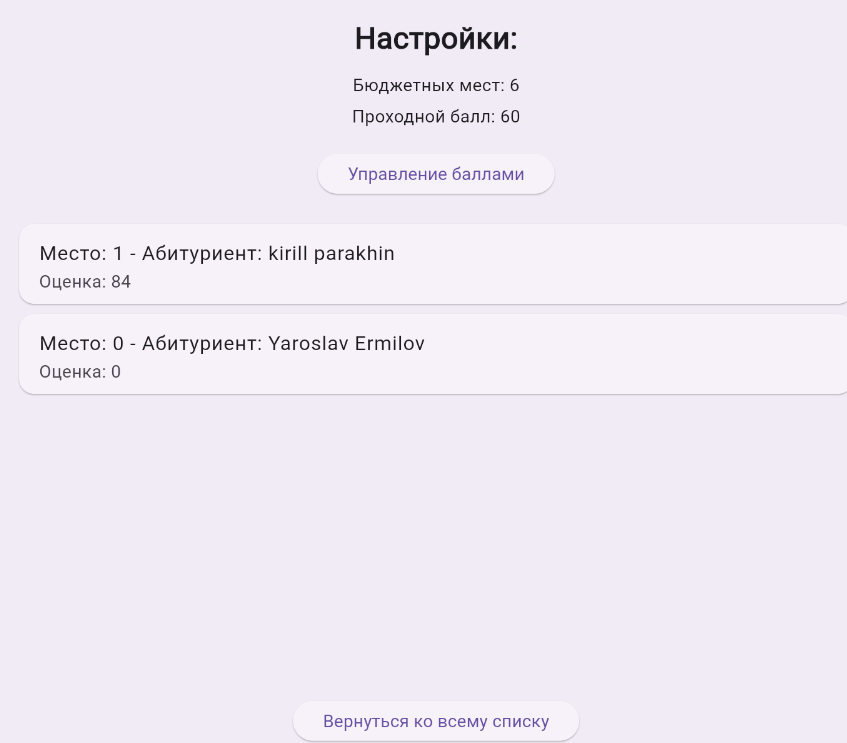
Страница регистрации пользователя:

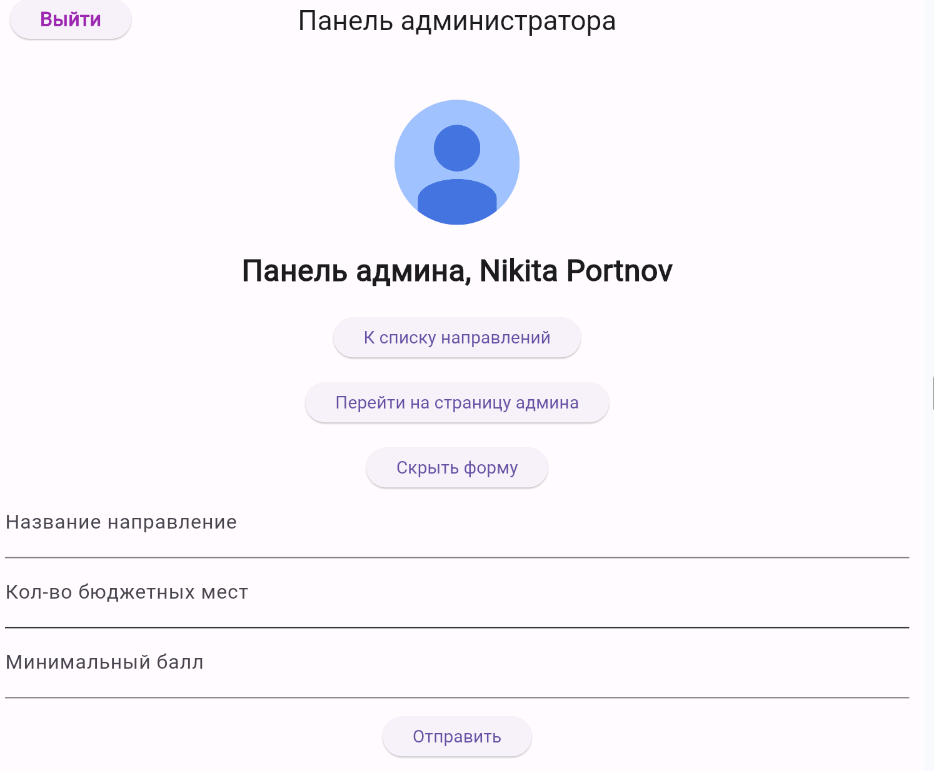


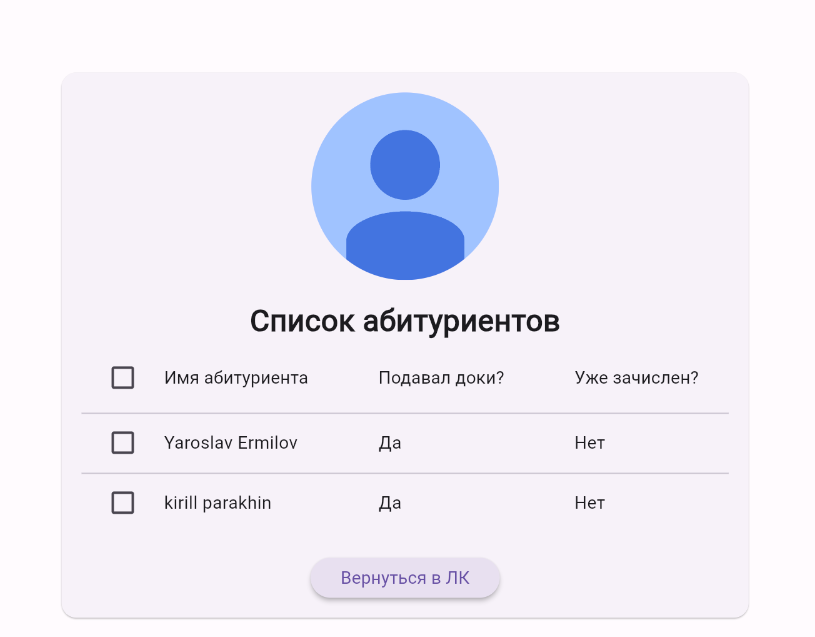
Страница личного кабинета пользователя:











1. Разработка приложения на Vue

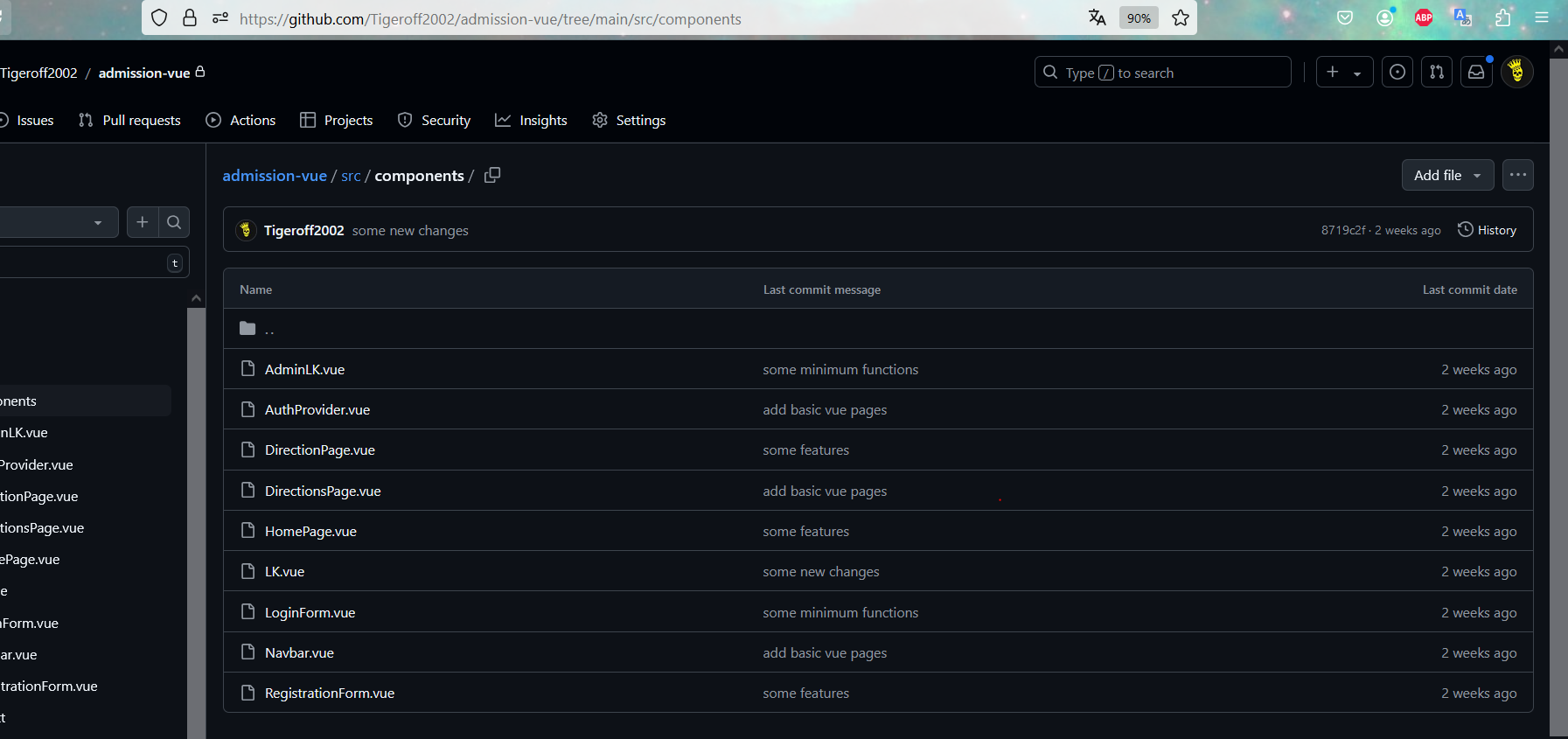
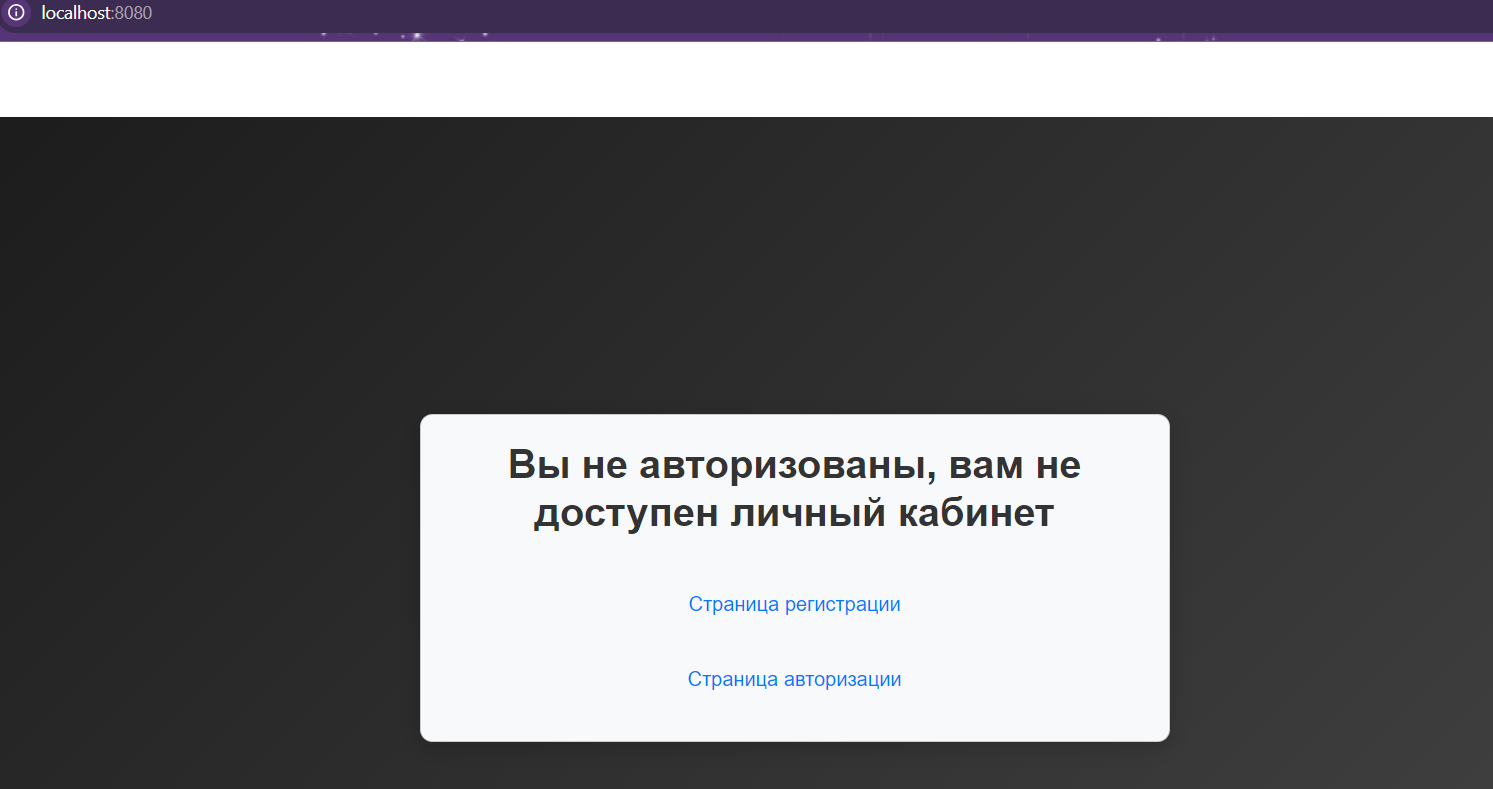
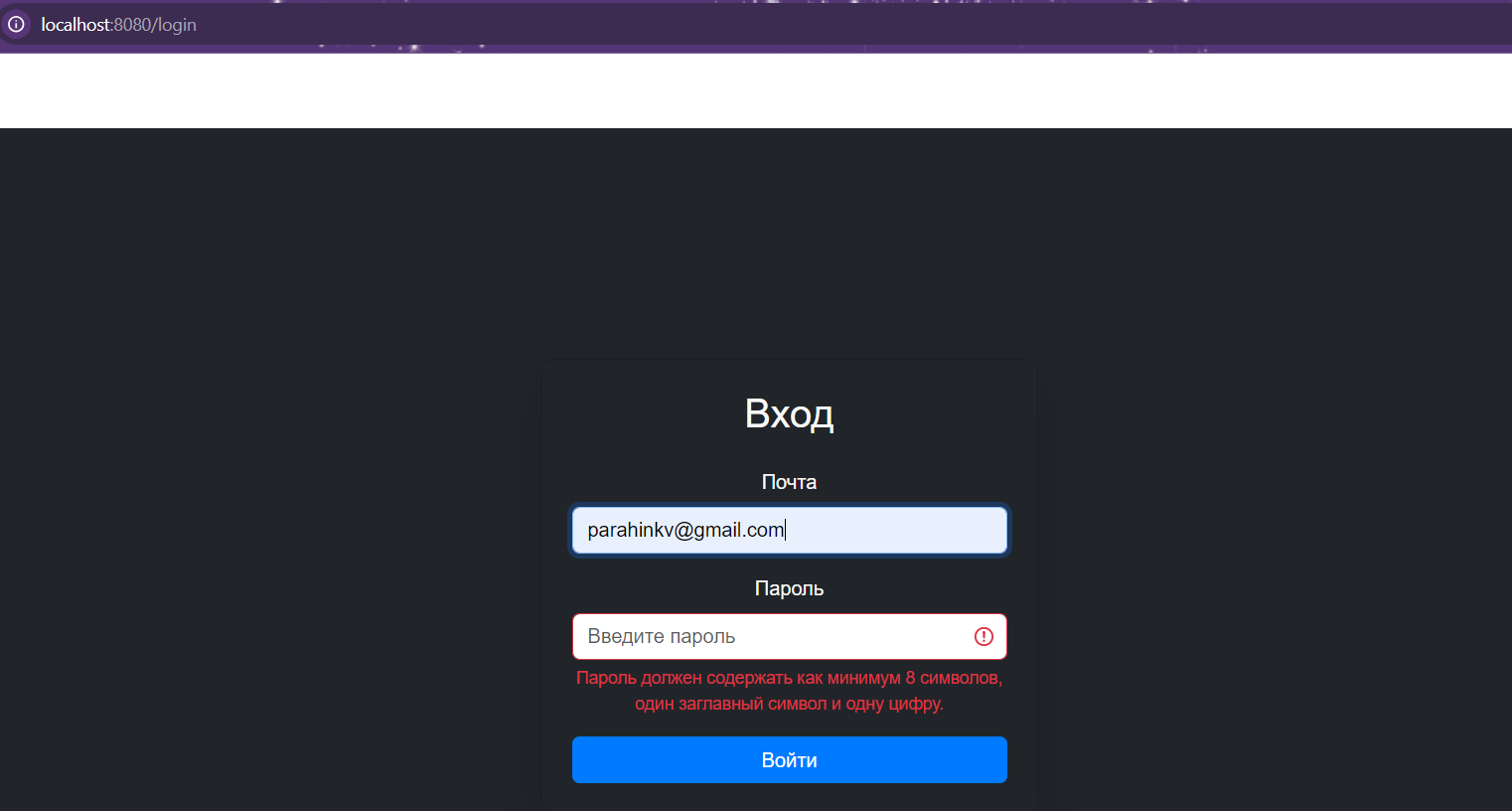


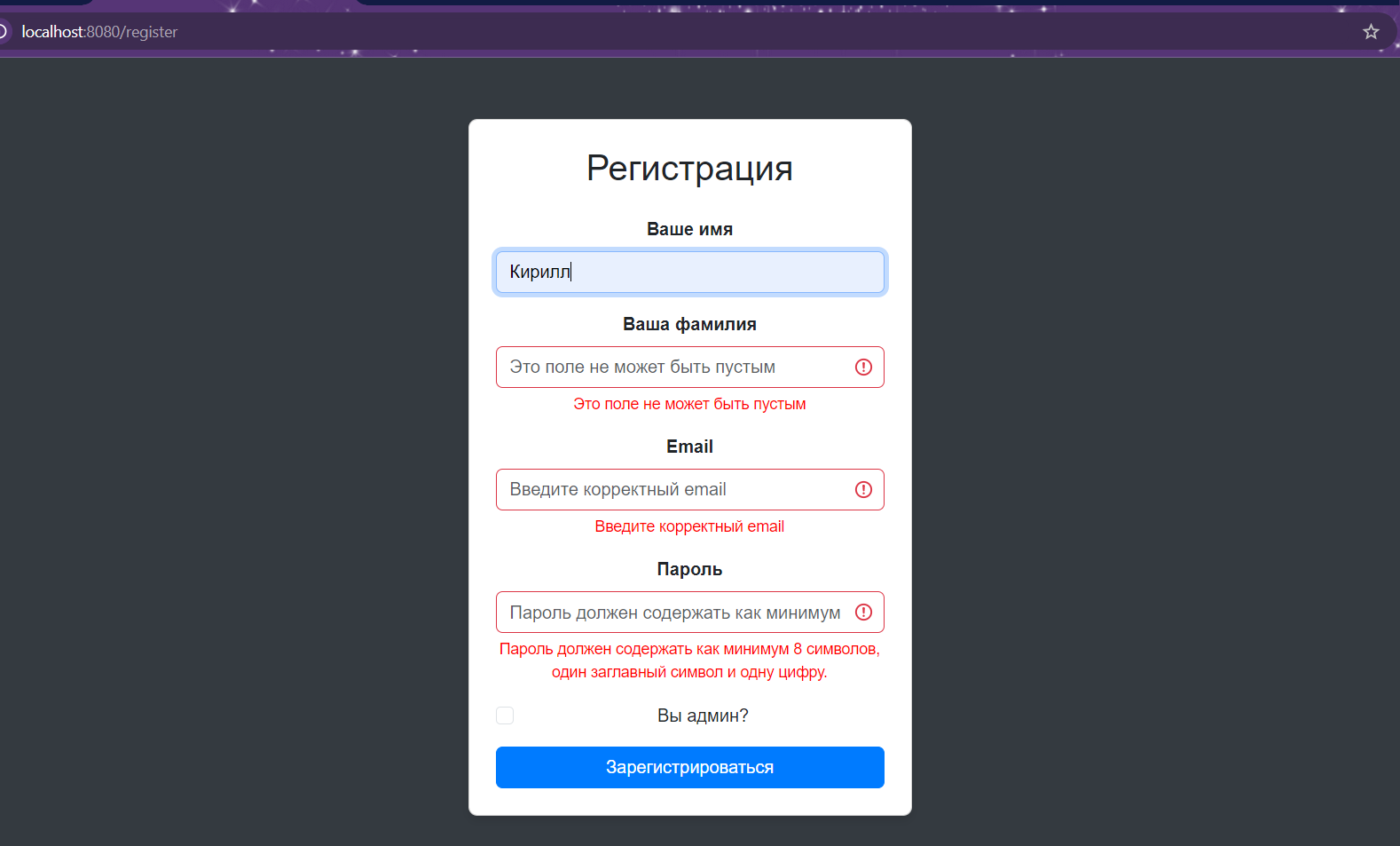
Рисунок 4.1. Репозиторий Vue приложения

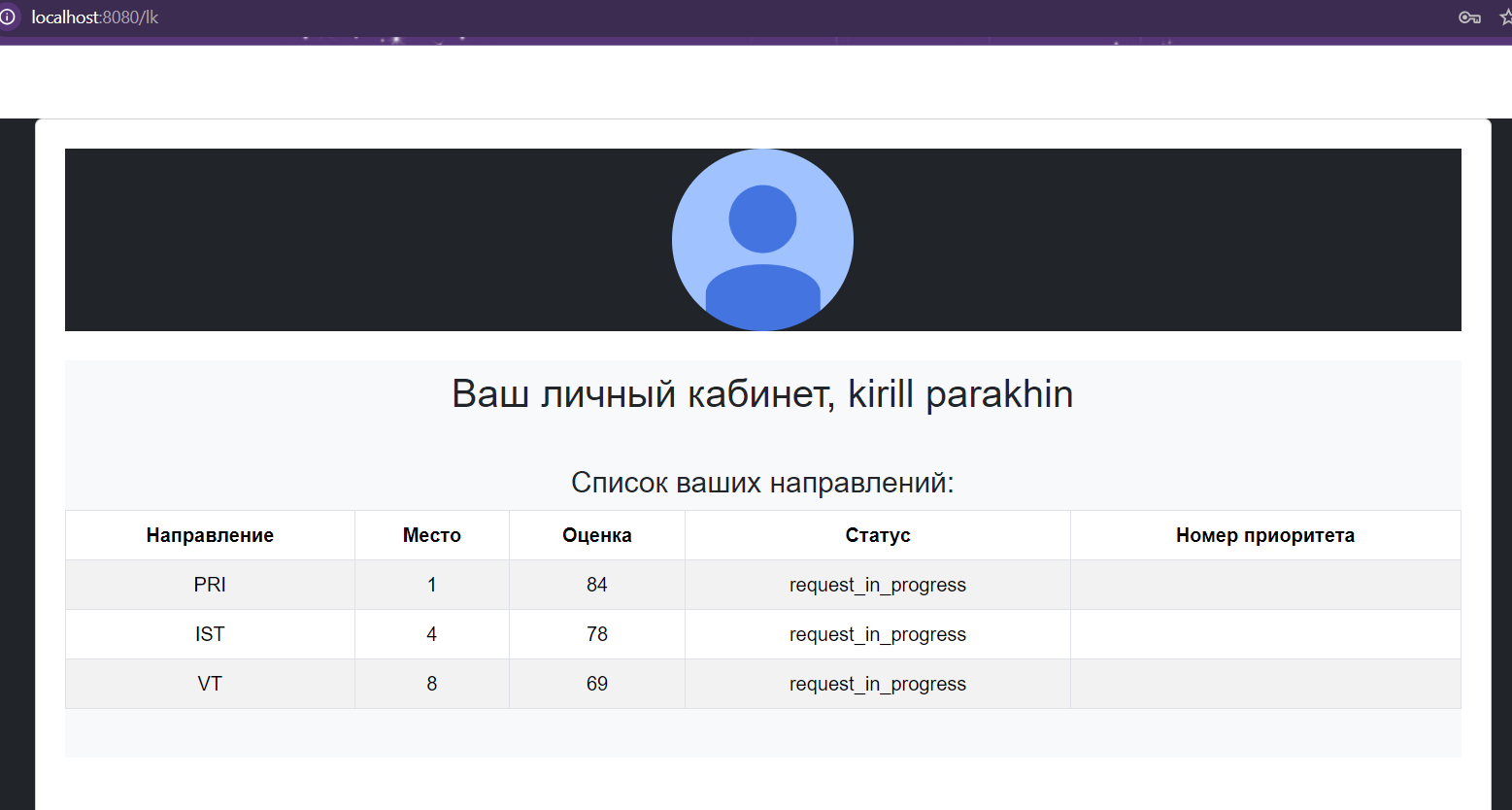
На Vue все делалось по похожему принципу – единственное, что верстку с React не получилось нормально переиспользовать.

В целом, основные вещи делались похожими методами – также делался контекст аутентификации – но как по мне, на Vue это делается более сложно.









На Vue я сделал все по минимуму – просто страницы авторизации и регистрации, а также личный кабинет пользователя. Функционал администратора решил тут не делать.

Для стилей использовалась та же библиотека Bootstrap CSS для npm

Плюсы и минусы технологий из того что я успел понять:

Плюсы технологий:

1. React

Очень быстрая сборка (и пересборка приложения), наличие горячей перезагрузки страницы, удобный маппинг urlов страниц. Удобная работа с контекстом аутентификации и обновления данных с сервера благодаря хукам, удобные библиотеки для работы с HTTP REST и работы с JSON – просто как с ассоциативным массивом, без доп классов моделей.

Наличие всех нужных HTML-элементом в виде REACT декораторов, удобная работа и настройка для них.

1. Flutter

Наличие горячей перезагрузки, наличие маппинга страниц в едином месте. Возможность прямого кэширования данных. Более долгая настройка, но зато лучшая поддерживаемость HTTP REST операций благодаря настройке сериализации и десериализации DTO в отдельных классах моделей. Удобная работа со стейтом виджетом – более удобная, чем на React. Огромное число виджетов для разных целей – их полная адаптивность хоть под веб-приложение, хоть под мобильное приложение.

1. Vue

В целом те же плюсы, что и у React. Есть разделение блоков кода для верстки, скриптов и стилей. А так сложно сказать что то большее, опыта работы с Vue получилось не очень много.

Минусы технологий:

1. React

Отсутствие возможности для прямого кэширования. Адаптивность страниц и ее элементов чуть сложнее настраивается, чем условно на Flutter.

Кажется непростой библиотекой, так как постоянно возникали сложности с пониманием того, как правильно работать с функциональными и классовыми компонентами, для многих страниц из-за ограничений React приходилось создавать врапперы.

1. Flutter

Чуть сложно начинать на нем работать, так как код получается более громоздким, верстка менее интуитивна для обычного веб-разработчика. Но я уже имею опыт работы с ним – поэтому мне было нормально.

Возможно, его использование избыточно для создания простых веб-приложения (и веб-приложений впринципе – так как он более подходит под создание мобильных приложений). Достаточно долго собирает файлы, занимает больше места и требует поэтому больше ресурсов.

1. Vue

Впринципе те же самые минусы, что и у React. Не смотря на мнение, что Vue – это упрщенный React – я тут этого не почувствовал. В целом, у меня возникло больше проблем с ним. Нативная верстка на html тэгах мне не понравилась, работа с контекстом аутентификации долго не поддавалась.

Выводы по использованию технологий:

Для несложных веб - приложений я бы использовал React (Vue – как его аналог во многом, мне не очень понравился). Так как опыта больше у меня было на Flutter – то я бы выбрал его для более глубокого изучения, так как он выглядит более продуманным, масштабируемым и гибким.  
На нем можно писать сразу как веб-приложения, так и собирать на основе них мобильные приложения, внося совсем небольшие правки.

Вывод:

В результате выполнения работы были освоены принципы создания клиентской части веб-приложений с использованием 3 различных технологий.