



Министерство науки и высшего образования Российской Федерации  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
«Московский государственный технический университет  
имени Н.Э. Баумана  
(национальный исследовательский университет)»  
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ \_\_\_\_\_ ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ \_\_\_\_\_

КАФЕДРА \_\_\_\_\_ СИСТЕМЫ ОБРАБОТКИ ИНФОРМАЦИИ И УПРАВЛЕНИЯ \_\_\_\_\_

# РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

## К ДОМАШНЕМУ ЗАДАНИЮ

**НА ТЕМУ:**

**Проверка кода студентов**

Студент ИУ5-51Б  
(Группа)  
(И.О.Фамилия)

\_\_\_\_\_  
(Подпись, дата) Цыпышев Т.А.

Руководитель курсовой работы

\_\_\_\_\_  
(Подпись, дата) А.И. Канев  
(И.О.Фамилия)

2024 г.

## **АННОТАЦИЯ**

Расчётно-пояснительная записка содержит 28 страниц. С приложениями объем составляет 48 страниц. Работа включает в себя 8 диаграмм и 26 изображений системы. В процессе выполнения было использовано 8 источников.

Объектом разработки является система для проверки кода студентов.

Цель работы заключается в создании набора программного обеспечения и сопутствующих веб-сервисов для обработки заявок на проверку кода, включающих проекты с файлами на различных языках программирования.

В рамках работы была разработана архитектура веб-сервиса, создан интерфейс для взаимодействия с ним, разработаны и развернуты веб-сервер, нативное приложение и прогрессивное веб-приложение, обеспечивающие проверку студенческого кода.

Пояснительная записка содержит 2 приложения.

## **СОДЕРЖАНИЕ**

АННОТАЦИЯ	2
ВВЕДЕНИЕ	4
1. ПРЕДМЕТНАЯ ОБЛАСТЬ	6
2. АРХИТЕКТУРА	9
3. АЛГОРИТМЫ	13
4. ОПИСАНИЕ ИНТЕРФЕЙСА	15
ЗАКЛЮЧЕНИЕ	26
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	28
ПРИЛОЖЕНИЕ А ТЕХНИЧЕСКОЕ ЗАДАНИЕ	1
1. Введение	5
2. Назначение разработки	5
3. Стадии и этапы разработки	5
4. Требования к функциональным характеристикам	6
5. Требования к составу и параметрам технических средств	11
6. Требования к информационной и программной совместимости	11
ПРИЛОЖЕНИЕ Б СПИСОК НТТР МЕТОДОВ	5

## **ВВЕДЕНИЕ**

В современном мире цифровые технологии играют ключевую роль в образовательном процессе, что особенно актуально для проверки кода студентов. Автоматизированные системы проверки кода предоставляют удобный доступ к анализу и оценке программных решений на различных языках программирования. С развитием технологий такие системы стали неотъемлемой частью учебного процесса, обеспечивая быструю и точную проверку кода, что значительно повышает качество обучения. Однако, несмотря на этот прогресс, многие образовательные платформы сталкиваются с проблемами в управлении проверками студенческих работ, что может негативно сказываться на удобстве использования и объективности оценки.

Целью данной системы является автоматизация процесса проверки кода студентов, включая управление заявками на проверку, поддержку различных языков программирования и анализ качества кода. Система должна предоставлять удобный интерфейс, позволяющий студентам загружать свои проекты, отслеживать их статус и получать детализированные результаты проверки. Кроме того, система должна обеспечивать эффективное управление процессами со стороны преподавателей и проверяющих, включая обработку заявок, анализ кода и вынесение заключений.

Назначение системы заключается в обеспечении удобного и прозрачного взаимодействия между студентами и проверяющими. Студенты смогут загружать проекты, содержащие файлы на различных языках программирования, указывая необходимые параметры проверки. Преподаватели и проверяющие получат возможность просматривать список заявок, анализировать код, предоставлять комментарии и контролировать процесс проверки. Кроме того, система будет поддерживать различные языки программирования, обеспечивая гибкость и широкий спектр функциональных возможностей.

Такая система позволит реализовать комплексный подход к управлению процессом проверки кода студентов, обеспечивая удобство, прозрачность и высокую эффективность работы. Студенты получают удобный инструмент для отправки своих программных проектов на проверку, а преподаватели смогут эффективно управлять процессом проверки, обеспечивая объективность и высокое качество оценки.

Нефункциональные требования к разрабатываемой системе:

1. Должна поддерживать кроссплатформенность.
2. Интерфейс системы и текст ошибок должны быть русифицированы.

В ходе работы необходимо выполнить следующие задачи:

1. Разработать дизайн приложения в Figma на основе [github.com](https://github.com).
2. Ознакомиться с разработкой бэкенда с использованием языка программирования Golang
3. Разработать структуру и создать базу данных PostgreSQL, подключить её к бэкенду.
4. Создать веб-сервис с бизнес-логикой обработки заявок, кроме авторизации, для использования в SPA.
5. Добавить авторизацию, использовать Redis для хранения сессий, а также внедрить Swagger в веб-сервис.
6. Разработать базовый интерфейс приложения для гостя на React.
7. Внедрить менеджер состояний Redux Toolkit для хранения значений фильтров, добавить адаптивность и PWA.
8. Завершить разработку интерфейса пользователя в React, использовать для взаимодействия с веб-сервисом Axios.
9. Реализовать React-интерфейс проверяющего, внедрить Real-time web.
10. Разработать десктопное приложение на Tauri.
11. Развернуть приложение на GitHub Pages.
12. Подготовить документацию, включая РПЗ, ТЗ и набор диаграмм.

13. Оформить git-репозиторий на сервисе GitHub, содержащий исходный код проекта.

### **ПРЕДМЕТНАЯ ОБЛАСТЬ**

Студенту предоставляется возможность загружать файлы с кодом в рамках заявки, формируя черновик проекта. После добавления хотя бы одного файла студент может перейти к его редактированию. Для успешной отправки заявки требуется указать основные параметры проверки, такие как язык программирования и критерии оценки. При необходимости из проекта можно удалить ненужные файлы. Более того, на этапе редактирования проект может быть полностью удален его владельцем.

После того как студент заполнил все данные о проекте и добавил нужные файлы, он может отправить заявку на проверку. После отправки заявка доступна только для просмотра, редактирование становится невозможным. Студент может отслеживать изменения статусов всех своих заявок.

Проверяющий отслеживает поступающие заявки в системе. Он может фильтровать их по имени студента, статусу и дате подачи. Для уточнения деталей или запроса дополнительных материалов проверяющий связывается со студентом по электронной почте, указанной в его аккаунте.

Если заявка принята к проверке, система фиксирует её статус. В случае невозможности проверки (например, если код содержит критические ошибки или не соответствует требованиям) заявка отклоняется. После завершения или отклонения проверки её статус изменить нельзя.

Бизнес-процесс создания, оформления и завершения или отклонения заявки описан в нотации BPMN 2.0 и представлен на рисунке 1.

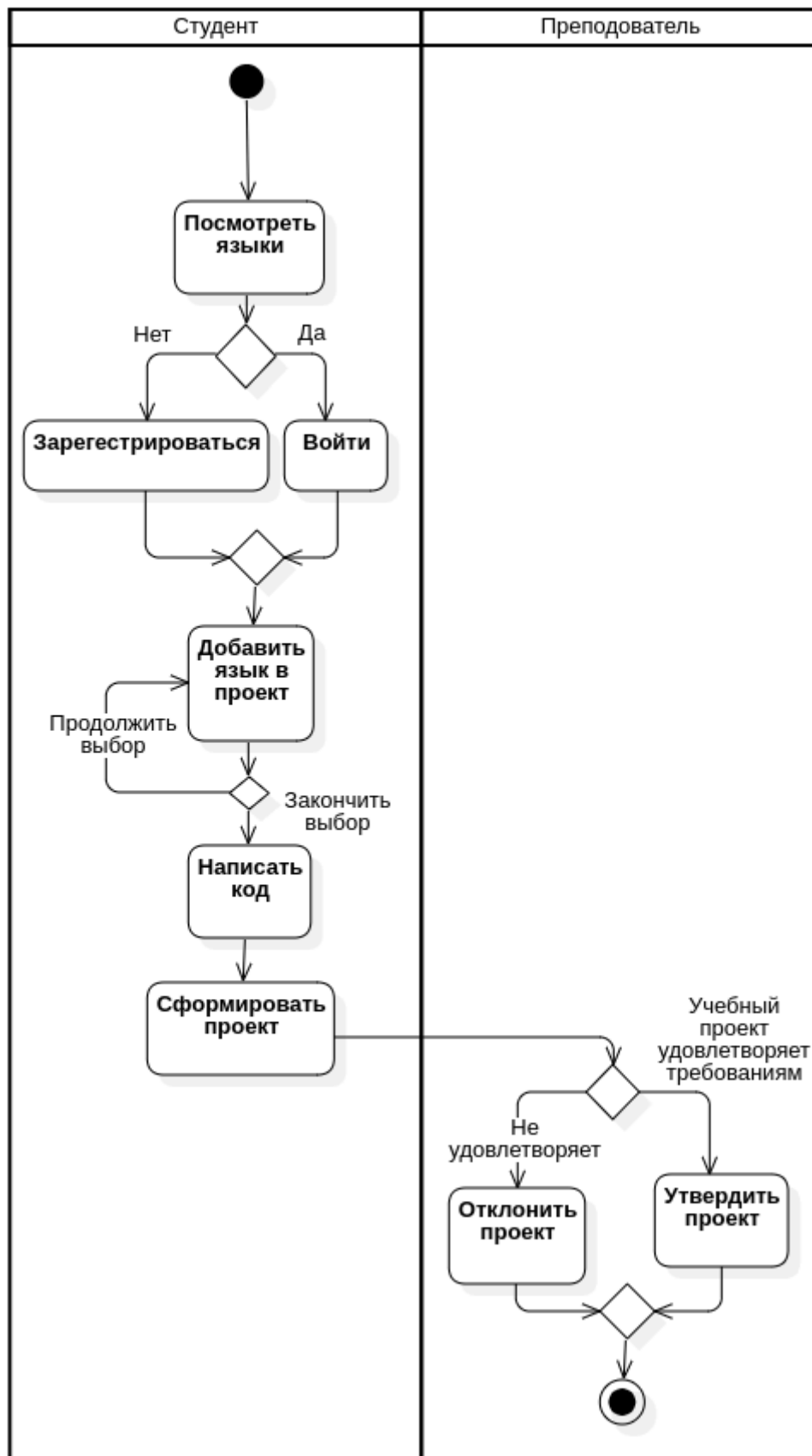


Рисунок 1 — Диаграмма взаимодействия в BPMN2.0

Диаграмма состояний приведена на рисунке 2.

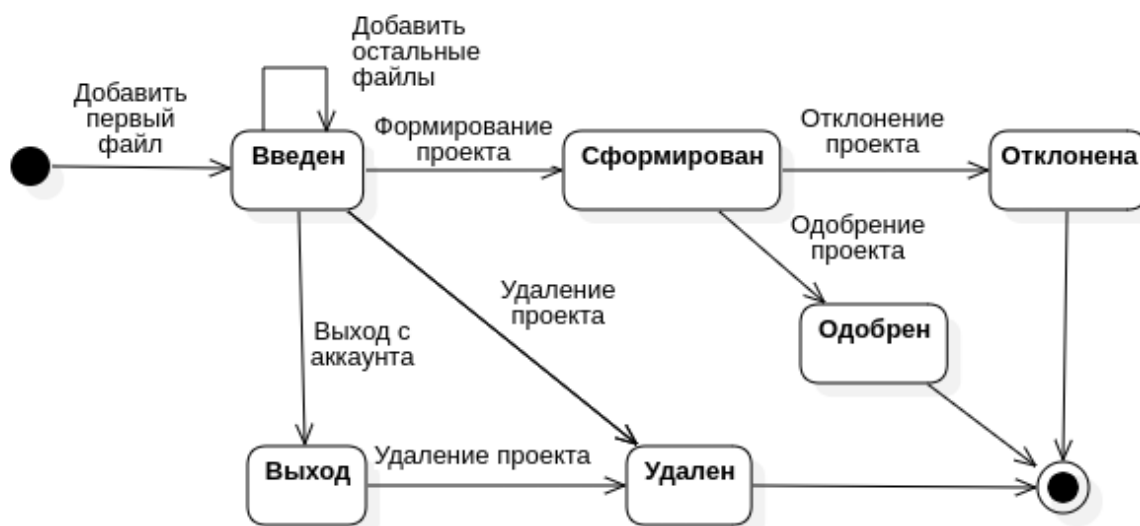


Рисунок 2 — Диаграмма состояний заказов

Проверяющий может редактировать список доступных языков программирования, а именно: изменять сведения о уже имеющихся в системе языках, добавлять новые и удалять устаревшие. При необходимости проверяющий может создать и оформить заявку вручную.

На рисунке 3 представлена диаграмма прецедентов, детально описывающая функции пользователей с различными ролями.



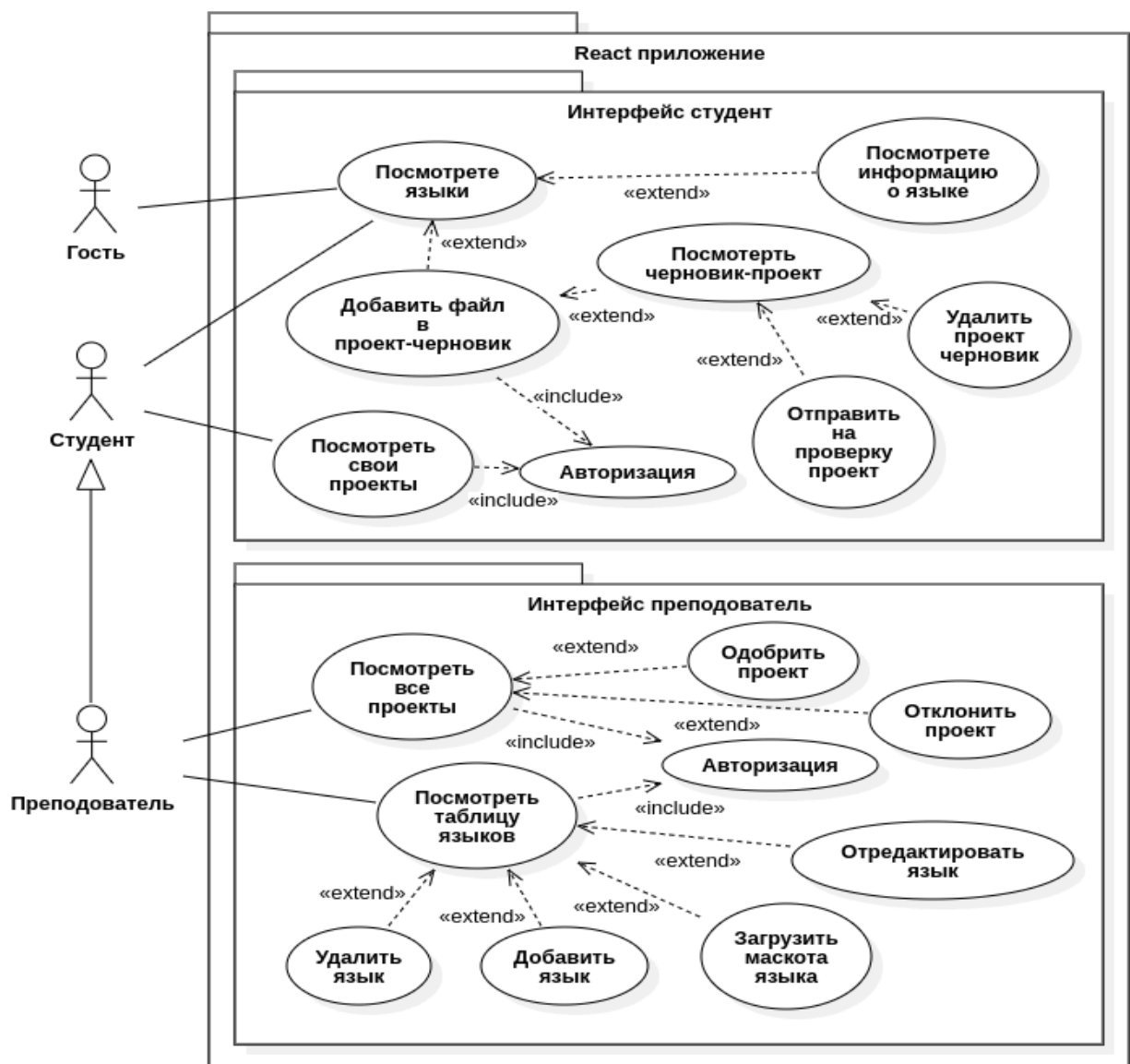


Рисунок 3 — Диаграмма прецедентов

## АРХИТЕКТУРА

Архитектура системы представлена на диаграмме развертывания, изображенной на рисунке 4.

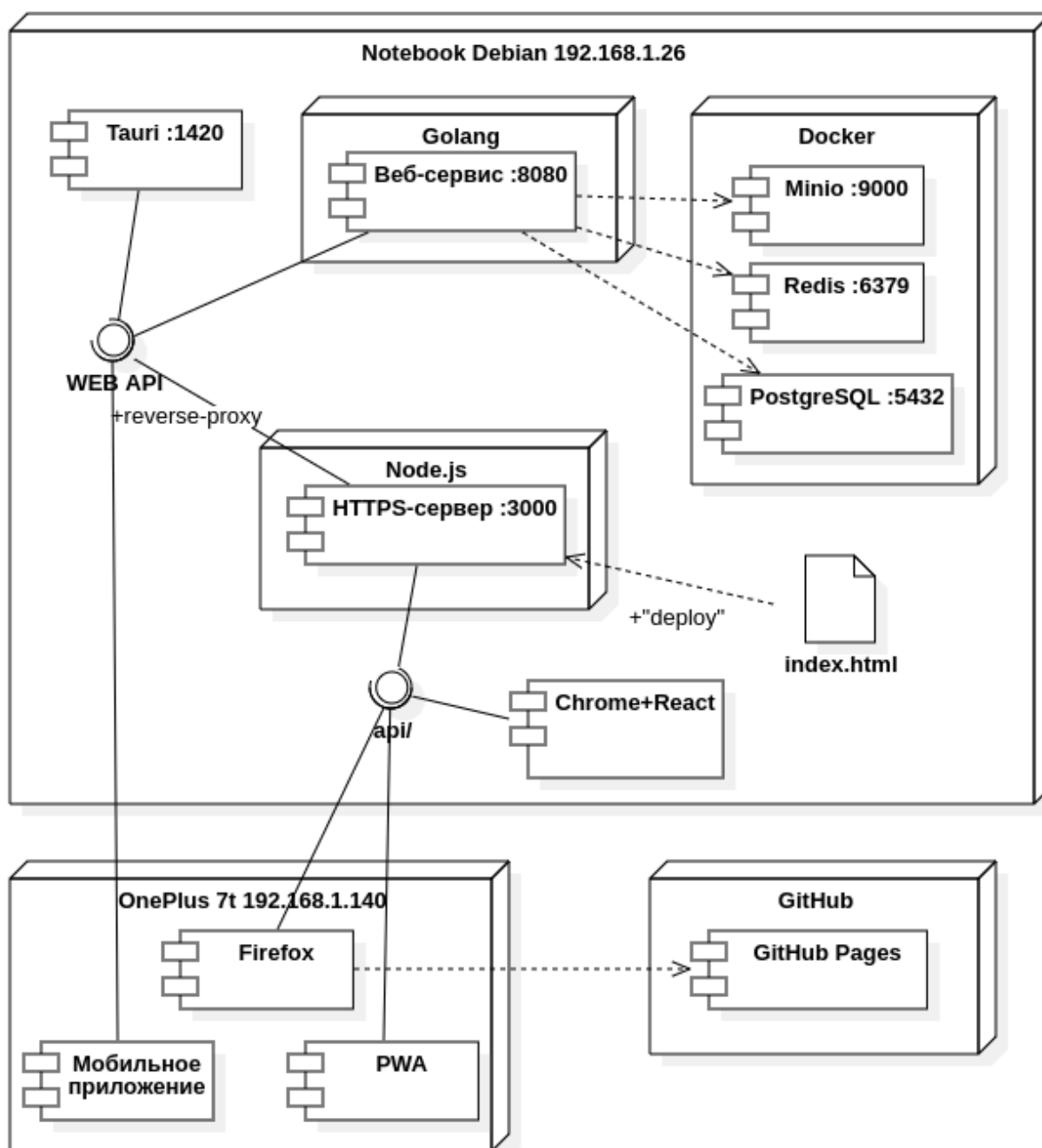


Рисунок 4 — Диаграмма развертывания

Веб-сервис, реализованный на языке программирования Go [3], взаимодействует с серверами Minio [3], Redis [4] и СУБД PostgreSQL [5]. В Redis осуществляется хранение активных сессий пользователей, а при выходе пользователя из системы – их удаление.

В качестве основного языка программирования выбран Golang, отличающийся высокой производительностью, простотой синтаксиса и эффективной работой с многопоточностью. Использование Go позволяет реализовать легковесный и быстрый бэкенд, способный обрабатывать

большое количество запросов. Для работы с API используется фреймворк Fiber, обеспечивающий высокую скорость обработки HTTP-запросов.

Согласно текущей политике импортозамещения, для хранения данных была выбрана СУБД PostgreSQL. Данная СУБД является стандартом индустрии и обеспечивает надежное хранение данных.

Структура данных приведена на ER-диаграмме (рисунок 5). Модель услуг системы представляет собой набор полей, характеризующих языки программирования. Сведения о поддерживаемых языках хранятся в таблице languages. Данные о заявках студентов хранятся в таблице projects. Для хранения в одной заявке нескольких файлов с кодом используется промежуточная таблица files, с помощью которой реализуется связь «многие ко многим». Данные о пользователях системы хранятся в таблице users.

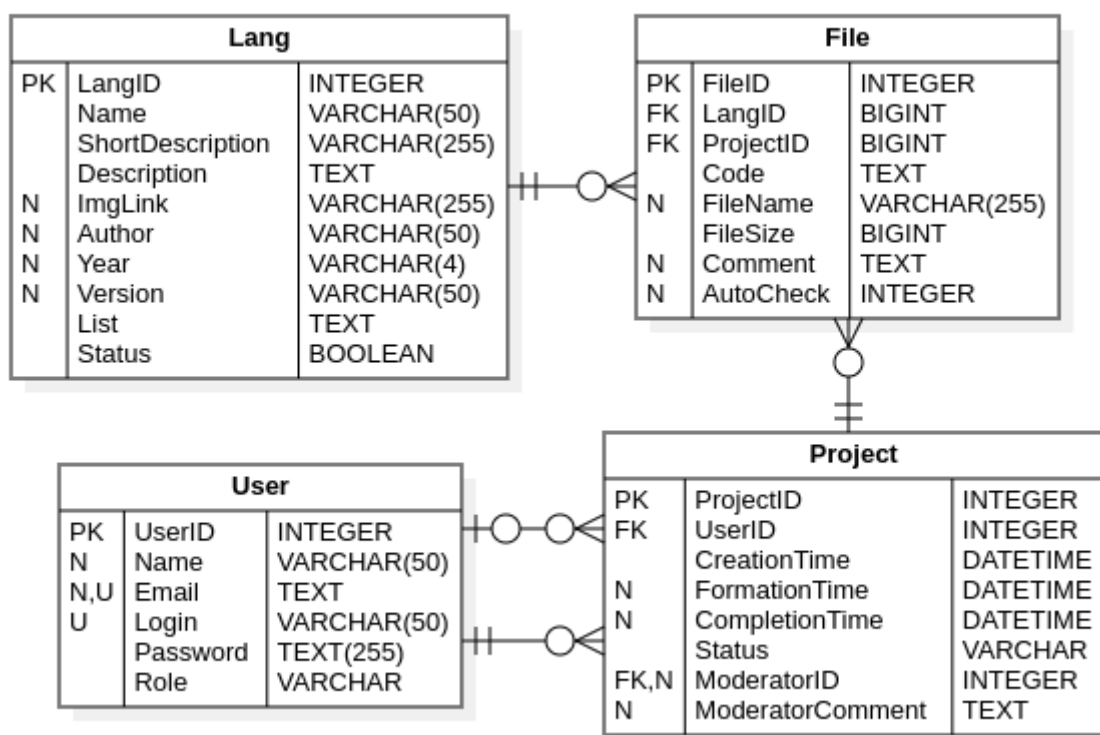


Рисунок 5 — ER-диаграмма

Устройство бэкенда разработанной системы отражено на диаграмме структурных компонентов (рисунок 6). Пользователи взаимодействуют с API-сервисом, который обрабатывает запросы и направляет их в

соответствующие доменные модули. Доменные модули содержат бизнес-логику и работают с моделями данных, которые, в свою очередь, связаны с таблицами в базе данных.

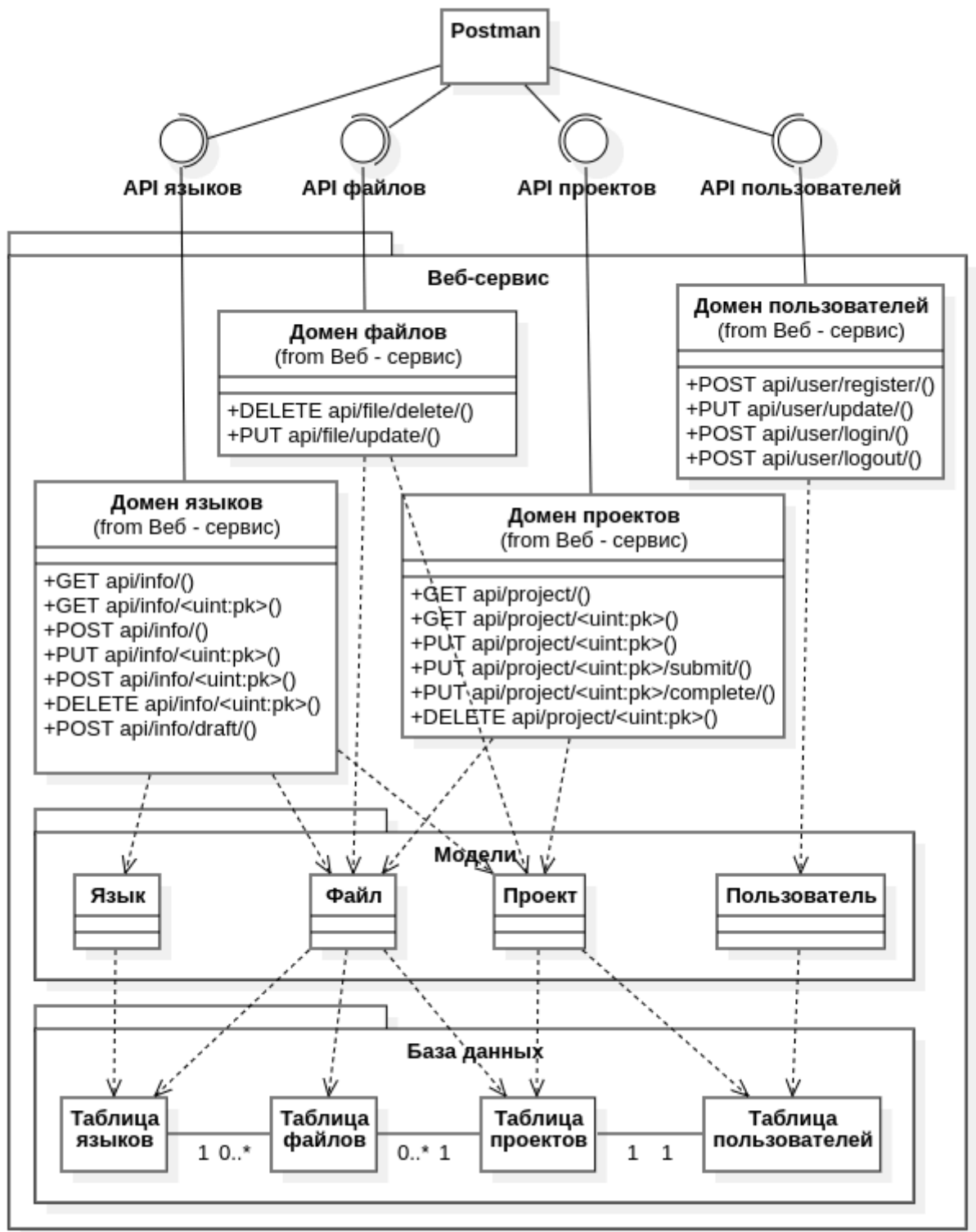


Рисунок 6 — Диаграмма классов бэкенда

Диаграмма классов фронтенда визуализирует связь фронтенда и бэкенда (рисунок 7). Каждая страница связана с API, используемым для

взаимодействия с данными на соответствующей страницы. Фронтенд реализован с помощью фреймворка React [10].

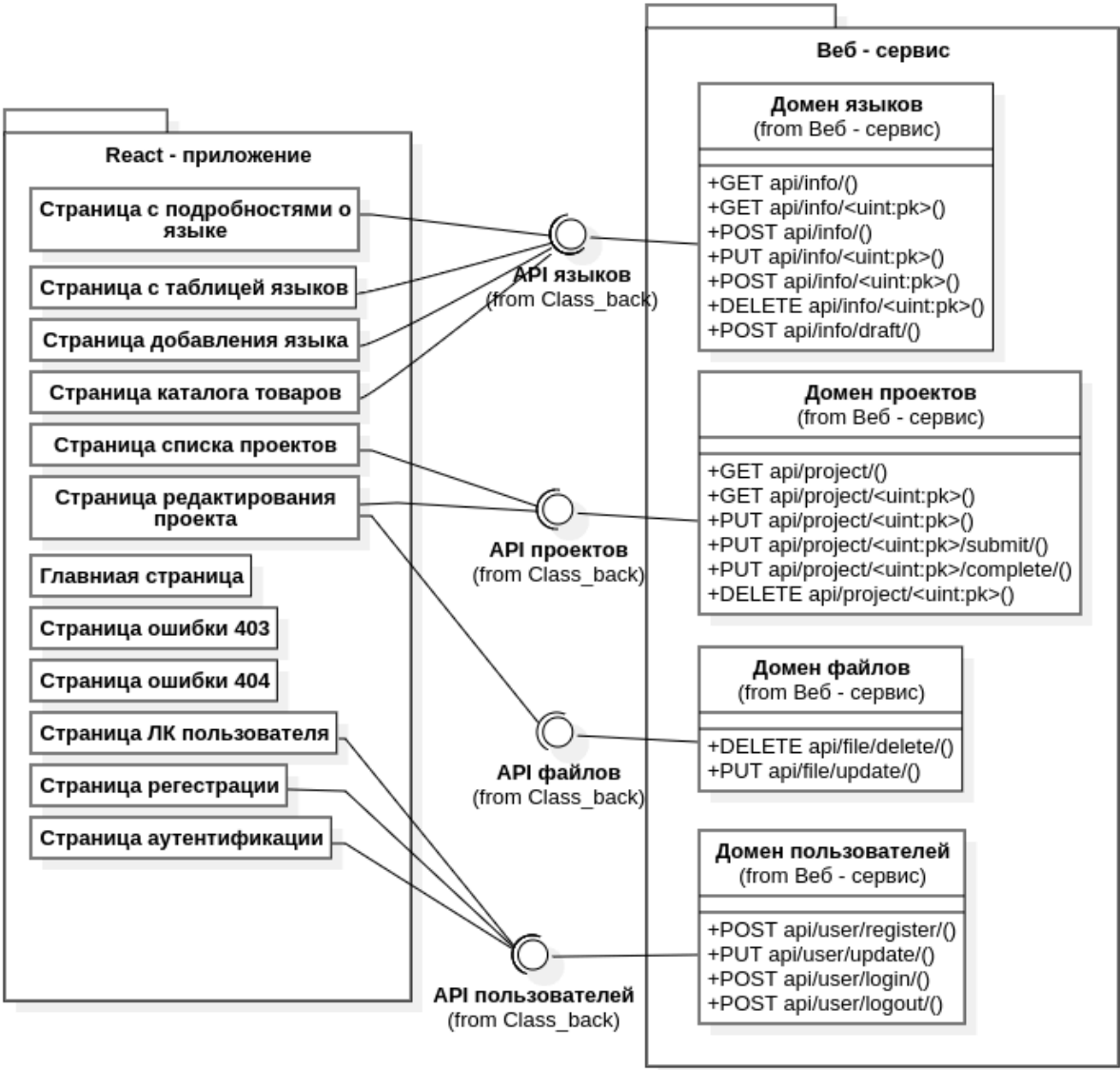


Рисунок 7 — Диаграмма классов фронтенда

## АЛГОРИТМЫ

Алгоритм работы разработанной системы отображен на диаграмме последовательности, приведенной на рисунке 8. В основе системы лежит веб-сервис, реализованный на Go, внутри которого находится вся бизнес-логика. Он предоставляет доступ к методам следующих доменов: языки программирования, заявки, файлы проектов, пользователи. Методы следуют REST API.

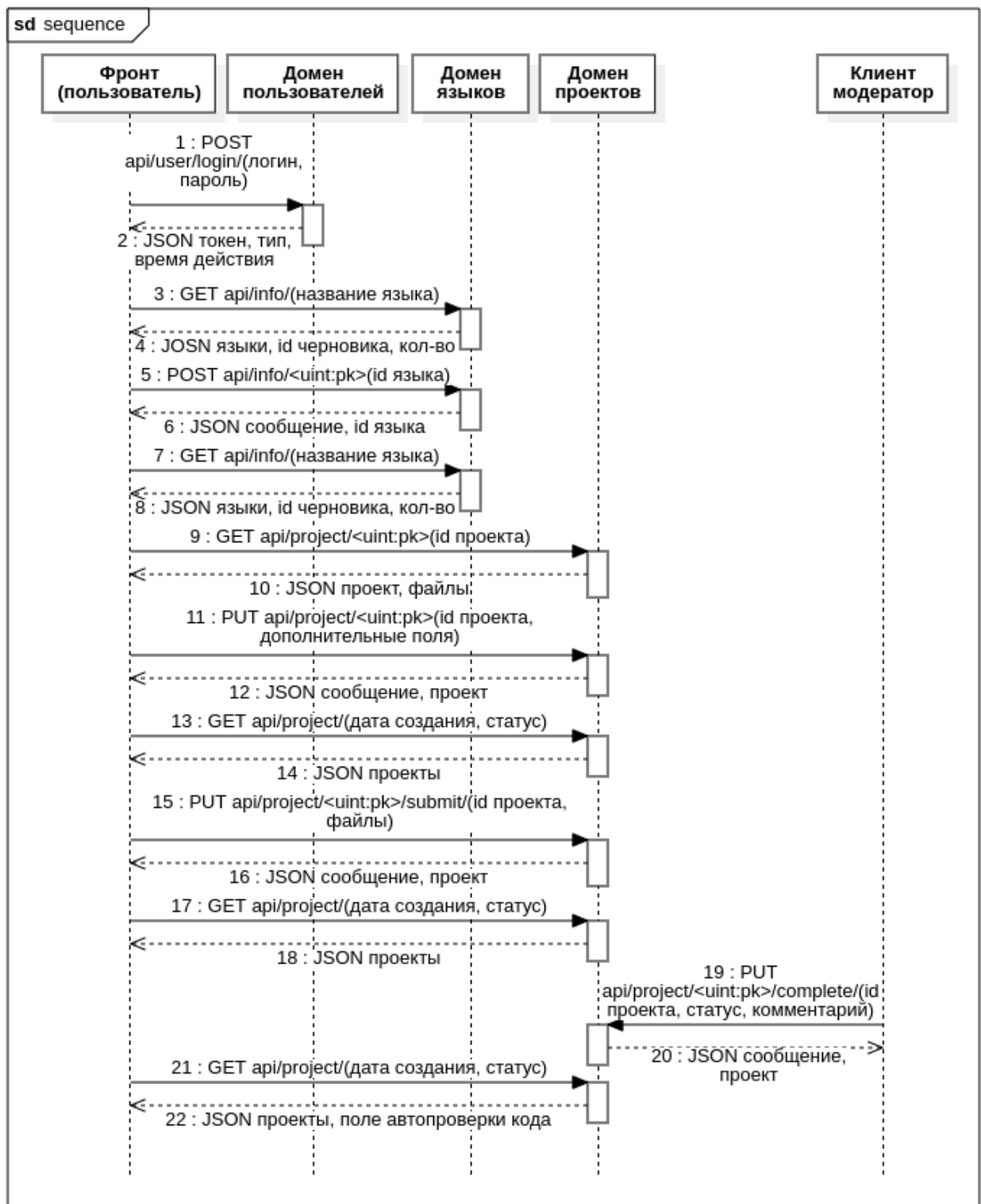


Рисунок 8 — Диаграмма последовательности

В начале бизнес-процесса происходит аутентификация пользователя. При помощи графического интерфейса студент или проверяющий отправляет запрос, передавая логин и пароль. Если аккаунт с указанными данными существует в базе, клиент получает информацию о пользователе и устанавливает Bearer token (идентификатор текущей сессии). Если учетные

данные неверны, возвращается ошибка, и пользователю необходимо либо зарегистрироваться, либо ввести корректный логин и пароль. На этом же этапе происходит проверка роли пользователя: гость, студент или проверяющий.

Графический интерфейс запрашивает у веб-сервиса список доступных языков программирования, которые возвращаются в формате JSON. Студент загружает файлы кода в черновик заявки, указывая язык программирования и код. Это действие можно повторять до завершения формирования заявки.

Когда студент определится с файлами и заполнит параметры проекта, он нажимает кнопку «Отправить» в графическом интерфейсе. После этого приложение отправляет на веб-сервис запрос на формирование заявки. После отправки студент может отслеживать статус проверки на соответствующей странице.

Процесс проверки заявок осуществляется через графический интерфейс. Проверяющие могут просматривать список всех заявок студентов, фильтровать их по имени студента, статусу и дате подачи, а также принимать или отклонять их с указанием причины. Проверяющие также могут редактировать список поддерживаемых языков программирования и управлять настройками системы. Для каждой из этих операций предусмотрены соответствующие методы REST API, отправляемые на веб-сервис и страницы веб-приложения.

## ОПИСАНИЕ ИНТЕРФЕЙСА

Главное меню приложения включает разделы, доступность которых зависит от роли пользователя.



Гость (рисунок 9) имеет доступ к страницам регистрации, входа и ознакомительной информации о сервисе.

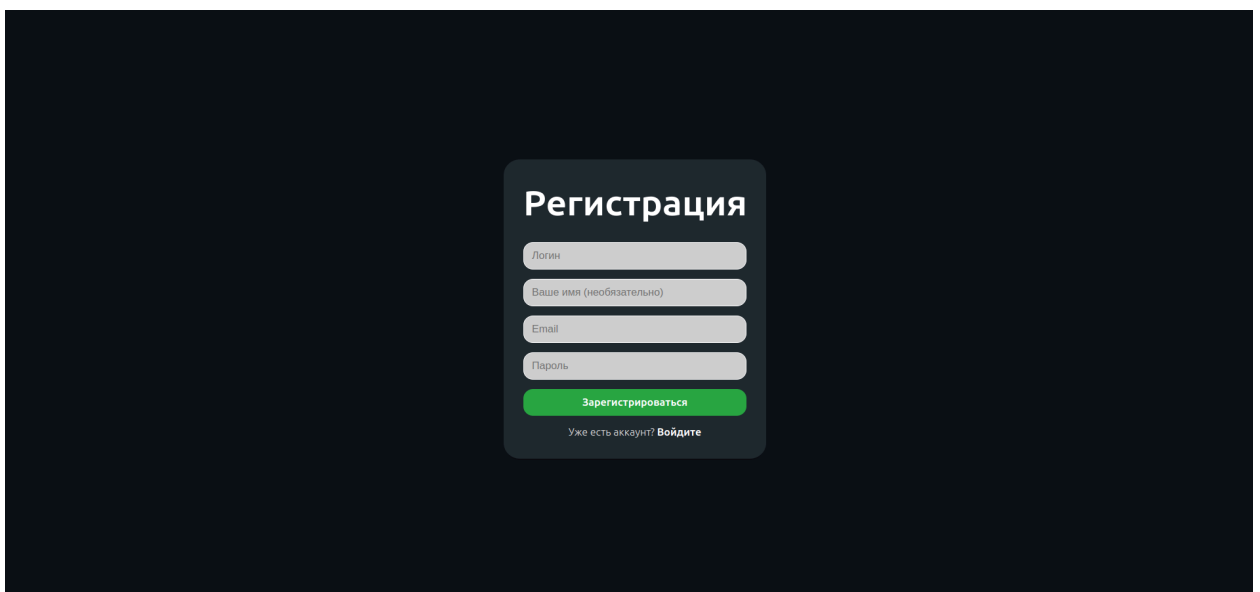


Студент (рисунок 10) может загружать файлы кода, создавать заявки на проверку, отслеживать их статус и просматривать историю проверок.



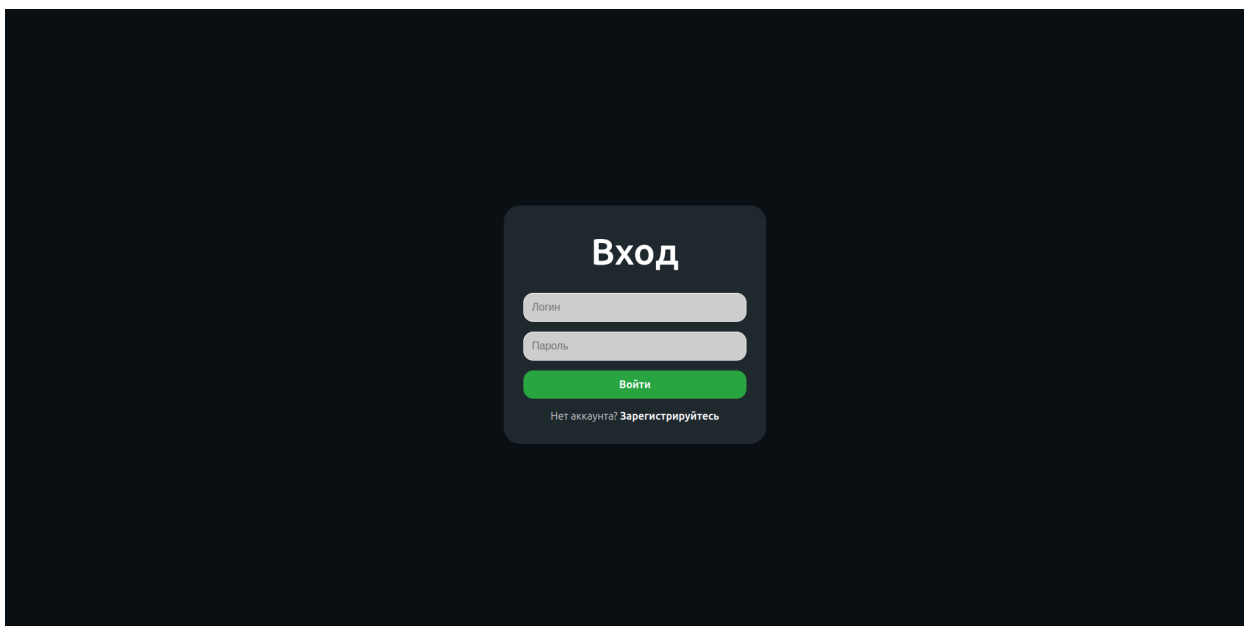
Проверяющий (рисунок 11) имеет доступ к управлению заявками, редактированию списка поддерживаемых языков программирования и управлению пользователями.

Регистрация и вход



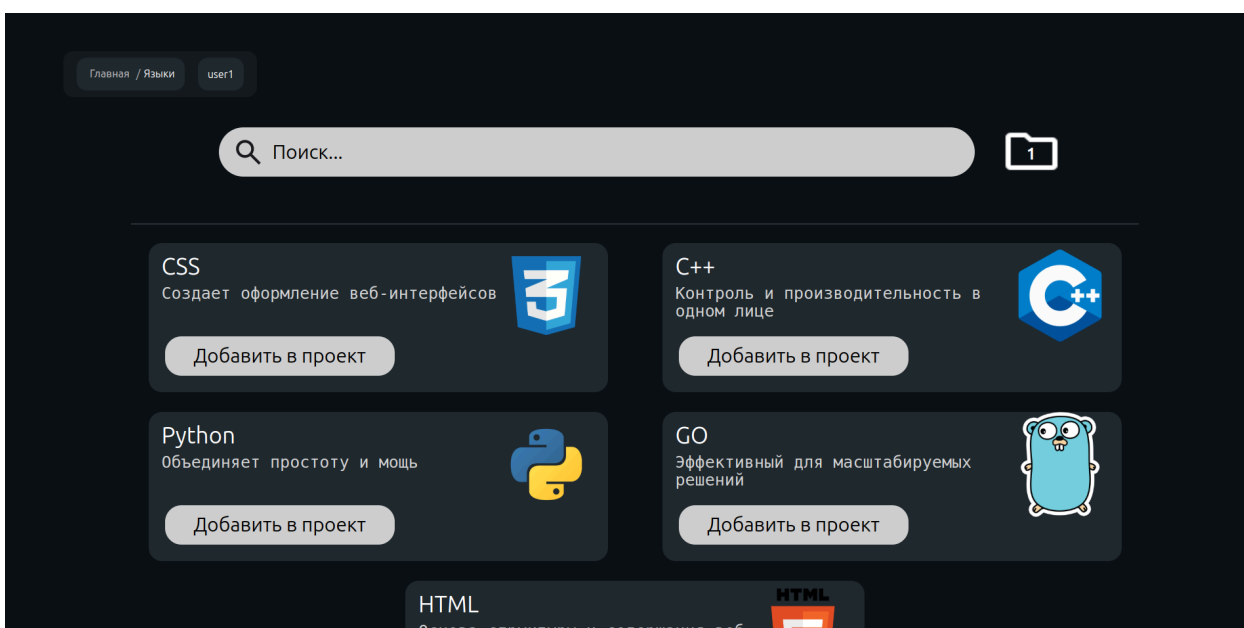
На странице регистрации (рисунок 12) отображается форма, позволяющая гостю создать новый аккаунт. После успешной регистрации пользователь попадает на главную страницу.



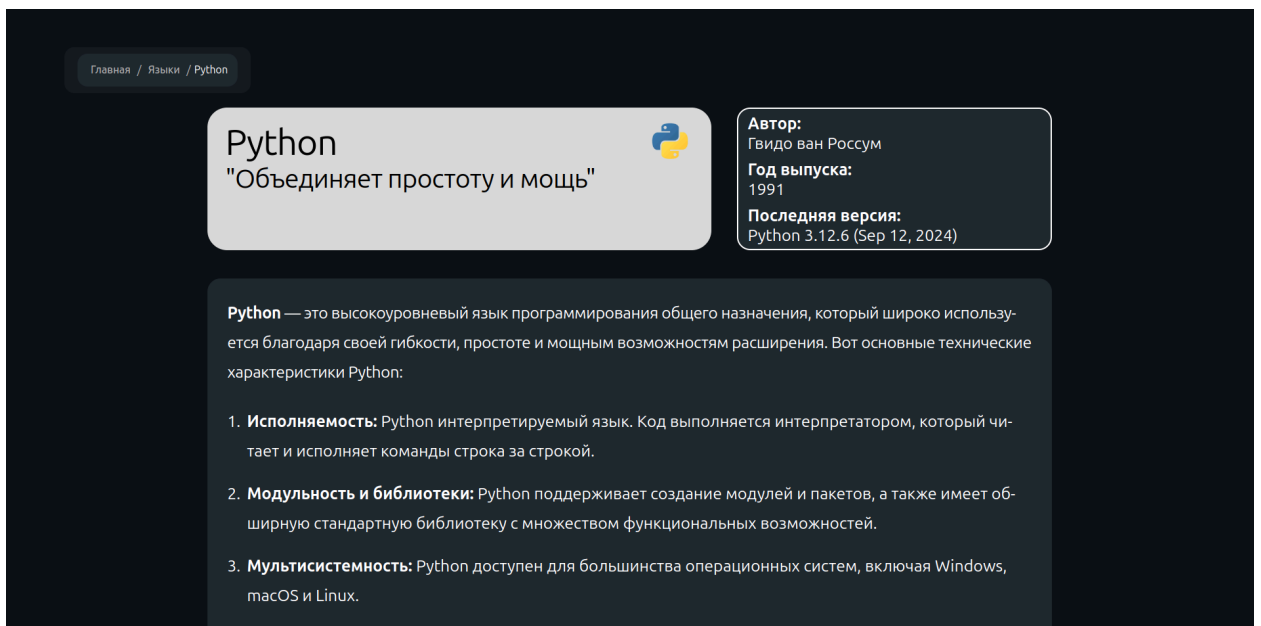


Форма входа (рисунок 13) позволяет зарегистрированному пользователю войти в аккаунт. При успешной авторизации устанавливается `session_id`, и пользователь перенаправляется на свою персонализированную страницу.

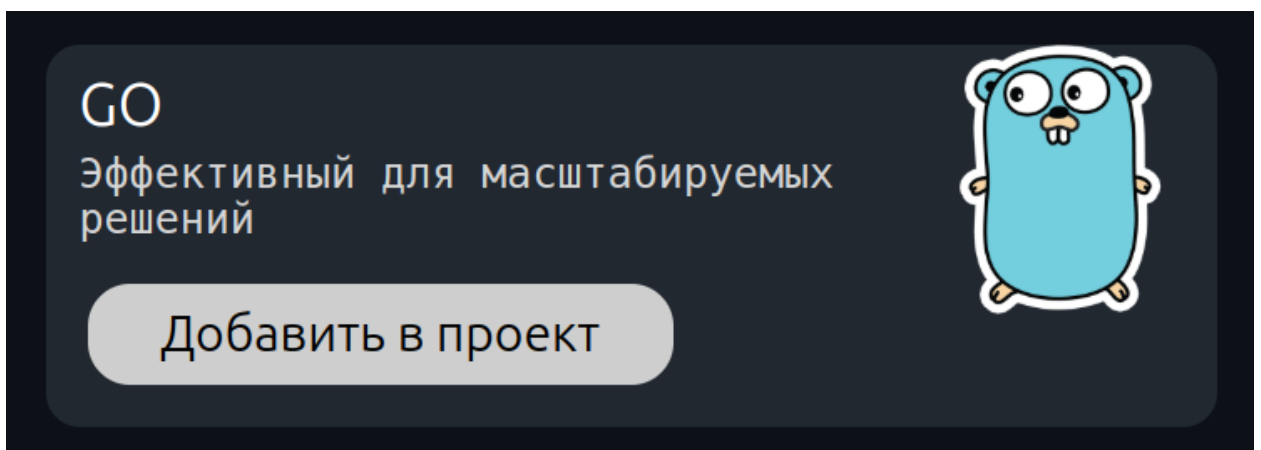
#### Работа с заявками



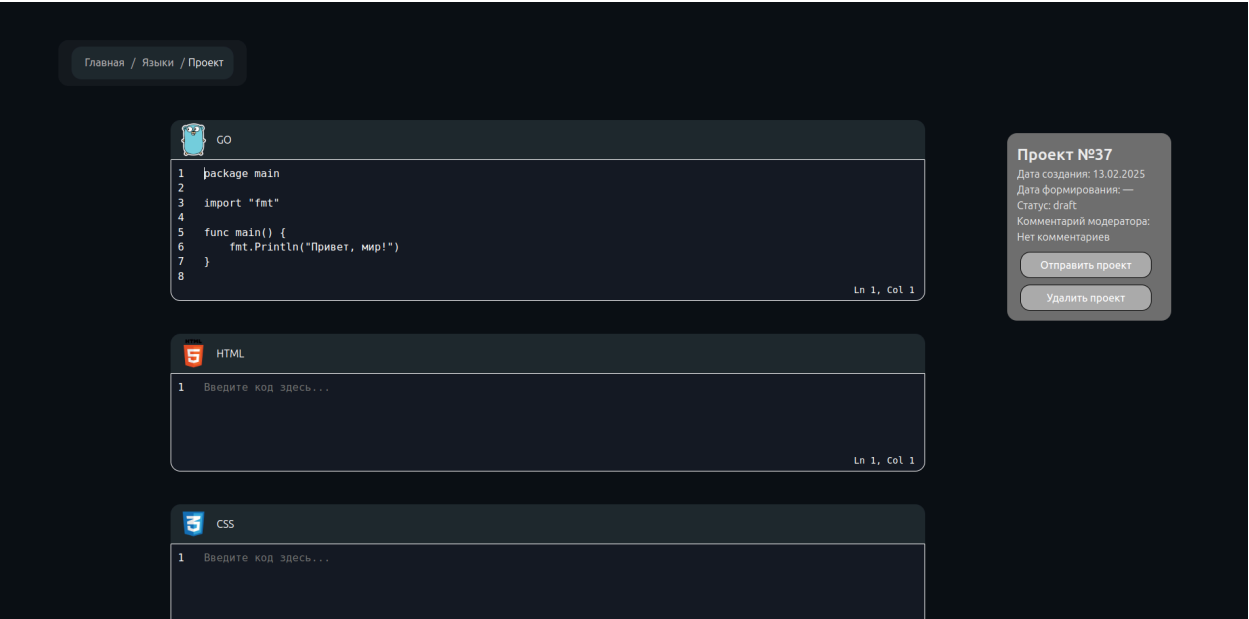
После входа студент попадает на страницу списка доступных языков программирования (рисунок 14). При выборе языка открывается страница загрузки файлов и оформления заявки.



На странице описания языка (рисунок 15) студенту доступна информация с описанием языка программирования.



В отличие от гостя, студенту доступна кнопка «Добавить файл» (рисунок 16), позволяющая прикрепить исходный код к заявке. Также он может просматривать загруженные файлы и редактировать черновик заявки.



Главная / Языки / Проекты

Список заявок

Дата начала:

дд.мм.гггг

Дата окончания:

дд.мм.гггг

Статус:

Все

Создатель:

user1

Дата создания	Дата формирования	Дата завершения	Создатель	Статус	Модератор	Комментарий	Ссылка	Оценка	QR
06.02.2025, 06:49	—	—	user1	created	Timofei Tsypyshev	test comment	Перейти	Изменить статус	
13.02.2025, 07:28	13.02.2025, 07:28	—	user1	created	—	—	Перейти	Изменить статус	
13.02.2025, 07:00	—	13.02.2025, 09:18	user1	completed	Timofei Tsypyshev	—	Перейти	Изменить статус	
13.02.2025, 13:13	—	13.02.2025, 13:15	user1	completed	Timofei Tsypyshev	123	Перейти	Изменить статус	
13.02.2025, 07:31	—	13.02.2025, 13:21	user1	rejected	Timofei Tsypyshev	1231231231	Перейти	Изменить статус	

Проверяющие могут принять или отклонить заявку, оставив комментарий (рисунок 19.2).

Управление профилем

Главная / Языки / Профиль

Личный кабинет

Имя:

Timofei Tsypyshev

Email:

timofeitsypyshev@yandex.ru

Новый пароль:

Введите новый пароль

Сохранить

Отмена





Пользователь может изменить пароль на странице личного кабинета (рисунок 20), доступной через меню.

Управление языками программирования

Главная / Языки / Редактирование

Список языков

Добавить язык

ID	Название	Автор	Год	Версия	Изображение	Действия
5	CSS	Габриел Маззоне	1996	CSS3 (2011)		<div>Редактировать</div> <div>Загрузить изображение</div> <div>Удалить</div>
2	C++	Бьерн Страуструп	1985	C++20 (Dec 2020)		<div>Редактировать</div> <div>Загрузить изображение</div> <div>Удалить</div>
1	Python	Гвидо ван Россум	1991	Python 3.12.6 (Sep 12, 2024)		<div>Редактировать</div> <div>Загрузить изображение</div> <div>Удалить</div>
3	GO	Роберт Гризмер, Роб Пайк	2009	Go 1.21.0 (Aug 2023)		<div>Редактировать</div> <div>Загрузить изображение</div>

Проверяющие могут редактировать список (рисунок 21) поддерживаемых языков программирования:

Главная / Языки / Редактирование

Добавить язык

Редактировать язык

C++

Контроль и производительность в одном лице

Это мощный и высокопроизводительный язык программирования, известный своей способностью обеспечивать низкоуровневый доступ к памяти и поддерживать сложные структуры данных. Вот основные технические характеристики C++

Бьерн Страуструп





1985

C++20 (Dec 2020)

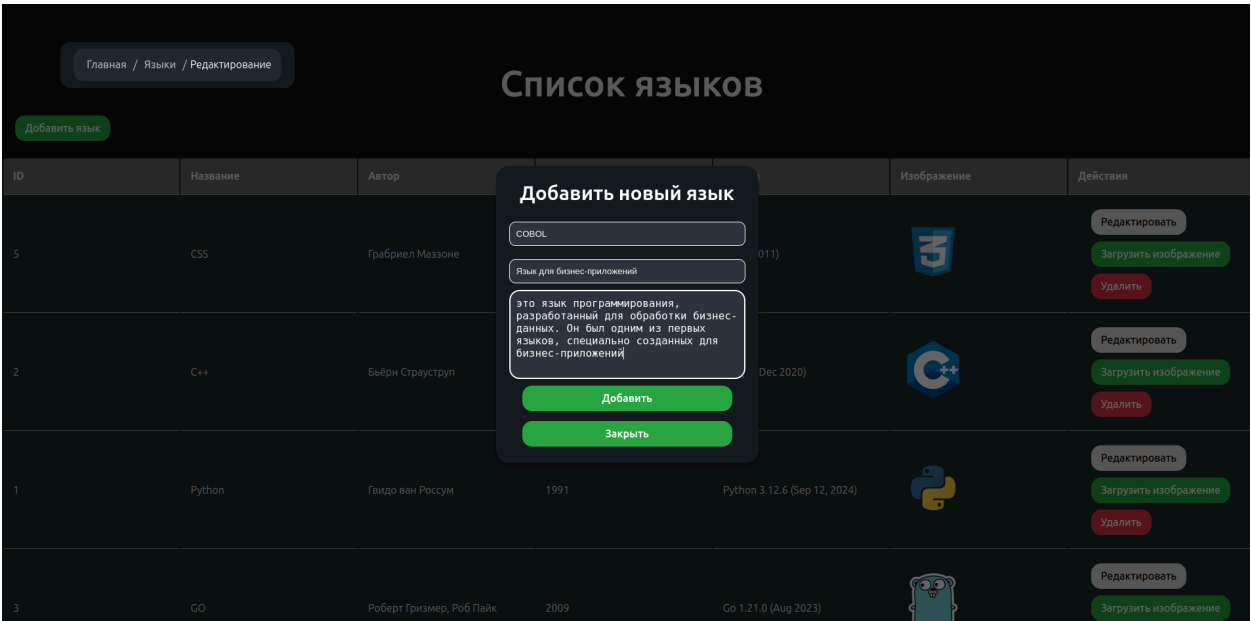
{ "Мультисистемность": "C++ доступен для множества операционных систем и платформ, что позволяет создавать"

Сохранить изменения

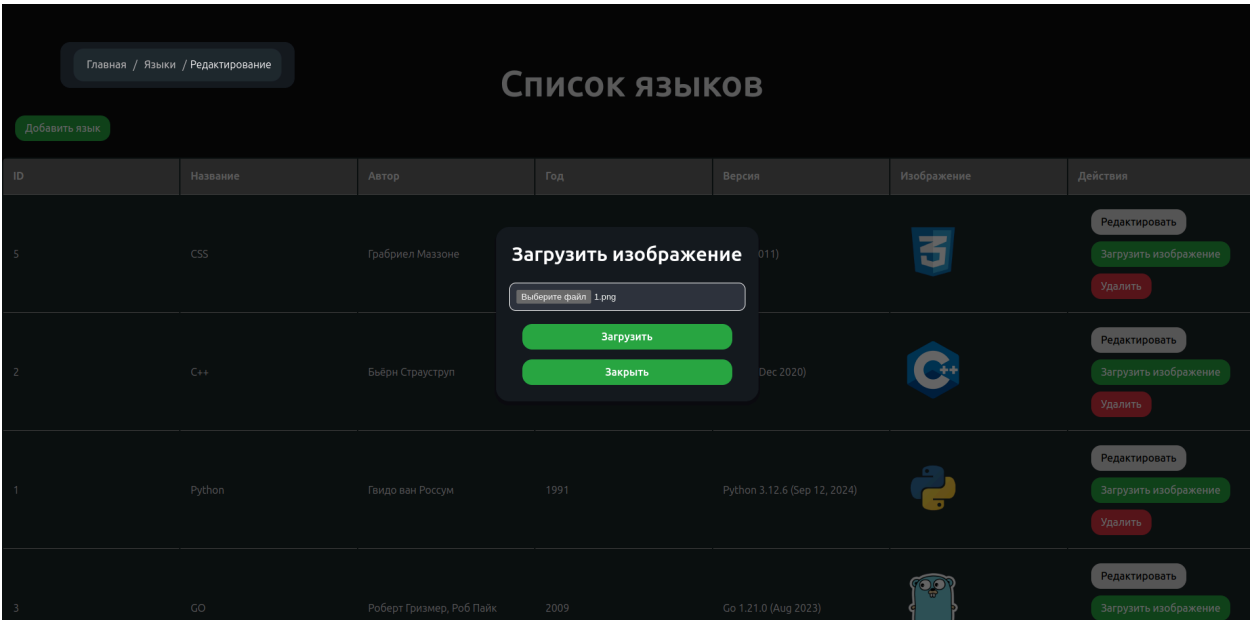
Закрыть

ID	Название	Автор	Год	Версия	Изображение	Действия
5	CSS	Габриел Маззоне	1996	CSS3 (2011)		<div>Редактировать</div> <div>Загрузить изображение</div> <div>Удалить</div>
2	C++	Бьерн Страуструп	1985	C++20 (Dec 2020)		<div>Редактировать</div> <div>Загрузить изображение</div> <div>Удалить</div>
1	Python	Гвидо ван Россум	1991	Python 3.12.6 (Sep 12, 2024)		<div>Редактировать</div> <div>Загрузить изображение</div> <div>Удалить</div>
3	GO	Роберт Гризмер, Роб Пайк	2009	Go 1.21.0 (Aug 2023)		<div>Редактировать</div> <div>Загрузить изображение</div>

Страница редактирования языков (рисунок 22) позволяет изменять параметры существующих языков.



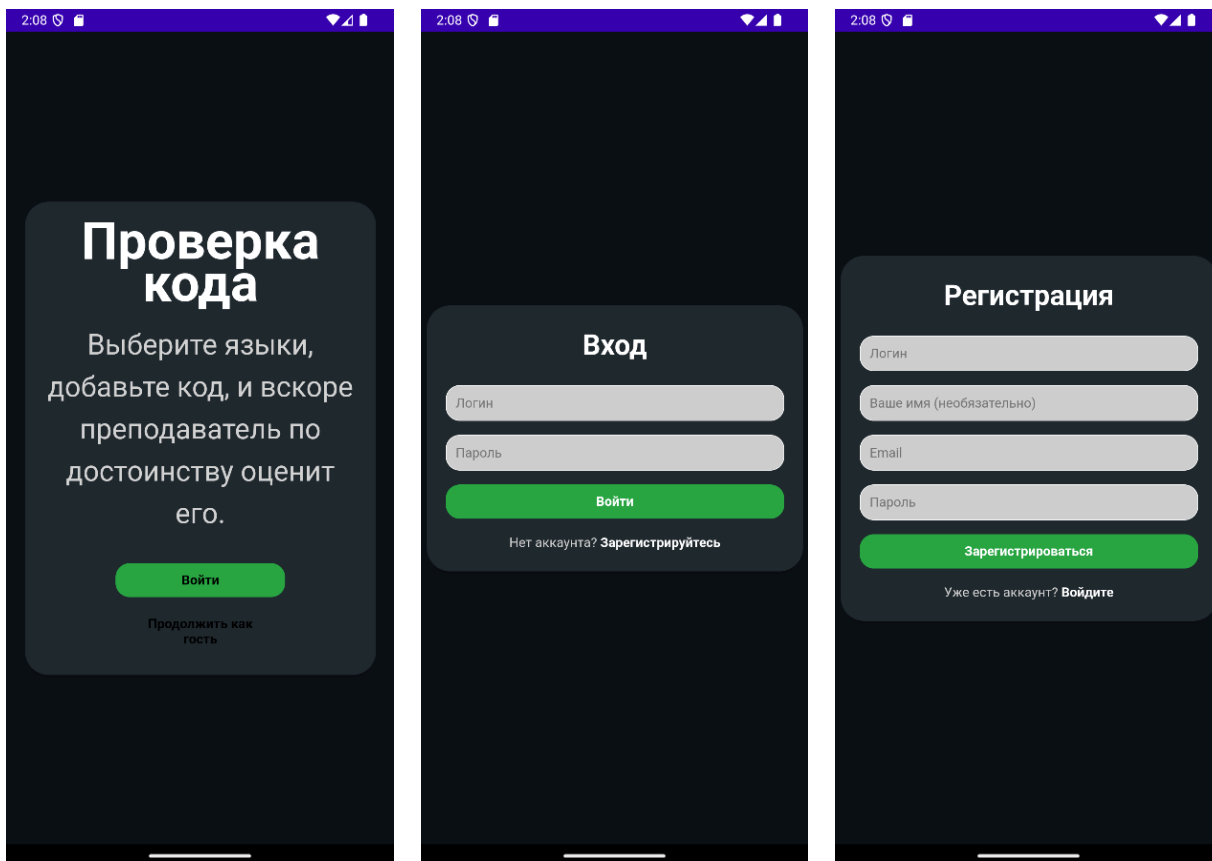
Создание нового языка (рисунок 23).



Загрузка новых изображений (рисунок 24)

Эта система обеспечивает удобный процесс проверки кода, автоматизируя взаимодействие между студентами и проверяющими.

Также было разработано приложения на Android и подключение к веб-сервису. Простое нативное приложение состоит из 9 страниц с фильтрацией и картинками. Приложение подключается к разработанному API через IP адрес в локальной сети. Интерфейс включает в себя все ранее описанные страницы для гостя и пользователя.



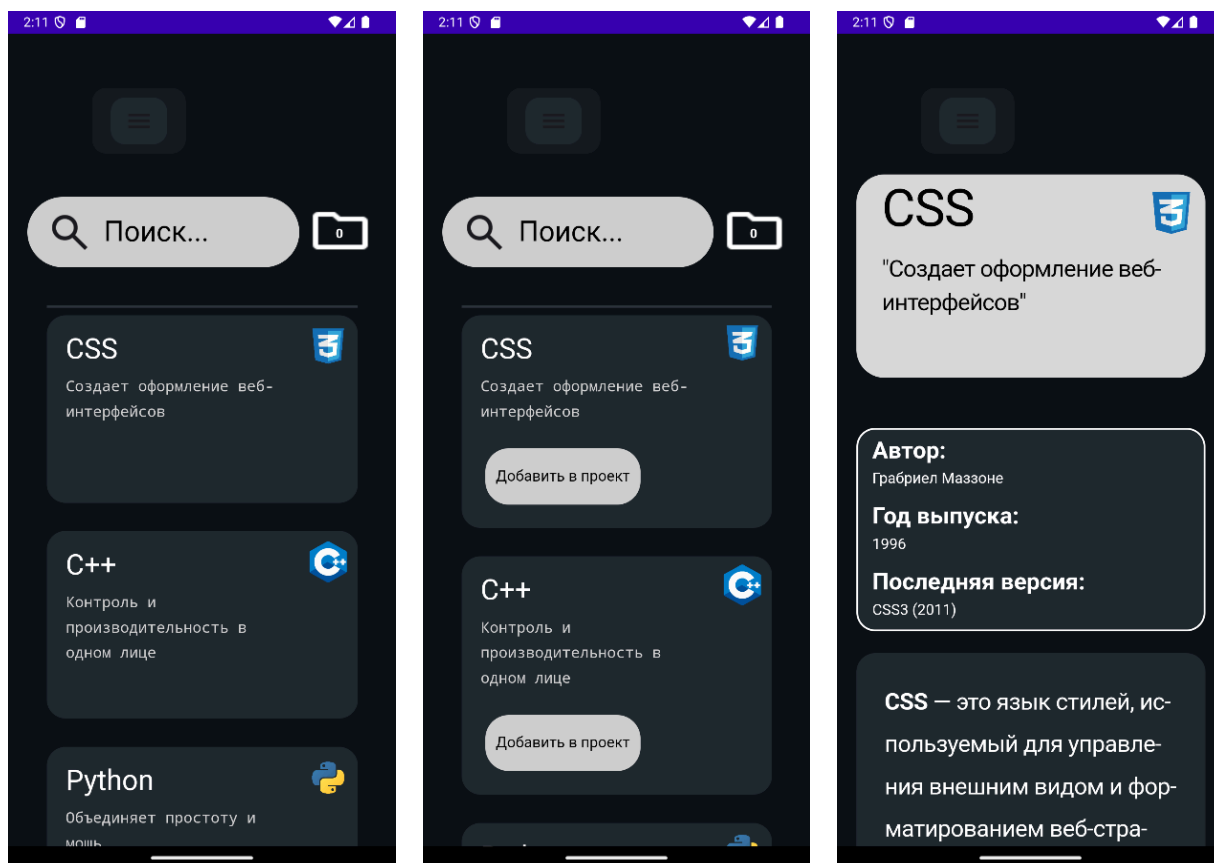
*а*

*б*

*в*

Рисунок 25 – Интерфейс домашней страницы

*а* – домашняя страница; *б* – страница авторизации; *в* – страница регистрации



*a*

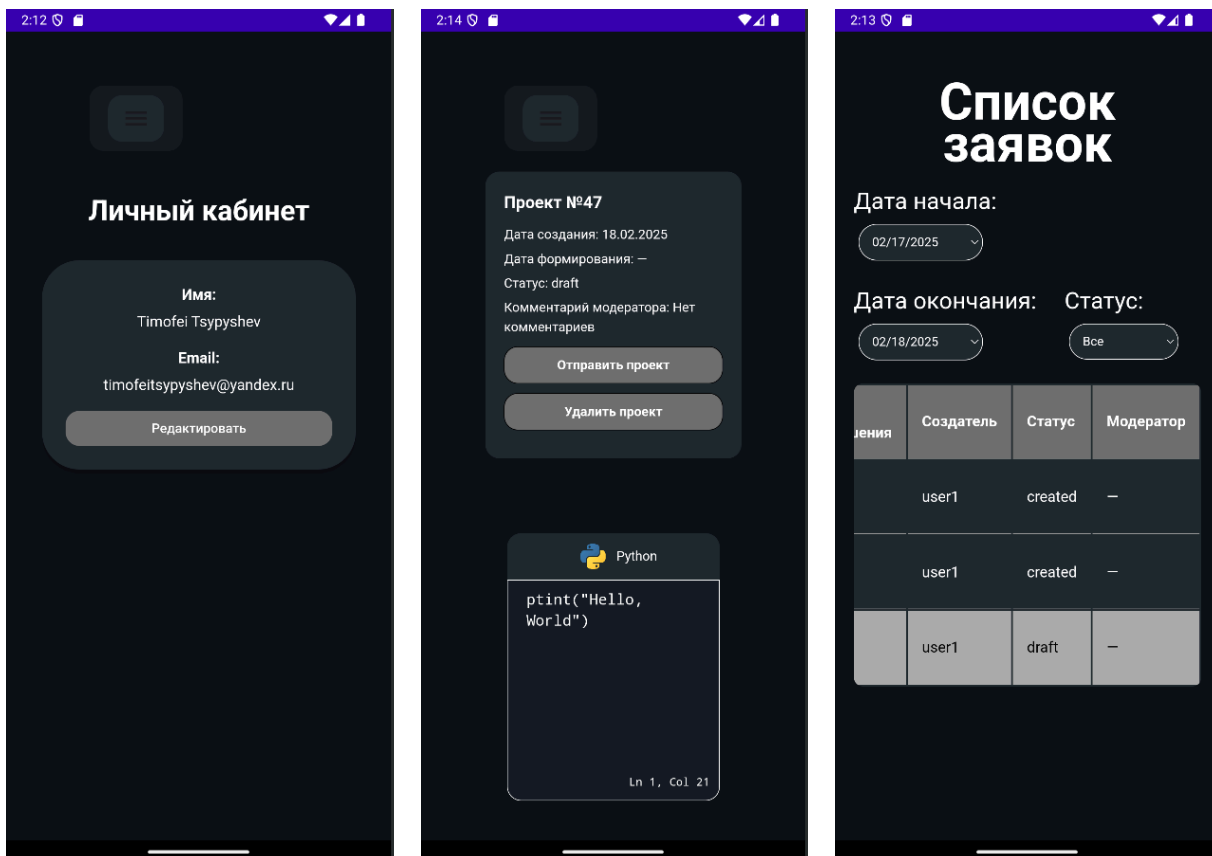
*б*

*в*

Рисунок 26 – Интерфейс языков программирования

*a* – страница со списком языков (гость); *б* – страница со списком языков (пользователь); *в* – страница с подробной информацией о языке





*а*

*б*

*в*

Рисунок 27 – Дополнительный интерфейс

*а* – страница профиля пользователя; *б* – страница проекта; *в* – страница с историей проектов

## ЗАКЛЮЧЕНИЕ

В ходе разработки системы для проверки студенческого кода с использованием различных языков программирования были достигнуты следующие результаты:

1. Дизайн приложения был разработан в Figma на основе шаблонов из github.com. После ознакомления с подходами в разработке бэкенда был создан MVP.
2. Проектирование и создание базы данных PostgreSQL, которая была успешно подключена к бэкенду системы.
3. Создан веб-сервис, реализующий всю бизнес-логику работы с заявками, кроме авторизации, для использования в SPA (Single Page Application).
4. В веб-сервис была добавлена авторизация, с использованием Redis для хранения сессий, а также внедрен Swagger для удобного тестирования API. Исходный код бэкенда был сохранён по ссылке: <https://github.com/ttsypyshev/code-inspector-back>
5. Базовый интерфейс приложения для гостя был разработан с использованием React.
6. Внедрен менеджер состояний Redux Toolkit для хранения значений фильтров, добавлена адаптивность интерфейса и поддержка PWA (Progressive Web App).
7. Завершена разработка интерфейса пользователя в React, для взаимодействия с веб-сервисом использован Axios для отправки запросов.
8. Реализован React интерфейс издателя, внедрен Real-time web, что позволило улучшить взаимодействие с системой в реальном времени.
9. Разработано десктопное приложение на Tauri, что расширяет возможности использования приложения.
10. Приложение было развернуто с использованием GitHub Pages и доступно по ссылке: <https://ttsypyshev.github.io/code-inspector-front/>.

11. Подготовлен набор документации, включающий РПЗ (расчетно-пояснительную записку), ТЗ (техническое задание) и набор диаграмм для визуализации системы.

12. Оформлен git-репозиторий на сервисе GitHub, содержащий исходный код проекта: <https://github.com/ttsypyshev/code-inspector-front>.

## **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. GitHub – платформа для управления версиями и совместной разработки. [Электронный ресурс]. URL: <https://docs.github.com/> (дата обращения: 12.02.2025).
2. Документация по языку программирования Go [Электронный ресурс]. URL: <https://go.dev/doc/> (дата обращения: 10.07.2024).
3. Документация по разработке REST API на Golang [Электронный ресурс]. URL: <https://go.dev/doc/tutorial/web-service-gin> (дата обращения: 15.08.2024).
4. Документация MinIO – объектное хранилище S3-совместимого типа. [Электронный ресурс]. URL: <https://min.io/docs/> (дата обращения: 28.08.2024).
5. Документация по Redis – система управления кешем и хранилище данных в памяти. [Электронный ресурс]. URL: <https://redis.io/docs/> (дата обращения: 05.09.2024).
6. Документация PostgreSQL – реляционная СУБД. [Электронный ресурс]. URL: <https://www.postgresql.org/docs/> (дата обращения: 12.09.2024).
7. Документация по Tauri – инструмент для создания кроссплатформенных десктопных приложений. [Электронный ресурс]. URL: <https://tauri.app/v2/guides/> (дата обращения: 14.12.2024).
8. Документация по React – библиотека для построения пользовательских интерфейсов. [Электронный ресурс]. URL: <https://react.dev/learn> (дата обращения: 25.01.2025).

**ПРИЛОЖЕНИЕ А ТЕХНИЧЕСКОЕ ЗАДАНИЕ**  
**МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**  
**им. Н.Э. Баумана**

---

Кафедра «Системы обработки информации и управления»

Утверждаю  
Заведующий кафедрой ИУ-5  
\_\_\_\_\_ В.И.Терехов  
" \_ " \_\_\_\_\_ 2024 г.

Согласовано  
Научный руководитель  
\_\_\_\_\_ А.И. Канев  
" \_ " \_\_\_\_\_ 2024 г.

**Проверка кода студентов**

Техническое задание  
(вид документа)

писчая бумага  
(вид носителя)

20  
(количество листов)

ИСПОЛНИТЕЛЬ:

\_\_\_\_\_ Цыпышев Тимофей Александрович  
" \_ " \_\_\_\_\_ 2024 г.

Москва - 2024

## **1. Введение**

Цель разработки заключается в создании универсальной платформы, которая обеспечивает удобный и эффективный процесс проверки, анализа и оптимизации студенческого кода преподавателями.

## **2. Назначение разработки**

Назначение системы заключается в создании платформы для организации эффективной проверки и анализа студенческого кода. Студенты могут создавать заявки, в которых указывают, на каком языке программирования написан код, задачу и требования для проверки. Преподаватели получают доступ к списку заявок, проводят проверку и добавляют комментарии или рекомендации. После анализа преподаватель отмечает заявку как выполненную, а студент может просмотреть подробный отчет с результатами.

## **3. Стадии и этапы разработки**

- 3.1. Создание MVP и базового дизайна на основе [github.com](https://github.com);
- 3.2. Реализовать хранение данных в БД (Postgres);
- 3.3. Создать веб-сервис в бэкенде на Golang (DRF);
- 3.4. Реализовать авторизацию и хранение сессий в Redis;
- 3.5. Разработать базовый SPA на React для гостя;
- 3.6. Внедрение адаптивности, разработка Tauri приложения;
- 3.7. Интерфейс студента на React с менеджером состояний Redux Toolkit;
- 3.8. Реализовать интерфейс преподавателя в React;
- 3.9. Реализовать нативное приложение;
- 3.10. Развернуть веб-приложение React на Github Pages;
- 3.11. Подготовка документацию к системе (РПЗ, ТЗ, набор диаграмм);
- 3.12. Создать репозиторий Git для системы.

## **4. Требования к функциональным характеристикам**

### **4.1. Список HTTP методов**

- 4.1.1. GET Список языков
- 4.1.2. GET Информация об одном языке
- 4.1.3. POST Добавление нового языка
- 4.1.4. PUT Изменение информации о языке
- 4.1.5. POST Добавление или изменение изображения для языка
- 4.1.6. DELETE Удаление языка
- 4.1.7. POST Добавление языка в заявку-черновик
- 4.1.8. GET Список проектов пользователя
- 4.1.9. GET Информация о конкретном проекте
- 4.1.10. PUT Изменение полей проекта
- 4.1.11. PUT Формирование заявки студентом
- 4.1.12. PUT Завершение или отклонение заявки преподавателем
- 4.1.13. DELETE Удаление проекта
- 4.1.14. DELETE Удаление файла из проекта
- 4.1.15. PUT Изменение информации о файле
- 4.1.16. POST Регистрация нового пользователя
- 4.1.17. PUT Изменение информации о пользователе
- 4.1.18. POST Аутентификация пользователя
- 4.1.19. POST Деавторизация пользователя

### **4.2. Меню**

- 4.2.1. Главная - перенаправляет на страницу 4.6
- 4.2.2. Список языков программирования – перенаправляет на страницу 4.7 (вызывается метод 4.1.1)
- 4.2.3. Список проектов – перенаправляет на страницу 4.9
- 4.2.4. Редактирование языков программирования – перенаправляет на страницу 4.12, доступно только преподавателям

4.2.5. Регистрация – перенаправляет на страницу 4.3 появляется только для гостей

4.2.6. Личный кабинет – перенаправляет на страницу 4.10, только аутентифицированным пользователям

4.2.7. Войти – перенаправляет на страницу 4.4, только для гостей, появляется только для гостей

4.2.8. Выйти – перенаправляет на страницу 4.6 (вызывается метод 4.1.19), доступно только аутентифицированным пользователям

#### 4.3.Регистрация

4.3.1. Доступно только гостям

4.3.2. Отображает форму регистрации

4.3.2.1. Поле имени пользователя

4.3.2.2. Поле почты

4.3.2.3. Поле короткого имени

4.3.2.4. Поле пароля

4.3.3. Действия

4.3.3.1. Регистрация пользователя – (вызывается метод 4.1.16)

4.3.3.2. Вернуться к аутентификации – перенаправляет на страницу 4.4

#### 4.4.Аутентификация

4.4.1. Доступно только гостям

4.4.2. Отображает форму аутентификации

4.4.2.1. Поле короткого имени

4.4.2.2. Поле пароля

4.4.3. Действия

4.4.3.1. Войти – (вызывается метод 4.1.18)

4.4.3.2. Регистрация – перенаправляет на страницу 4.3

#### 4.5.Личный кабинет

4.5.1. Доступно пользователю

4.5.2. Действия



4.5.2.1. Изменить данные пользователя – (вызывается метод 4.1.17)

#### 4.6. Главная

4.6.1. Доступна всем

4.6.2. Отображается статическая информация

4.6.2.1. Назначение сервиса

4.6.2.2. Контакты для связи

#### 4.7. Список языков программирования

4.7.1. Доступна всем

4.7.2. Отображаются элементы карточек с языками программирования, вызывается метод 4.1.1

4.7.2.1. Название языков программирования

4.7.2.2. Краткое описание

4.7.3. Действия

4.7.3.1. Поиск – перенаправляет на страницу 4.7, (используется метод 4.1.1), с фильтрующим параметром

4.7.3.2. Подробнее – перенаправляет на страницу 4.8 (используется метод 4.1.2)

4.7.3.3. Добавить в проект – добавляет язык в проект-черновик, (вызывается метод 4.1.7), только пользователи.

4.7.3.4. Кнопка корзины – перенаправляет на страницу 4.9, только пользователи.

#### 4.8. Подробнее о языке программирования

4.8.1. Доступна всем

4.8.2. Отображается подробная информация выбранного языка, (вызывается метод 4.1.2)

#### 4.9. Проект

4.9.1. Доступно только пользователям

4.9.2. Отображает текущий проект-черновик пользователя, (метод 4.1.9)

4.9.2.1. Список выбранных языков

4.9.2.2. Дата создания проекта

- 4.9.2.3. Дата последнего сохранения проекта
- 4.9.2.4. Поле ввода кода
- 4.9.2.5. Комментарии к каждому файлу
- 4.9.3. Действия, доступны только в случае, если статус «черновик»
  - 4.9.3.1. Сохранить – сохраняет текущий проект-черновик, (вызывается метод 4.1.11)
  - 4.9.3.2. Удалить – удаляет проект-черновик, (вызывается метод 4.1.13)
- 4.10. Личный кабинет
  - 4.10.1. Доступно пользователю
  - 4.10.2. Отображается список проектов (метод 4.1.8)
    - 4.10.2.1. Только проекты, созданные данным студентом
    - 4.10.2.2. Только проекты, к которым данный преподаватель закреплён как проверяющий
  - 4.10.3. Действия
    - 4.10.3.1. Фильтрация – фильтрует проекты по дате создания или статусу, вызывается (метод 4.1.8)
    - 4.10.3.2. Принять – сохраняет положительное решение по проекту, выполняется метод 4.1.12, доступно только преподавателю
    - 4.10.3.3. Отклонить – сохраняет отрицательное решение по проекту, вызывается метод 4.1.12, доступно только преподавателю
    - 4.10.3.4. Посмотреть подробную информацию о проекте – перенаправляет на страницу 4.9 (вызывается метод 4.1.9)
- 4.11. Список языков программирования
  - 4.11.1. Доступно преподавателю
  - 4.11.2. Отображаются все языки программирования (вызывается метод 4.1.1)
  - 4.11.3. Действия
    - 4.11.3.1. Удалить – удаляет язык (вызывается метод 4.1.6)
    - 4.11.3.2. Редактирование/создание – переход на страницу 4.12

#### 4.12. Редактирование/создание языка программирования

4.12.1. Доступно преподавателю

4.12.2. Отображается информация об изменяемом/добавляем языке  
(вызывается метод 4.1.3)

4.12.2.1. Название

4.12.2.2. Короткое описание

4.12.2.3. Подробное описание

4.12.2.4. Автор языка

4.12.2.5. Год выпуска

4.12.2.6. Актуальная версия

4.12.2.7. Список особенностей языка

4.12.3. Действия

4.12.3.1. Сохранить – если добавляет новый язык, то вызывается метод, 4.1.3, иначе если изменяется существующий, то вызывается метод 4.1.4

4.12.3.2. Картинка – добавляет/изменяет картинку языка программирования (вызывается метод 4.1.5)

#### 4.13. 404

4.13.1. Доступно всем

4.13.2. Отображается в случае отсутствия ресурса

#### 4.14. 403

4.14.1. Доступно всем

4.14.2. Отображается в случае запрета на использование ресурса

#### 4.15. 401

4.15.1. Доступно только неавторизованным пользователям

4.15.2. Отображается в отсутствия/недостатка прав на использования ресурса

### 5. Требования к составу и параметрам технических средств

#### 5.1. Сервер

5.1.1. Процессор минимум 2-ядерный с частотой от 2 ГГц.

5.1.2. Оперативная память от 4 Гб.

5.1.3. Место на жестком диске от 2 Гб.

5.2. Клиентская часть

5.2.1. Процессор с частотой от 2 ГГц.

5.2.2. Оперативная память от 4 Гб.

**6. Требования к информационной и программной совместимости**

6.1. Серверная часть

6.1.1. Операционная система Linux (версия ядра 5.10 и выше)

6.1.2. Minio (RELEASE.2022-10-15T19-57-03Z и выше)

6.1.3. Redis (6.2.16 и выше)

6.1.4. Golang (1.21.13 и выше)

6.1.5. СУБД PostgreSQL (12.2 и выше)

6.1.6. Node JS

6.1.7. Docker

6.2. Веб-браузер (любой из)

6.2.1. Chrome (119.0.6045 и выше)

6.2.2. Opera (105.0.4970.16 и выше)

6.2.3. Mozilla Firefox (121.0 и выше)

## ПРИЛОЖЕНИЕ Б СПИСОК HTTP МЕТОДОВ

Таблица 1 – HTTP методы разрабатываемого веб-сервиса

№	Метод	URL	Описание	Входные данные	Выходные данные
4.1.1	GET	/api/info	Список языков	langname=string	{ count: int, draftID: int, langs: [ { ID: uint, Name: string, ShortDescription: string, Description: string, ImgLink: string, Author: string, Year: string, Version: string, List: {string: string}, Status: bool } ] }

4.1.2	GET	/api/info/{pk}	Информация об одном языке	pk:uint	<pre> {   info: {     ID: uint,     Name: string,     ShortDescription: string,     Description: string,     ImgLink: string,     Author: string,     Year: string,     Version: string,     List: {string: string},     Status: bool   } } </pre>
4.1.3	POST	/api/info	Добавление нового языка	<pre> {   Name: string   ShortDescription: string   Description: string   Author: string   Year: string   Version: string } </pre>	<pre> {   message: string,   serviceID: int } </pre>

				List: {string:string} }	
4.1.4	PUT	/api/info/{pk}	Изменение информации о языке	pk:uint { Name: string ShortDescription: string Description: string Author: string Year: string Version: string List: {string:string} }	{ message: string, service: { ID: uint, Name: string, ShortDescription: string, Description: string, ImgLink: string, Author: string, Year: string, Version: string, List: {string: string}, Status: bool } }
4.1.5	POST	/api/info/{pk}	Добавление или изменение изображения для языка	pk:uint	{ message: string, service: {

				<pre>{   file: Image }</pre>	<pre>ID: uint, Name: string, ShortDescription: string, Description: string, ImgLink: string, Author: string, Year: string, Version: string, List: {string: string}, Status: bool } }</pre>
4.1.6	DELETE	/api/info/{pk}	Удаление языка	pk:uint	<pre>{   message: string,   status: bool }</pre>
4.1.7	POST	/api/info/add-service	Добавление языка в заявку-черновик	<pre>{   IDLang: uint }</pre>	<pre>{   message: string,   projectID: int }</pre>



4.1.8	GET	/api/project	Список проектов пользователя	start_date=string end_date=string status=string	{ projects: [ { ID: uint, UserID: string, User: { id: string, name: string, email: string, login: string, role: string }, CreationTime: string, FormationTime: string, CompletionTime: string, Status: string, ModeratorID: string, Moderator: { id: string, name: string, email: string, }, }, ], }
-------	-----	--------------	---------------------------------	---	--

					<pre> login: string, role: string }, ModeratorComment: string, Count: int } ] } </pre>
4.1.9	GET	/api/project/{pk}	Информация о конкретном проекте	pk:uint	<pre> {   files: [     {       ID: uint,       LangID: uint,       Lang: {         ID: uint,         Name: string,         ShortDescription: string,         Description: string,         ImgLink: string,         Author: string,         Year: string, </pre>

					<pre>Version: string, List: {string: string}, Status: bool }, ProjectID: uint, Code: string, FileName: string, FileSize: int, Comment: string, AutoCheck: int } ], project: {   ID: uint,   UserID: string,   CreationTime: string,   FormationTime: string,   CompletionTime: string,   Status: string,   ModeratorID: string,   ModeratorComment: string,</pre>
--	--	--	--	--	---

					Count: int } }
4.1.10	PUT	/api/project/{pk}	Изменение полей проекта	pk:uint { status: string comment: string } 	{ message: string, project: { ID: uint, UserID: string, CreationTime: string, FormationTime: string, CompletionTime: string, Status: string, ModeratorID: string, ModeratorComment: string, Count: int } } 
4.1.11	PUT	/api/project/{pk}/submit	Формирование заявки студентом	pk:uint { FileCodes: {uint:string} 	{ message: string, project: { 

				}	ID: uint, UserID: string, CreationTime: string, FormationTime: string, CompletionTime: string, Status: string, ModeratorID: string, ModeratorComment: string, Count: int  }
4.1.12	PUT	/api/project/{pk}/complete	Завершение или отклонение заявки преподавателем	pk:uint { status: string comment: string }	{ message: string, project: { ID: uint, UserID: string, CreationTime: string, FormationTime: string, CompletionTime: string, Status: string, ModeratorID: string, }

					ModeratorComment: string, Count: int } }
4.1.13	DELETE	/api/project/{pk}/complete	Удаление проекта	pk:uint { FileCodes: {uint:string} }	{ message: string, status: bool }
4.1.14	DELETE	/api/file/delete	Удаление файла из проекта	{ IDProject: uint IDLang: uint }	{ message: string, status: bool }
4.1.15	PUT	/api/file/update	Изменение информации о файле	{ IDProject: uint IDLang: uint Code: string FileName: string Comment: string }	{ file: { ID: uint, LangID: uint, ProjectID: uint, Code: string, FileName: string, FileSize: int, } }

					Comment: string, AutoCheck: int }, message: string }
4.1.16	POST	/api/user/register	Регистрация нового пользователя	{ Name: string Email: string Login: string Password: string }	{ message: string, userID: string }
4.1.17	PUT	/api/user/update	Изменение информации о пользователе	{ Name: string Email: string Password: string }	{ message: string, user: { id: string, name: string, email: string, login: string, role: string } }

					}
4.1.18	POST	/api/user/update	Аутентификация пользователя	{ Login: string Password: string }	{ expires_in: int, access_token: string, token_type: string }
4.1.19	POST	/api/user/logout	Деавторизация пользователя		{ message: string, status: bool }