|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | **Министерство науки и высшего образования Российской Федерации**  Калужский филиал  федерального государственного бюджетного  образовательного учреждения высшего образования  ***«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»***  ***(КФ МГТУ им. Н.Э. Баумана)*** |

|  |  |
| --- | --- |
| **ФАКУЛЬТЕТ** | **ИУК «Информатика и управление»** |
| **КАФЕДРА** | ***ИУК4 «Программное обеспечение ЭВМ, информационные*** |
| ***технологии»*** | |

**РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**к курсовому проекту на тему:**

|  |
| --- |
| **Разработка личного кабинета пользователя системы документирования** |
| **сети** |

|  |  |
| --- | --- |
| по дисциплине | Объектно-ориентированное программирование |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Студент гр. | ИУК4-52Б | |  |  | ( | Калашников А.С. | ) |
|  | |  |  | (подпись) |  | (Ф.И.О.) |  |
| Руководитель | |  |  |  | ( | Глебов С.А. | ) |
|  | |  |  | (подпись) |  | (Ф.И.О.) |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Оценка руководителя |  | баллов |  |  |
|  | 30-50 |  | (дата) |  |
| Оценка защиты |  | баллов |  |  |
|  | 30-50 |  | (дата) |  |
| Оценка проекта |  | баллов |  | |
|  |  |  | (оценка по пятибалльной шкале) | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Комиссия: |  |  | ( | Глебов С.А. | ) |
|  |  | (подпись) |  | (Ф.И.О.) |  |
|  |  |  | ( | Гришунов С.С. | ) |
|  |  | (подпись) |  | (Ф.И.О.) |  |
|  |  |  | ( | Гагарин Ю.Е. | ) |
|  |  | (подпись) |  | (Ф.И.О.) |  |

Калуга, 2022

**Калужский филиал   
федерального государственного бюджетного образовательного учреждения высшего образования**

***«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»   
(КФ МГТУ им. Н.Э. Баумана)***

УТВЕРЖДАЮ

Заведующий кафедрой \_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_ ( \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ )

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_г.

**ЗАДАНИЕ**

**на выполнение курсового проекта**

|  |  |
| --- | --- |
| по дисциплине | Объектно-ориентированное программирование |

|  |  |  |
| --- | --- | --- |
| Студент | Калашников А.С. ИУК4-52Б | |
|  | | (фамилия, инициалы, индекс группы) |
| Руководитель | | Глебов С.А. |
|  | | (фамилия, инициалы) |

График выполнения проекта: 25% к 4 нед., 50% к 7 нед., 75% к 10 нед., 100% к 16 нед.

|  |
| --- |
| ***1. Тема курсового проекта*** |
| Разработка личного кабинета пользователя системы документирования сети |
|  |
|  |
| ***2. Техническое задание*** |
| Разработать личный кабинет пользователя системы документирования сети в виде |
| веб-приложения |
|  |

|  |
| --- |
| ***3. Оформление курсового проекта*** |
| 3.1. Расчетно-пояснительная записка на 40 листах формата А4. |
| 3.2. Перечень графического материала КП (плакаты, схемы, чертежи и т.п.) |
| 1. *Диаграмма классов* |
| 1. *Демонстрационный чертеж* |
| 1. *Демонстрационный чертеж* |
|  |

Дата выдачи задания «09» сентября 2022 г.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Руководитель курсового проекта | | |  | | / | Глебов С.А |
|  | | (подпись) | | |  | (Ф.И.О.) |
| Задание получил |  | | / | Калашников А.С. | / | «09» сентября 2022 г. |
|  | (подпись) | |  | (Ф.И.О.) |  |  |

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 4](#_Toc121872475)

[1.МЕТОДЫ И ИНСТРУМЕНТЫ ПРОГРАММНОЙ ИНЖЕНЕРИИ 5](#_Toc121872476)

[1.1. Техническое задание 5](#_Toc121872477)

[1.2. Постановка задачи проектирования 7](#_Toc121872478)

[1.3. Анализ существующих аналогов 7](#_Toc121872479)

[1.4. Перечень задач, подлежащих решению в процессе разработки. 7](#_Toc121872480)

[1.5. Выбор языка программирования 10](#_Toc121872481)

[1.6. Выбор средств разработки 14](#_Toc121872482)

[1.7. Выводы 16](#_Toc121872483)

[2.ПРОЕКТИРОВАНИЕ КОМПОНЕНТОВ ПРОГРАММНОГО ПРОДУКТА 17](#_Toc121872484)

[2.1. Общие сведения 17](#_Toc121872485)

[2.2. Структура приложения 17](#_Toc121872486)

[2.3. Механизм работы с токенами 23](#_Toc121872487)

[2.4. Механизм смены языка 25](#_Toc121872488)

[2.4. Реализованные компоненты 26](#_Toc121872489)

[3.ТЕСТИРОВАНИЕ И ИНТЕГРАЦИЯ КОМПОНЕНТОВ ПРОГРАММНОГО ПРОДУКТА 29](#_Toc121872490)

[3.1. Системные требования 29](#_Toc121872491)

[3.2. Руководство администратора 29](#_Toc121872492)

[3.3. Пользовательское руководство 31](#_Toc121872493)

[ЗАКЛЮЧЕНИЕ 37](#_Toc121872494)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 38](#_Toc121872495)

ВВЕДЕНИЕ

Веб-приложение — [клиент-серверное](https://ru.wikipedia.org/wiki/%D0%9A%D0%BB%D0%B8%D0%B5%D0%BD%D1%82_%E2%80%94_%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80) приложение, в котором клиент взаимодействует с сервером при помощи браузера. В отличие от обычных приложений речь идет о тех программах, которых не нужно скачивать и устанавливать в постоянную память можно получать доступ через браузер. Разработка веб-приложения представляет собой сферу IT, которая в последнее время развивается быстрыми темпами.

Данное веб-приложение является хорошим проектом для отработки и закрепления знаний в области ООП.

**Актуальность** темы курсового проекта обуславливается активным уходом с Российского рынка иностранных веб-приложений.

**Объектом** работы является веб-разработка, **предмет** – личный кабинет веб-приложения.

**Цель** проекта – разработка личного кабинета пользователя для документирования сети в веб-приложении

Для достижения поставленной цели решаются следующие задачи:

1. Выполнить анализ предметной области;
2. Определить оптимальную структуру системы;
3. Осуществить выбор средств реализации программного продукта, соответствующего выбранной структуре;
4. Разработать веб-приложение;
5. Осуществить тестирование компонентов;
6. Разработать сопроводительную документацию.
7. МЕТОДЫ И ИНСТРУМЕНТЫ ПРОГРАММНОЙ ИНЖЕНЕРИИ

1.1. Техническое задание

**Наименование системы**

Настоящее Техническое задание определяет требования и порядок создания личного кабинета пользователя системы документирования сети.

Основания для разработки

Основание для разработки программного продукта служит задание в курсовой работе по дисциплине «Объектно-ориентированное программирование» и выполнение учебного плана.

Исполнитель

Исполнителем проекта является студент Калужского филиала МГТУ им. Н. Э. Баумана, факультета ИУК, группы ИУК4-52Б, Калашников Артем Сергеевич.

Краткая характеристика области применения

Разрабатываемое веб-приложение предназначено для безопасности приложений.

Целевая аудитория

Веб-приложение может представлять интерес для пользователей среднего возраста.

**Назначение приложения**

Разрабатываемое в данном проекте веб-приложение является личным кабинетом системы документирования сети. Назначение разрабатываемого приложения – применяться в качестве системы документирования сети.

Цели разработки

Целью создания данного приложения является выполнение учебного плана и формирование практических навыков по разработке и реализации структуры классов. Получение первичных навыков ведения проектной и производственно-технологической деятельности, развитие творческих способностей индивидуально для каждого студента. Усвоение методов грамотного ведения, оформления и редактирования технической документации. Как итог, создание веб-приложения и подготовка отчета по проделанной работе.

Плановые сроки начала и окончания работы по созданию системы

Планируемые сроки начала и окончания работы над проектом: 09.09.2022 – 16.12.2022.

Требования к приложению

Разрабатываемое приложение должно являться системой и обеспечивать следующий функционал:

1. Осуществлять вход в веб-приложение использую email и пароль;
2. Предоставлять возможность регистрации в веб-приложении;
3. Предоставлять возможность восстанавливать пароль;
4. Предоставлять возможность верифицирования почты;
5. Предоставлять возможность получать refresh и access-токены;
6. Осуществлять изменение данных с помощью настроек.

Требования к временным характеристикам

Время восстановления после отказа, вызванного сбоем электропитания технических средств (иными внешними факторами), не фатальным сбоем (не крахом) операционной системы, не должно превышать времени восстановления операционной системы и восстановления работы сети.

Время восстановления после отказа, вызванного неисправностью технических средств, фатальным сбоем (крахом) операционной системы, не должно превышать времени, требуемого на устранение неисправностей технических средств и переустановки программных средств.

Этапы разработки

На стадии разработки технического задания должен быть выполнен этап разработки, согласования и утверждения настоящего технического задания.

На стадии проектирования программы должен быть выполнен этап выборки программного обеспечения, библиотек для создания, этап проектирования системы в целом, разработка технической документации.

На стадии реализации производится разработка и тестирование спроектированного приложения.

1.2. Постановка задачи проектирования

Разработать веб-приложение личного кабинета пользователя системы документирования сети.

1.3. Анализ существующих аналогов

Веб-приложения с личным кабинетом появились достаточно давно с появлением термина Web 2.0. На Web 2.0 Conference в 2004 году в своих докладах Дейл Догерти и известный издатель Тим О’Райли описали новый веб как платформу для приложений и подчеркнули ценность контента, который генерируют сами пользователи.

Web 2.0 сейчас — это сеть интерактивных веб-сайтов и платформ, где контент производят пользователи, а не владелец ресурса. Facebook, YouTube и Twitter — платформы эпохи Web 2.0, ориентированные на пользовательский контент и социальные взаимодействия. В последствие для каждого пользователя начали появляться собственные личные кабинеты, где он мог бы безопасно хранить свои личные данные и выкладывать посты.

Суть данного приложения заключается в проверке вводимых данных при подключении:

* Проверка на валидативность полей
* Проверка на правильность вводимых данных в поля
* Проверка на существование данного пользователя в системе
* Работа в приложение с выбранными пользователем настройками

1.4. Перечень задач, подлежащих решению в процессе разработки.

В процессе разработки были поставлены такие задачи:

Создание интерфейса личного кабинета пользователя и настроек личных данных для пользователя. В личный кабинет пользователя должны входить окна авторизации и регистрации. В окно авторизации должно входить несколько элементов управления:

* Email
* Password
* Forgot password

После заполнением пользователем полей для ввода должна произойти проверка на валидность данных:

1. В поле email должна быть не больше 40 символов иметь в составе строки «@;.». Является обязательным полем для ввода.
2. Пароль не должен быть короче 4 символов и не длиннее 20 символов. Является обязательным полем для ввода.

Также выполняется проверка на существование данного пользователя в нашей системе



Рис. 2.1. Поле для ввода Email



Рис. 2.2. Поле для ввода пароля

D:\3 курс 1 семестр\Курсовая\Картинки\7.3.png

Рис. 2.3. Кнопка для восстановления пароля

Следующее окно, которое нам нужно создать - это окно регистрации. В окно регистрации нам необходимо добавить такие поля как:

* Email
* Password
* Confirm password



Рис. 3.1. Поле для ввода Email



Рис. 3.2. Поле для ввода пароля



Рис. 3.2. Поле для подтверждения пароля

И последним окно – это окно настроек. Мы будем изменять полученную информацию при регистрации:

* Email
* Password
* Name
* Language

Кроме окон, должна быть реализована логика работы с токенами, чтобы пользователю выдаются access или refresh – токены, для повышения безопасности работы пользователя в веб-приложении.

Access-токен — это токен, который предоставляет доступ его владельцу к защищенным ресурсам сервера. Обычно он имеет короткий срок жизни и может нести в себе дополнительную информацию, такую как IP-адрес стороны, запрашивающей данный токен.

Refresh-токен — это токен, позволяющий клиентам запрашивать новые access-токены по истечении их времени жизни. Данные токены обычно выдаются на длительный срок.

И последней задачей является подтверждение почты. Проверка будет проводиться с помощью автоматически отправленного письма с ссылкой о верификации аккаунта. Сразу после того, как пользователь выполнит регистрацию, на почту ему будет отправлено письмо с просьбой подтвердить регистрацию. Если регистрация не будет подтверждена в течение двух дней, то ссылка перестанет быть действительной и пользователю придется заново проходить этап регистрации.

1.5. Выбор языка программирования

**Backend**

**Python** - [высокоуровневый язык программирования](https://ru.wikipedia.org/wiki/%D0%92%D1%8B%D1%81%D0%BE%D0%BA%D0%BE%D1%83%D1%80%D0%BE%D0%B2%D0%BD%D0%B5%D0%B2%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F) общего назначения с [динамической](https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D0%BD%D0%B0%D0%BC%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D1%82%D0%B8%D0%BF%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F) [строгой](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D1%80%D0%BE%D0%B3%D0%B0%D1%8F_%D1%82%D0%B8%D0%BF%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F) типизацией и автоматическим управлением памятью, ориентированный на повышение производительности разработчика, читаемости [кода](https://ru.wikipedia.org/wiki/%D0%98%D1%81%D1%85%D0%BE%D0%B4%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%B4) и его качества, а также на обеспечение переносимости написанных на нём программ. Язык является полностью [объектно-ориентированным](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F) в том плане, что всё является [объектами](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)). Необычной особенностью языка является выделение [блоков кода](https://ru.wikipedia.org/wiki/%D0%91%D0%BB%D0%BE%D0%BA_%D0%BA%D0%BE%D0%B4%D0%B0) пробельными отступами. [Синтаксис](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D0%BD%D1%82%D0%B0%D0%BA%D1%81%D0%B8%D1%81_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) ядра языка минималистичен, за счёт чего на практике редко возникает необходимость обращаться к документации. Сам же язык известен как [интерпретируемый](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D0%BF%D1%80%D0%B5%D1%82%D0%B8%D1%80%D1%83%D0%B5%D0%BC%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F) и используется в том числе для написания [скриптов](https://ru.wikipedia.org/wiki/%D0%A1%D0%BA%D1%80%D0%B8%D0%BF%D1%82). Недостатками языка являются зачастую более низкая скорость работы и более высокое потребление памяти написанных на нём программ по сравнению с аналогичным кодом, написанным на компилируемых языках, таких как [C](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8_(%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F)) или [C++](https://ru.wikipedia.org/wiki/C%2B%2B).

**Java** - [строго](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D0%B8_%D1%81%D0%BB%D0%B0%D0%B1%D0%B0%D1%8F_%D1%82%D0%B8%D0%BF%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F) [типизированный](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D0%B0%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D1%82%D0%B8%D0%BF%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F) [объектно-ориентированный язык программирования](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F) общего назначения, разработанный компанией [Sun Microsystems](https://ru.wikipedia.org/wiki/Sun_Microsystems) (в последующем приобретённой компанией [Oracle](https://ru.wikipedia.org/wiki/Oracle)). Разработка ведётся сообществом, организованным через [Java Community Process](https://ru.wikipedia.org/wiki/Java_Community_Process); язык и основные реализующие его технологии распространяются по лицензии [GPL](https://ru.wikipedia.org/wiki/GPL). Права на торговую марку принадлежат корпорации [Oracle](https://ru.wikipedia.org/wiki/Oracle).

Приложения Java обычно [транслируются](https://ru.wikipedia.org/wiki/%D0%A2%D1%80%D0%B0%D0%BD%D1%81%D0%BB%D1%8F%D1%82%D0%BE%D1%80) в специальный [байт-код](https://ru.wikipedia.org/wiki/%D0%91%D0%B0%D0%B9%D1%82-%D0%BA%D0%BE%D0%B4), поэтому они могут работать на любой [компьютерной архитектуре](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D0%B0%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0), для которой существует реализация [виртуальной Java-машины](https://ru.wikipedia.org/wiki/Java_Virtual_Machine).

Скорость выполнения приложений на Java в несколько раз быстрее чем на Python. Так например, простое двоичное дерево выполняется в Java в 10 раз быстрее.

**C#** - [объектно-ориентированный](http://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) [язык программирования](http://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F) для платформы .NET. Основным постулатом С# является высказывание: "всякая сущность есть объект". Язык основан на строгой компонентной архитектуре и реализует передовые механизмы обеспечения безопасности кода.

C# относится к семье языков с C-подобным синтаксисом. Поддерживает черты объектно-ориентированного программирования. Является надежным и устойчивым языком, за счет использования «сборки мусора», безопасности типов и обработки исключений. C# разрабатывался "с нуля" и вобрал в себя много полезных свойств таких языков, как C++, Java, Visual Basic, а также Pascal, Delphy и др., избежав при этом многих отрицательных сторон своих предшественников. Мощная библиотека каркаса Framework.Net поддерживает удобство построения различных типов приложений и позволяет достаточно просто сохранять и получать информацию из баз данных.

Главное различие между двумя языками программирования в C# или Java заключается в их предполагаемом использовании. Java в основном предназначен для разработки мобильных приложений (точнее - Android). Напротив, C# фокусируется на веб-разработке и разработке игр.

После проведенного сравнения языков программирования был выбран наилучший вариант – С#. Так как он быстрее чем Python и лучше подходит для веб-разработки, чем Java

**Frontend**

**JavaScript** — это язык программирования, который используют для написания frontend-части сайтов, а также мобильных приложений. Часто в текстах и обучающих материалах название языка сокращают до JS. Это язык программирования высокого уровня, то есть код на нем понятный и хорошо читается. JS поддерживают все популярные браузеры. Во frontend-части сайтов язык используют для создания интерактива (анимаций, всплывающих форм, автозаполнения), так как он связан с HTML и CSS и может ими манипулировать.

JavaScript называют языком скриптов или сценариев. Скрипты — это набор инструкций, которые выполняются при загрузке страницы. Браузер самостоятельно интерпретирует код на JavaScript, для этого даже не требуется компиляция (перевод языка программирования в машинный код).

**CoffeeScript**  — язык программирования, [транслируемый](https://ru.wikipedia.org/wiki/%D0%A2%D1%80%D0%B0%D0%BD%D1%81%D0%BB%D1%8F%D1%82%D0%BE%D1%80) в [JavaScript](https://ru.wikipedia.org/wiki/JavaScript). CoffeeScript добавляет [синтаксический сахар](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D0%BD%D1%82%D0%B0%D0%BA%D1%81%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D1%81%D0%B0%D1%85%D0%B0%D1%80) в духе [Ruby](https://ru.wikipedia.org/wiki/Ruby) и [Erlang](https://ru.wikipedia.org/wiki/Erlang) для того, чтобы улучшить читаемость кода и уменьшить его размер. CoffeeScript позволяет писать более компактный код по сравнению с [JavaScript](https://ru.wikipedia.org/wiki/JavaScript).

**TypeScript** — это язык программирования для веб-разработки, основанный на [JavaScript](https://blog.skillfactory.ru/glossary/javascript/). Делает код понятнее и надежнее, добавляет статическую типизацию (переменные привязаны к конкретным типам данных), а также может быть скомпилирован в JavaScript. TypeScript используют[фронтенд-](https://skillfactory.ru/frontend-razrabotchik/?utm_source=blog&utm_medium=referral&utm_campaign=TypeScript&utm_content=&utm_term=text) и [бэкенд-разработчики](https://skillfactory.ru/backend-razrabotchik-na-golang/?utm_source=blog&utm_medium=referral&utm_campaign=TypeScript&utm_content=&utm_term=text). TypeScript добавляет в язык строгую типизацию. Каждой переменной при создании присваивается определенный тип (type) — стандартный или созданный самим разработчиком. Создать тип можно в пределах возможностей языка: например, число от 1 до 31 для записи дня в месяц или массив из двух элементов для записи координат. TypeScript помогает сократить время на выявление и устранение багов, которые иногда сложно найти в динамической среде JavaScript. С помощью TypeScript можно написать более понятный и читаемый код, который максимально описывает предметную область. Таким образом архитектура становится более выраженной. TypeScript добавляет в JS множество возможностей ООП: и модификаторы доступа, и абстрактные классы, и другие особенности подхода.

**Выбор инструмента стилизации страниц**

**CSS**  — язык описания внешнего вида HTML-документа. Это одна из базовых технологий в современном интернете. Практически ни один сайт не обходится без CSS, поэтому HTML и CSS действуют в единой связке.

Каскадные таблицы стилей работают с [HTML](https://skillbox.ru/media/code/chto_takoe_html/), но это совершенно другой язык. HTML структурирует документ и упорядочивает информацию, а CSS взаимодействует с браузером, чтобы придать документу оформление.

**SASS** — это препроцессор, позволяющий писать код для стилей CSS, используя всевозможные: переменные, циклы, миксины, функции и многое другое.

**SCSS**  — "диалект" языка [SASS](http://sass-lang.com/). Синтаксис этого языка очень гибок, он учитывает множество мелочей, которые так желанны в CSS. Более того, в нём есть даже логика (@if, [each](https://habr.com/en/users/each/)), математика (можно складывать как числа, строки, так и цвета). Возможно, некоторые возможности SCSS покажутся вам избыточными, но, на мой взгляд, лишними они не будут, останутся "про запас".

Для разработки frontend части был выбран TypeScript и SCSS. TypeScript – типизированный язык и поэтому будет намного меньше ошибок на этапе тестирования, чем в том же React.js, а SCSS более приближен к обычному CSS и все вложенные стили записываются через фигурные скобки - {}. В SASS же все вложенные свойства выделяются отступами, при чем очень важно соблюдать отступы и их количество

1.6. Выбор средств разработки

Данный выбор оказывает значительное влияние на удобство разработки, имеющийся инструментарий, возможности отладки и развертывания системы на различных платформах, эффективность и качество реализуемых приложений.

**WebStorm**

Интегрированная среда разработки для JavaScript и смежных технологий. Как и другие IDE от JetBrains, она делает процесс разработки более приятным, автоматизируя рутинную работу и помогая вам легко справляться со сложными задачами

**Visual Studio Code**

Это среда разработки, которая была разработана компанией Microsoft для Windows, Linux и macOS. Позиционируется как легкий редактор кода для кроссплатформенной разработки веб- и облачных приложений. Включает в себя отладчик, инструменты для работы с Git, подсветку синтаксиса с помощью технологии IntelliSense и средства рефакторинга. Имеет широкие возможности для кастомизации: пользовательские темы, сочетания клавиш, файлы конфигурации.

Многие возможности Visual Studio Code недоступны через графический интерфейс, зачастую они используются через палитру команд или JSON-файлы. Палитра команд представляет собой подобие командной строки, которая вызывается сочетанием клавиш.

Для сохранения прогресса работы было необходимо использование системы Git. В качестве средства ведения такого репозитория был выбран GitHub за свои особенности:

* Система представляет собой своеобразную социальную сеть для разработчиков, в которой можно хранить код, общаться с коллегами программистами и анализировать их деятельность, отслеживать ошибки.
* Добавлен инструмент управления проектами, сделанный на основе техники Канбан-досок.
* В сервисе есть возможность добавлять и утверждать новый запрос непосредственно в коде.
* На самой доске можно перетаскивать карточки задач, багов или заметок.
* GitHub позволяет вносить изменения в строки.
* На любой стадии процесса можно откатить приложение и запустить его локально.
* Доступна опция истории в листе коммитов, и нет необходимости пролистывать каждый из них, когда нужно выяснить момент добавления кода.
* GitHub предоставляет информацию о том, кто из участников процесса изменил линию файла.
* Сервис абсолютно бесплатен для проектов с открытым исходным кодом и предоставляет им все возможности (включая SSL), а для частных проектов предлагаются различные платные тарифные планы.
* Кроме размещения кода, участники могут общаться, комментировать правки друг друга, а также следить за новостями знакомых.
* С помощью широких возможностей Git программисты могут объединять свои репозитории — GitHub предлагает удобный интерфейс для этого и может отображать вклад каждого участника в виде дерева.
* Для проектов есть личные страницы, небольшие Вики и система отслеживания ошибок.
* Прямо на сайте можно просмотреть файлы проектов с подсветкой синтаксиса для большинства языков программирования.
* Есть возможность прямого добавления новых файлов в свой репозиторий через веб-интерфейс сервиса.
* Код проектов можно не только скопировать через Git, но и скачать в виде обычных архивов с сайта. (Для этого достаточно добавить /zipball/master/ в конец адресной строки.)
* Кроме Git, сервис поддерживает получение и редактирование кода через SVN и Mercurial.
* На сайте есть pastebin-сервис gist.github.com для быстрой публикации фрагментов кода.

1.7. Выводы

Таким образом, исходя из требований к реализуемому веб-приложению, рассмотрения возможностей наиболее подходящих инструментов, вариантов разработки и последующего их сравнения было решено использовать следующие решения:

В качестве инструментов для разработки было решено использовать: язык программирования TypeScript, SCSS и C#.

В качестве среды разработки была выбрана Visual Studio Code.

1. ПРОЕКТИРОВАНИЕ КОМПОНЕНТОВ ПРОГРАММНОГО ПРОДУКТА

2.1. Общие сведения

Данный курсовой проект представляет собой личный кабинет пользователя системы документирования сети

.

2.2. Структура приложения

Разрабатываемое приложение состоит из следующих основных классов:

* Класс User;
* Класс MyMailService;
* Класс MailSettings;
* Класс MongoDBUserRepositoryService;
* Класс RamUserRepositoryService;
* Класс TokenApiDto;
* Класс UserDBSettings;
* Класс RsaTokenService;
* Класс JwtSettings;
* Класс Program.

Основная структура приложения представлена на рисунке 4



Рис. 4. Иерархия классов

Ниже приведено описание каждого класса.

Таблица 1 – Класс «User» – хранит данные пользователя.

|  |  |  |
| --- | --- | --- |
| Наименование | Тип данных | Примечание |
| Переменные класса | | |
| Id | string | Хранит id пользователя |
| Name | string | Устанавливаемый фон |
| Email | string | Хранит почту пользователя |
| PasswordHash | string | Хранит хеш пароля |
| Salt | string | Хранит в себе данные для хеширования пароля |
| Verified | bool | Хранит данные пройдена ли верификация или нет |
| VerificationCode | string | Хранит верификационный код |
| VerificationCodeExpiryTime | DateTime | Хранит время смены верификационного кода |
| RefreshToken | string | Хранит refresh-токен |
| RefreshTokenExpiryTime | DateTime | Хранит время смены refresh-токена |

Таблица 2 – Класс «MyMailService» – сервис отправки сообщений. Генерирует сообщения для отправки и посылает пользователям каждому с его сгенерированным.

|  |  |  |
| --- | --- | --- |
| Наименование | Тип данных | Примечание |
| Переменные класса | | |
| \_mailSettings | MailSettings | Хранит конфигурацию почты и работы с ней |
| \_templateDirectory | string | Хранит директорию в которой находятся шаблоны |
| \_imageDirectory | string | Хранит директорию в которой находятся картинки |
| Методы класса | | |
| SendAuthMailAsync () | None | выполняет сборку письма определенному пользователю |

Таблица 3 – Класс «MailSettings» – хранит конфигурацию настроек, связанную с отправкой писем

|  |  |  |
| --- | --- | --- |
| Наименование | Тип данных | Примечание |
| Переменные класса | | |
| Mail | string | Хранит почту пользователя |
| DisplayName | string | Хранит имя пользователя |
| Password | string | Хранит пароль пользователя |
| Host | string | Хранит хост |
| Port | int | Хранит порт |

Таблица 4 – Класс «MongoDBUserRepositoryService» – хранит информацию о пользователях в хранилище и производит работу с ними.

|  |  |  |
| --- | --- | --- |
| Наименование | Тип данных | Примечание |
| Переменные класса | | |
| \_usersCollection | Users | Хранит коллекцию пользователей хранилища |
| Методы класса | | |
| AddAsync () | None | добавление user |
| UpdateAsync() | None | обновление user |
| GetByEmailAsync() | None | получение email |

Таблица 5 – Класс «RamUserRepositoryService» – работает с коллекцией пользователей в памяти

|  |  |  |
| --- | --- | --- |
| Наименование | Тип данных | Примечание |
| Переменные класса | | |
| \_users | User | Хранит список пользователей |
| Методы класса | | |
| AddAsync() | None | добавление user в памяти |
| UpdateAsync() | None | обновление user в памяти |
| GetByEmailAsync() | None | получение email в памяти |

Таблица 6 – Класс «TokenApiDto» – хранит данные токенов

|  |  |  |
| --- | --- | --- |
| Наименование | Тип данных | Примечание |
| Переменные класса | | |
| AccessToken | string | Хранение access токена |
| RefreshToken | string | Хранение refresh токена |

Таблица 7 – Класс «UserDBSettings» – отвечает за настройки пользователя связь его с хранилищем всех пользователей

|  |  |  |
| --- | --- | --- |
| Наименование | Тип данных | Примечание |
| Переменные класса | | |
| ConnectionString | string | Хранит строку подключение к нашему хранилищу пользователей |
| DatabaseName | string | Хранит имя хранилища |
| UsersCollection | string | Хранит коллекцию имен из хранилища |

Таблица 8 – Класс «RsaTokenService» – выполняет ассиметричный алгоритм шифрования.

|  |  |  |
| --- | --- | --- |
| Наименование | Тип данных | Примечание |
| Переменные класса | | |
| \_expiryDuration | TimeSpan | Хранит срок действия jwt-токенов |
| \_settings | JwtSettings | Хранит конфигурацию jwt-токенов |
| Методы класса | | |
| GenerateAccessToken () | None | Генерация аccess токена |
| GenerateRefreshToken () | None | Генерация refresh токена |
| GetPrincipalFromExpiredToken () | None | Проверка действительности токена |

Таблица 9 – Класс «JwtSettings» – хранит приватный и публичный ключ для выполнения ассиметричного алгоритма с jwt-токенами.

|  |  |  |
| --- | --- | --- |
| Наименование | Тип данных | Примечание |
| Переменные класса | | |
| PrivateKey | RsaSecurityKey | Приватный ключ для ассиметричного алгоритма RSA |
| PublicKey | RsaSecurityKey | Публичный ключ для ассиметричного алгоритма RSA |
| Issuer | string | Пользователь |

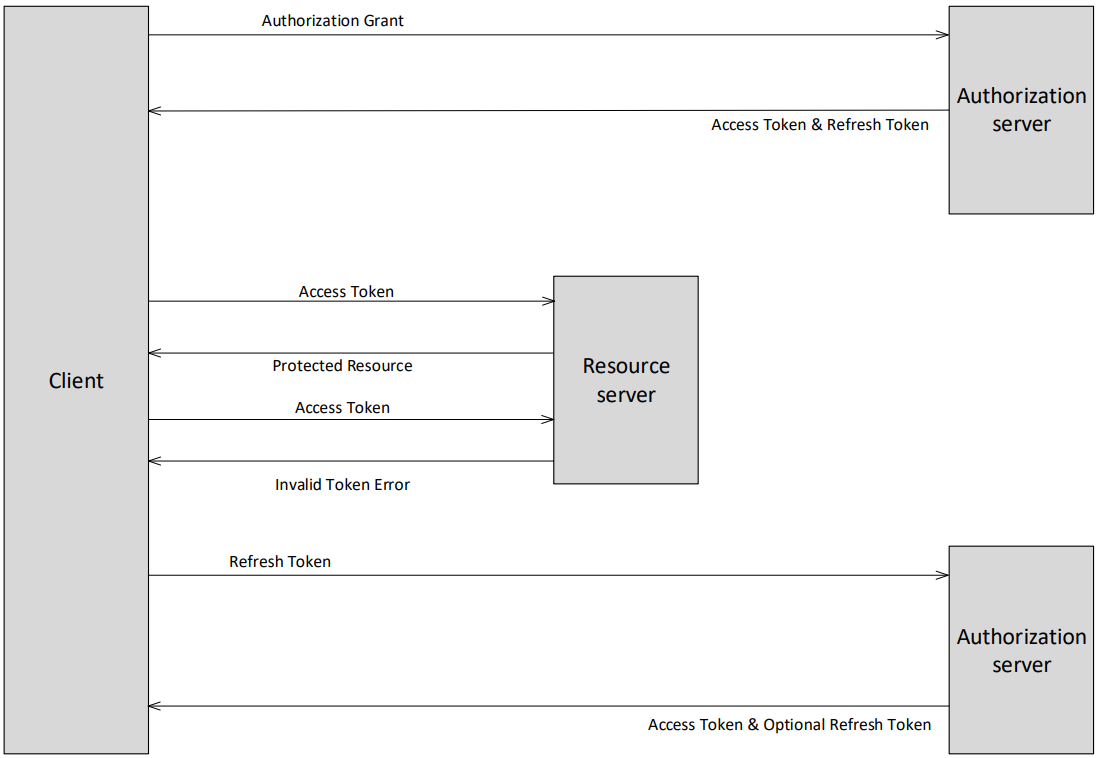
Таблица 10 – Класс «Program»– хранит основные настройки веб-приложения.

|  |  |  |
| --- | --- | --- |
| Наименование | Тип данных | Примечание |
| Переменные класса | | |
| builder | var | Отвечает за создание приложения |
| dbSettings | UserDBSettings | Хранит конфигурацию подключения к хранилищу |
| mailSettings | MailSettings | Хранит конфигурацию почты и работы с ней |
| jwtSettings | JwtSettings | Хранит конфигурацию jwt-токенов |

2.3. Механизм работы с токенами

Когда пользователь авторизуется на нашем сервере он в ответ помимо jwt-токена получает refresh-токенов. Refresh-токен представляет из себя набор символов – уникальный ключ.

Сервер Авторизации сохраняет у себя что у этого пользователя этот refresh-токен при этом access-токен, который нужен для доступа к нашему закрытому ресурсу он делается короткоживущим 10 минут и в нем указываем время его жизни. Соответственно имея access-токен, клиент может отправлять запросы к нашему серверу в течение 10 минут. При выполнении запроса к серверу по истечению времени access-токен отправляется уже просроченный на наш сервер ресурсов. Наш сервер ресурсов расшифровывает пришедшие токены и может проверить время жизни и если токен просрочен вместо того чтобы вернуть данные он вернет ошибку клиенту, что токен просрочен. В данном случае клиент передает jwt-токен и refresh-токен обратно серверу авторизации. Refresh-токен же нигде и никуда не передавался он один раз отдался клиенту и клиент его у себя хранит поэтому он не может утечь. Передавая серверу jwt-токен и refresh-токен серверу авторизации для определения пользователя. Сервер авторизации сравнивает совпадает ли refresh-токен, который хранит у себя в базе. Если они совпадают, то сервер генерирует новый access-токен и refresh-токен и дальше в течение 10 минут клиент общается с сервером при помощи access-токен (см. рис. 5).



**Рис. 5** Серверное взаимодействие JWT- токенов с пользователем

Если злоумышленник захватывает access-токен, он в течение жизни токены может выполнять какие-то злодейства, поэтому access-токен делается короткоживущим. По истечению 10 минут он получит ошибку, ему нужно будет обновиться, но он не сможет получить новый jwt-токен от сервиса, так как не имеет refresh-токен, чтобы совершить новые злодейства ему нужно будет снова захватить новый access-токен.

Для того чтобы каждый микросервис проверял подписи нашего jwt-токена, но ключом, которым мы подписываем, может обладать только один микросервис, который проводит авторизацию, то здесь выполняется ассиметричный алгоритм шифрования.

Ассиметричный алгоритм шифрования – где для шифрования и для дешифрования данных используются разные пары ключей, они взаимосвязаны, но по одному нельзя получить второй. Шифрование данных мы проводим одним ключом закрытым, а расшифровываем другим открытым. В данном веб-приложении мы использовали алгоритм RSA. Единственный микросервис, который используется для авторизации обладает секретным ключом, которым будут подписываться токены. Также он обладает публичным ключом, которым можно будет расшифровать эти токены, проверить валидность пришедших данных.

Сначала клиент авторизуется в нашей системе наш микросервис генерирует jwt-токен подписывает его секретным ключом и отправляет клиенту refresh-токен. Дальше наш клиент может взаимодействовать с произвольным микросервисом, который обладает публичным ключом, с помощью которого можно проверить наш токен. После чего на эти данные приходит либо какойто ответ, либо сообщается ошибка о просрочке jwt-токена. При просрочке наш клиент обращается к серверу и отправляет jwt-токен и refresh-токен и в ответ получает новый jwt-токен и refresh-токен

Такой механизм, достаточно широко распространенный позволяет обезопасить в целом наше приложение

2.4. Механизм смены языка

Механизм смены языка реализован с помощью интерфейса (см. рис. 6):



**Рис. 6** Интерфейс перевода

От интерфейса реализуются объекты – языки. Каждый язык соответствует одинаковый набор переменных типа string. Каждая страница имеет свой массив с набором переменных.

Также помимо интерфейса был создан тип данных Language и список данного типа, в котором хранятся все языки, которые есть в нашем веб-приложении.   
 Весь перевод слов хранится в словаре Для каждого языка имеется свой словарик.

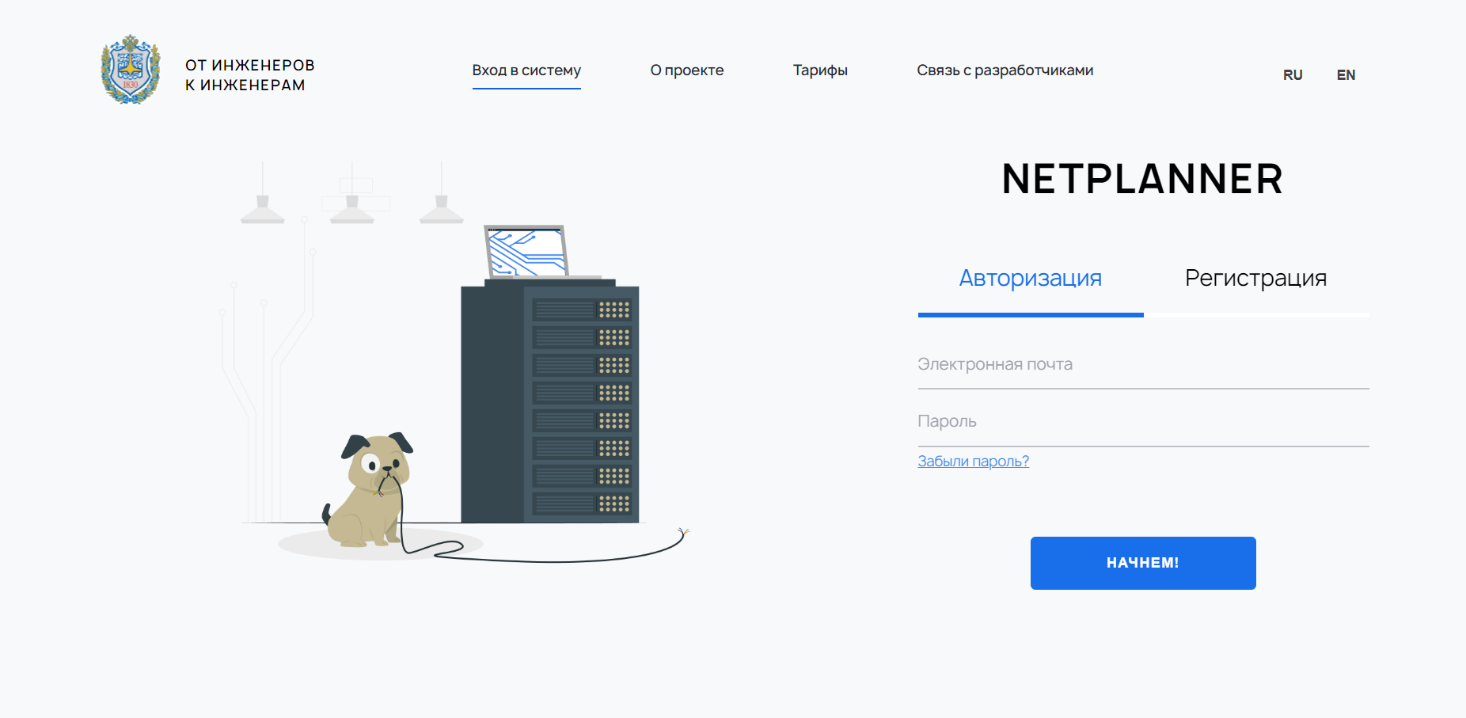
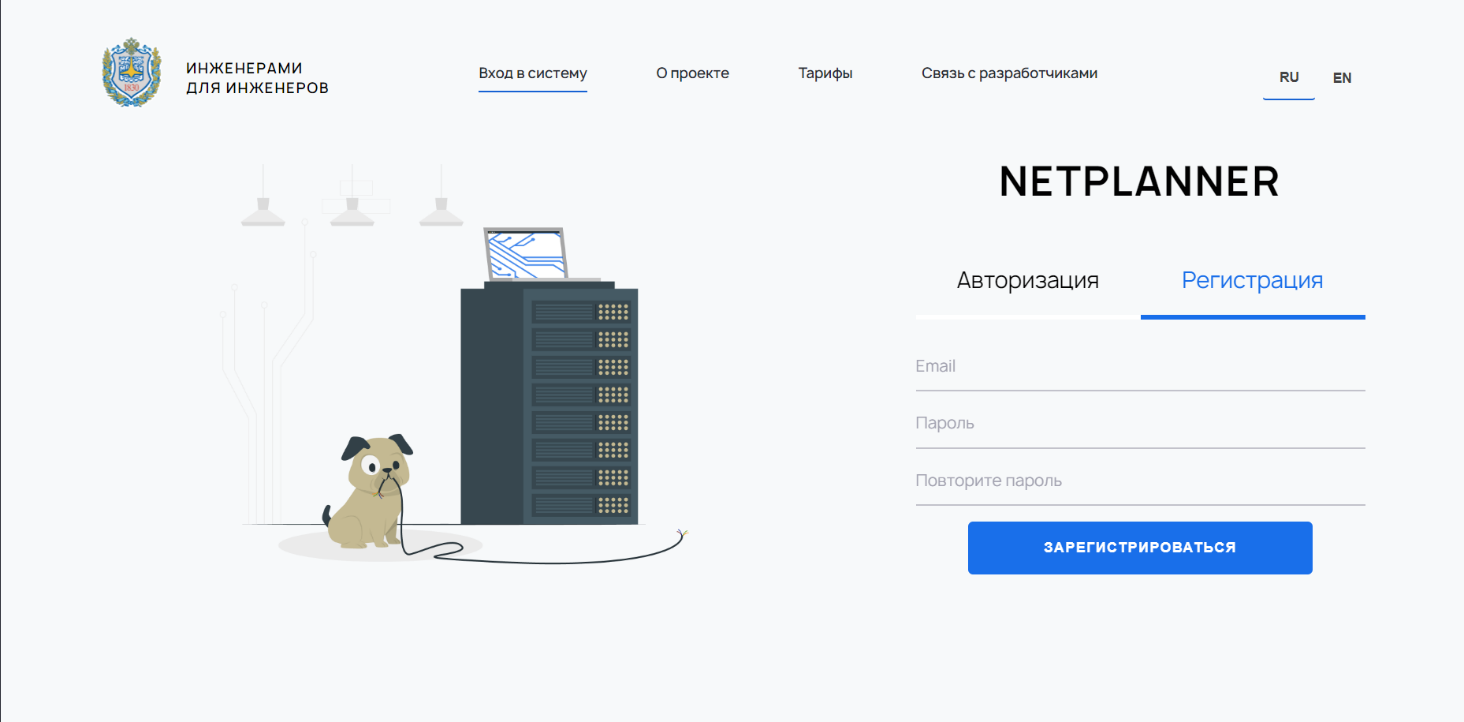
Переключение языка осуществляется с помощью изменения состояния «language» типа «Language» при изменении состояния, начальное значение равное «en» меняется на выбранный пользователем язык из созданного списка языков.

2.4. Реализованные компоненты

Была реализована страница авторизации на которой присутствует два поля ввода:

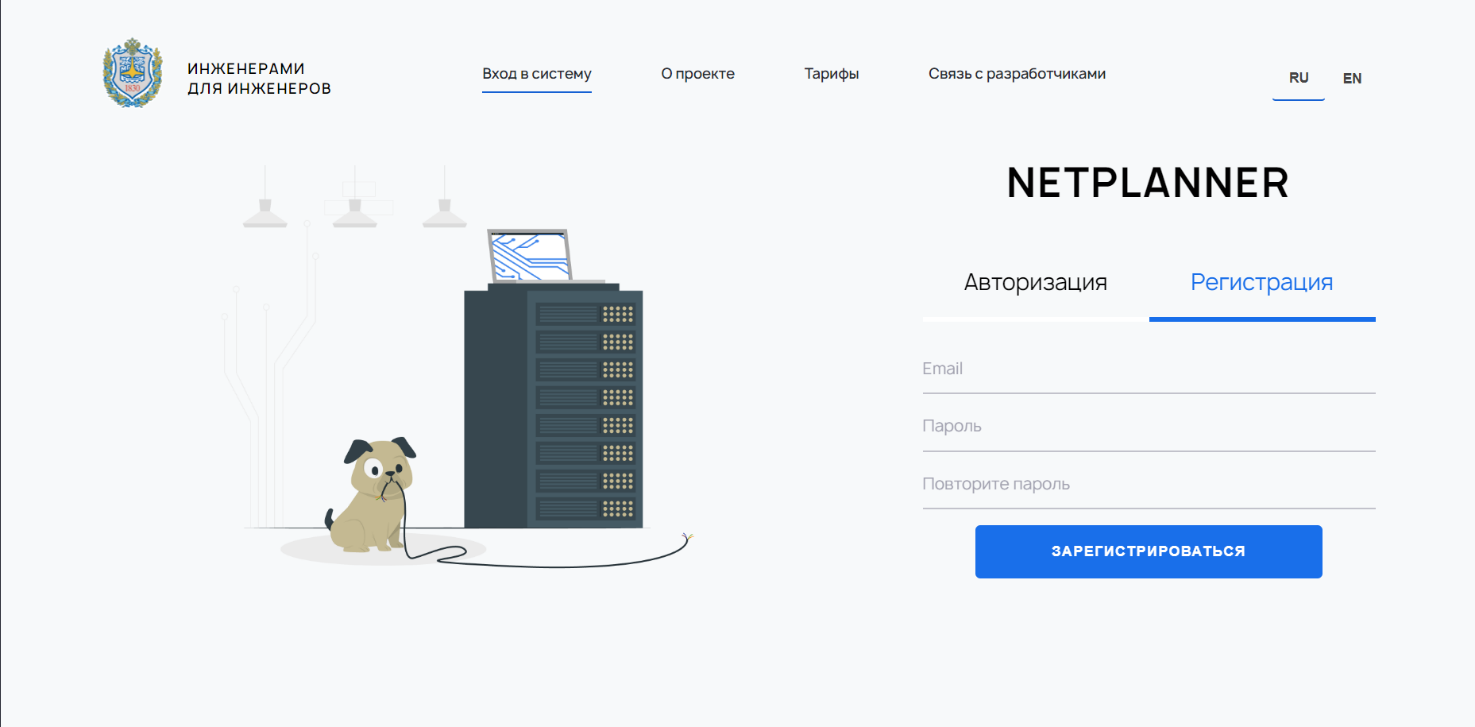
1. Электронная почта
2. Пароль

Также на странице была реализована кнопка «Забыли пароль?» при нажатии на которую на почту будет выслан новый пароль (см. рис. 7)

****

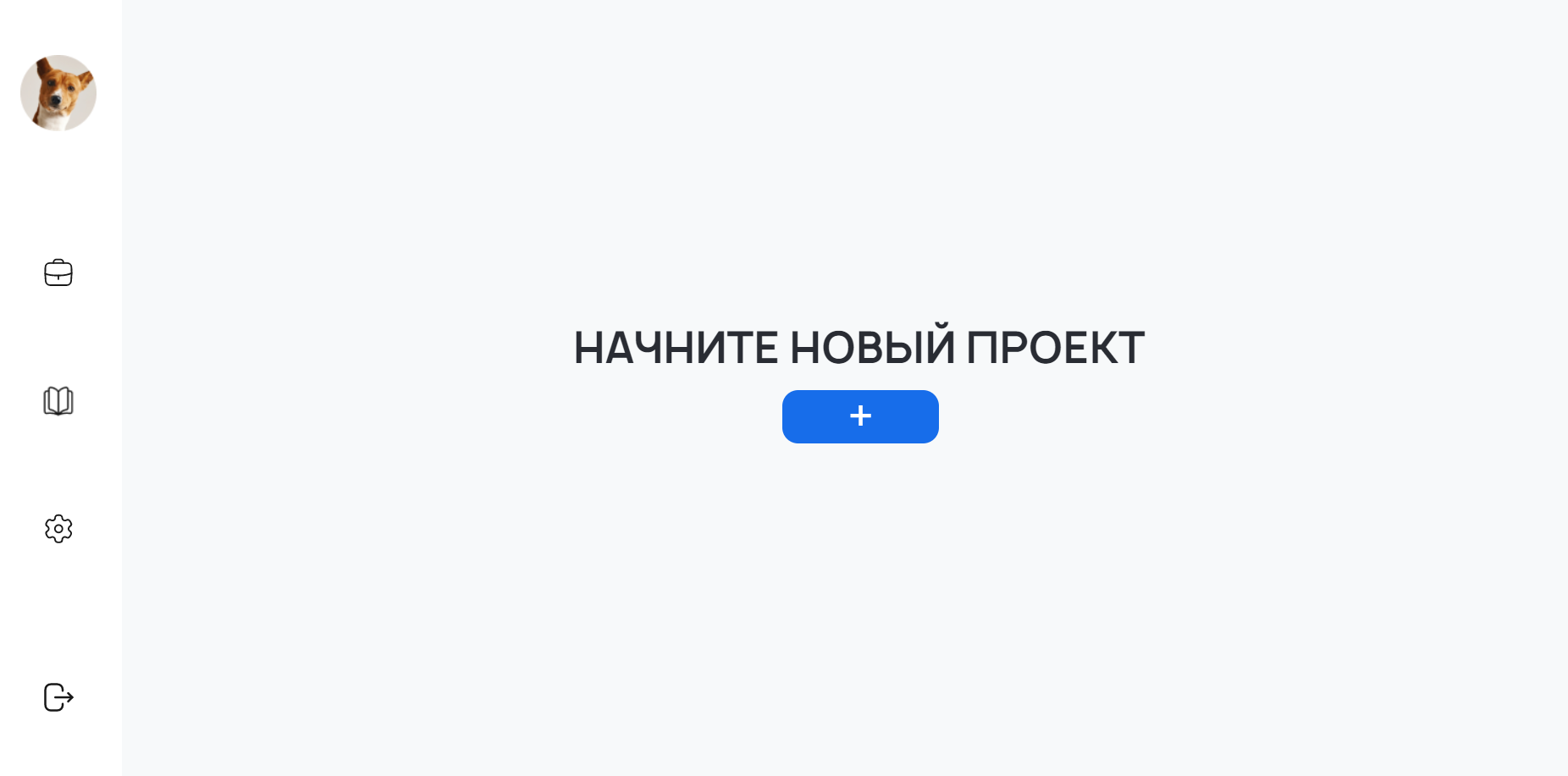
**Рис. 7** Окно авторизации

Окно регистрации предлагает пользователю зарегистрироваться и внести свои личные данные email и пароль (см. рис. 8)

****

**Рис. 8** Окно регистрации

Главное окно веб-приложения. В этом окне пользователь может перейти к настройкам или выйти из своего аккаунта (см. рис. 9)



**Рис. 9** Главное окно

Окно настроек пользователя. В этом окне пользователь может сменить свои данные (см. рис. 10):

* Email
* Пароль
* Language
* Name



**Рис. 10** Окно настроек

1. ТЕСТИРОВАНИЕ И ИНТЕГРАЦИЯ КОМПОНЕНТОВ ПРОГРАММНОГО ПРОДУКТА

3.1. Системные требования

Минимальные требования:

* Операционная система Windows, Linux или MacOS;
* 256 МБ ОЗУ;
* Процессор с частотой 500МГц
* Доступ в интернет;
* Браузер
* 200 Мб дискового пространства

3.2. Руководство администратора

Для запуска нашего веб-приложения надо установить docker

После установки докера нужно создать контейнер с приложением(см. рис. 11). Контейнеризация обеспечивает надежную изоляцию процессов и повышает уровень безопасности систем.

Приложения, которые работают внутри контейнера, не имеют доступа к основной ОС и не могут на неё влиять.

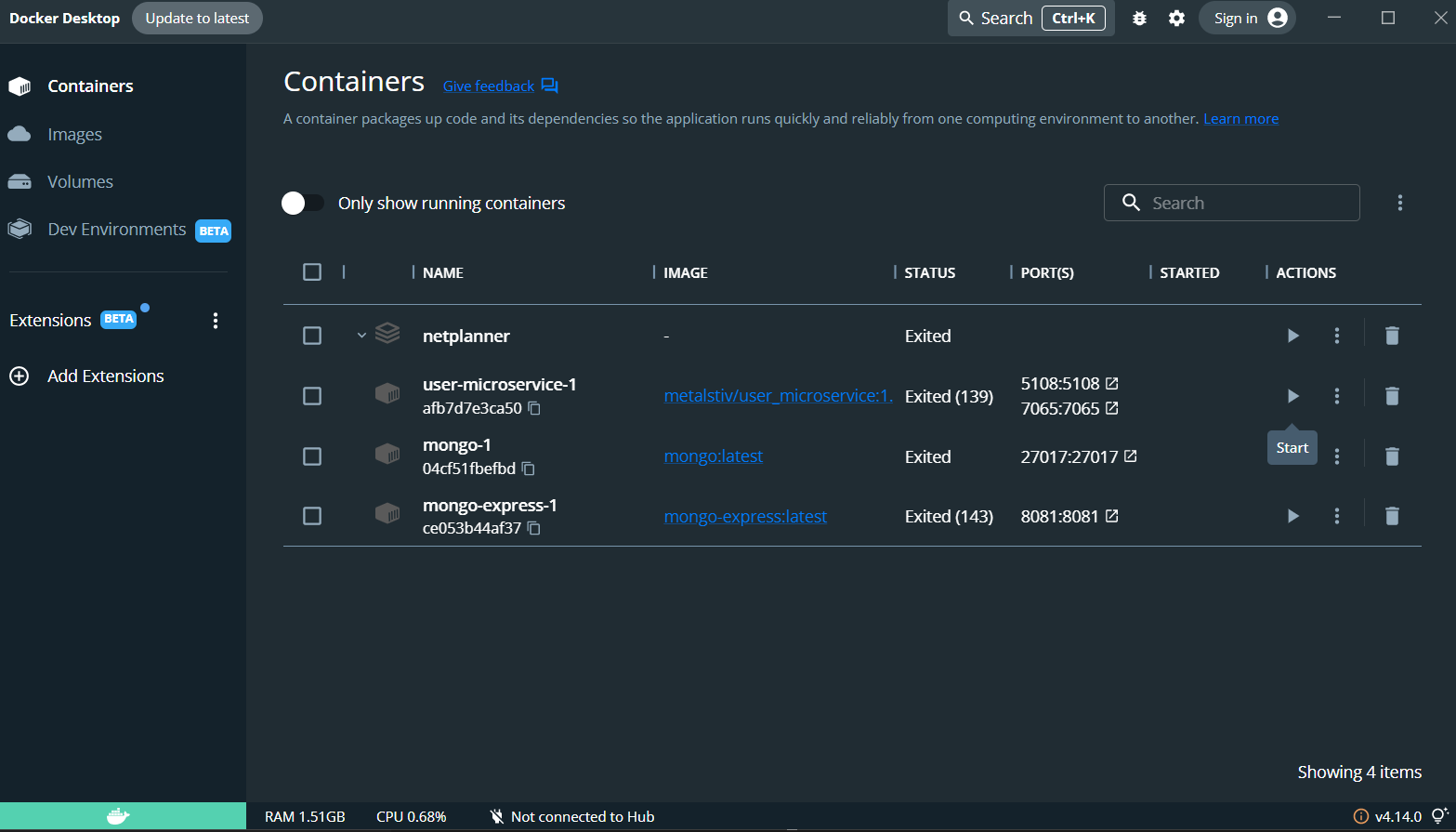


Рис. 11 Контейнеры в docker

После чего нам надо перейти в папку frontend и установить npm с помощью команды в терминале «npm install» (см. рис. 12)

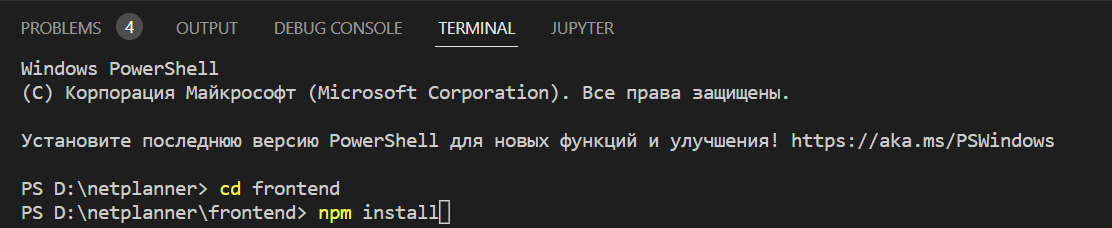


Рис. 12 Установка npm

Как только npm установится, нужно будет запустить его с помощью команды “npm start” из папки frontend (см. рис. 13)

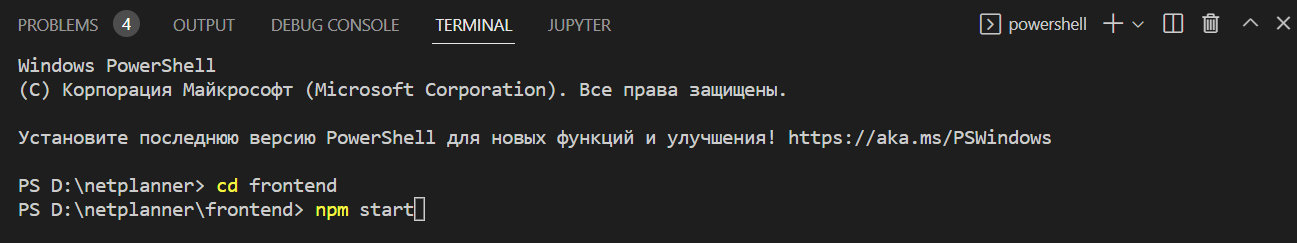


Рис. 13 Запуск веб-приложение

После чего по localhost:3000 мы сможем увидеть наш результат (см. рис. 14).

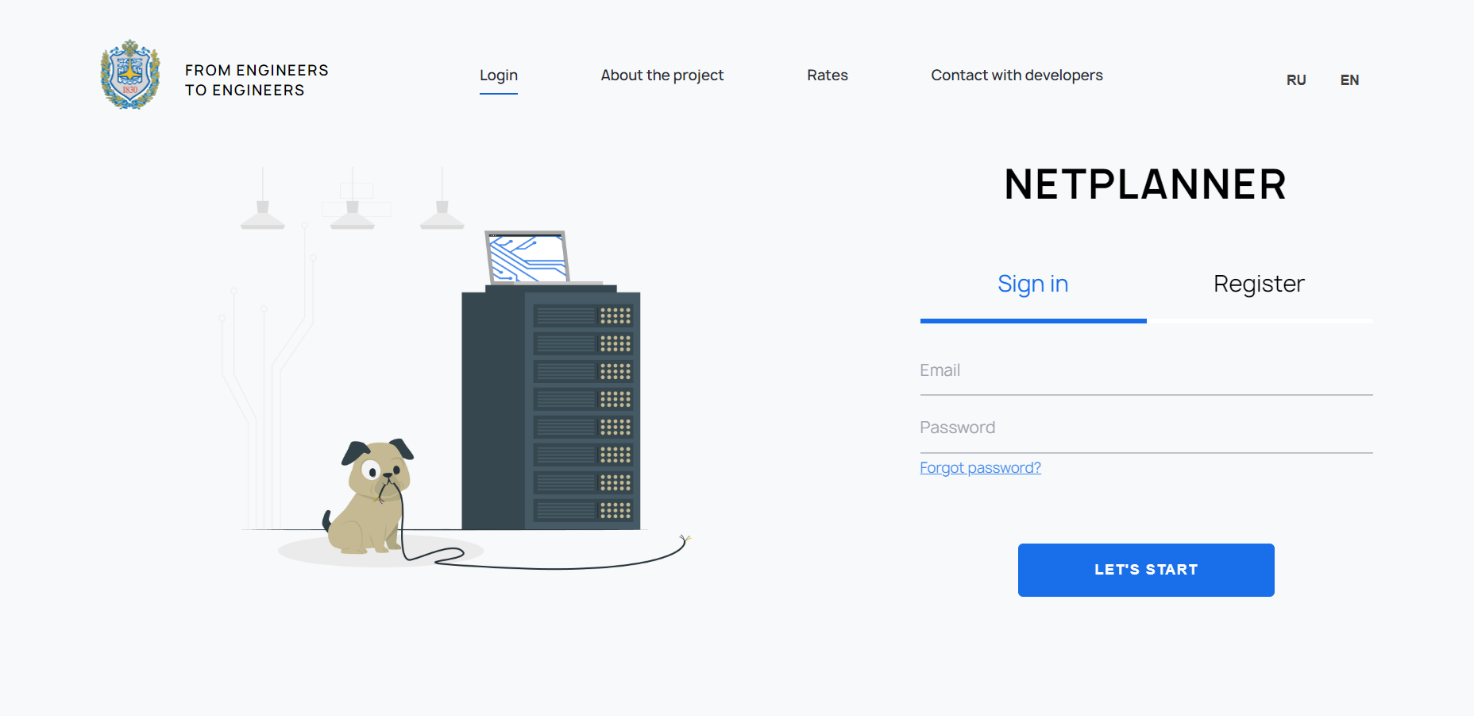


Рис. 14 Веб-приложение

3.3. Пользовательское руководство

Продукт не требует установки, необходимо лишь перейти на сайт

После открытия веб-приложения перед нами появляется окно авторизации, которое потребует от пользователя ввести его данные:

* Электронная почта
* Пароль

Также если пользователь забыл пароль, то можно воспользоваться вспомогательной кнопкой «Забыли пароль?», которая поможет восстановить ваш пароль. Если все данные введены, то можно нажимать на кнопку авторизоваться (см. рис. 15).

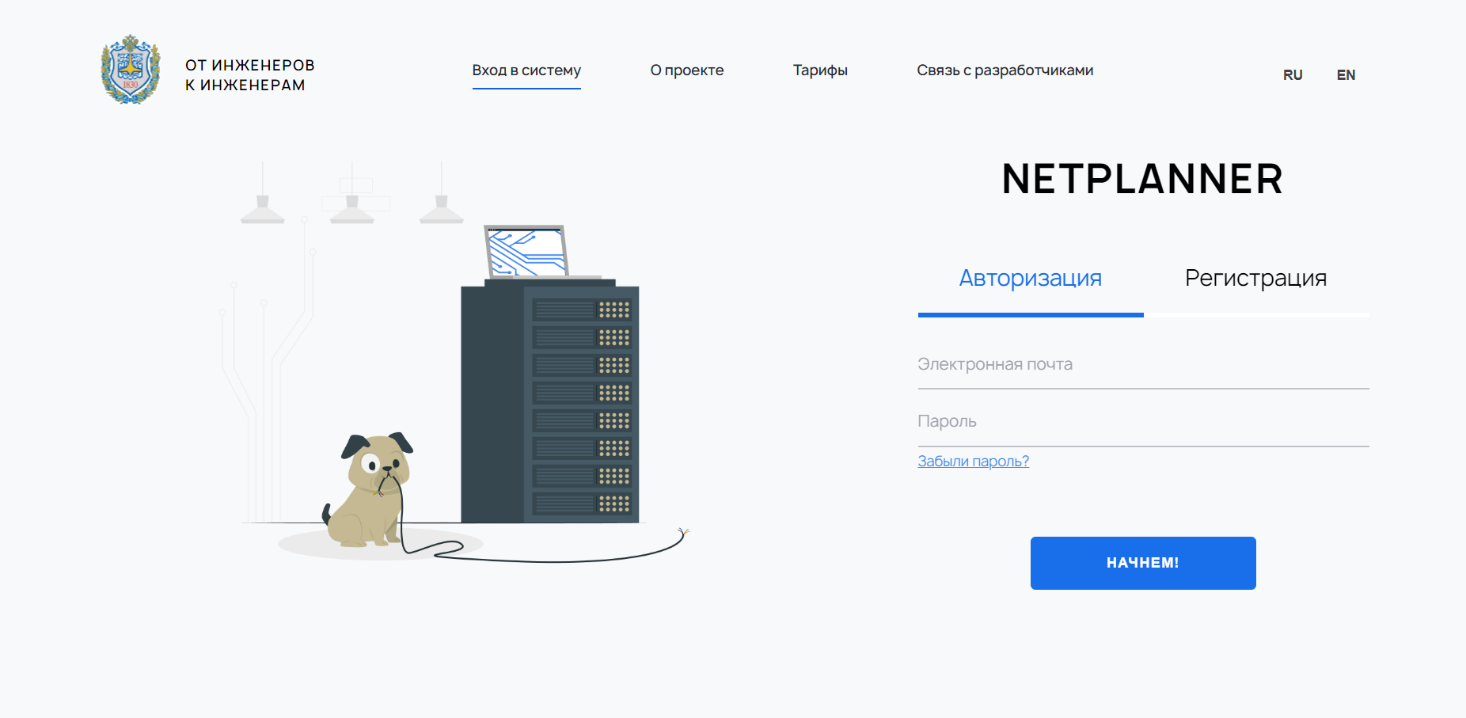
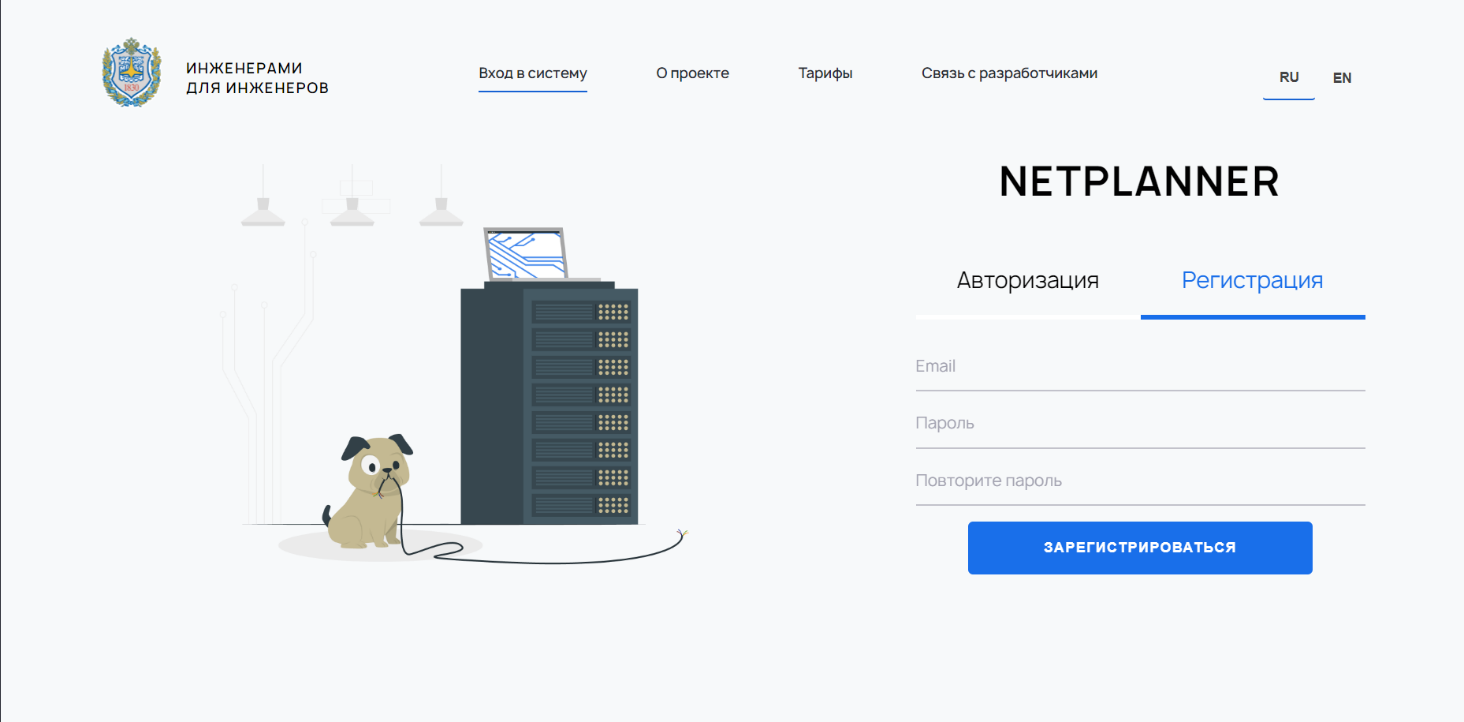


Рис. 15. Окно авторизации

Если же пользователь не помнит пароль, то на почту придет новый пароль (см. рис. 16).



Рис. 16. Пароль

При нажатии на первую кнопку «Регистрация» пользователь переходит в раздел регистрации.

В данном разделе от пользователя потребуется внести некоторые данные:

* Email
* Пароль
* Повторить пароль

После чего пользователь может нажать на кнопку зарегистрироваться. При регистрации в веб-приложение, пользователю будет предложено выбрать язык, на котором он хочет получать информацию и по окончанию регистрации язык, который выбрал пользователь сохранится и будет с ним в дальнейшем. Если же у пользователя возникнет необходимость в смене языка он может перейти во вкладку «Настройки» и изменить язык (см. рис. 17).

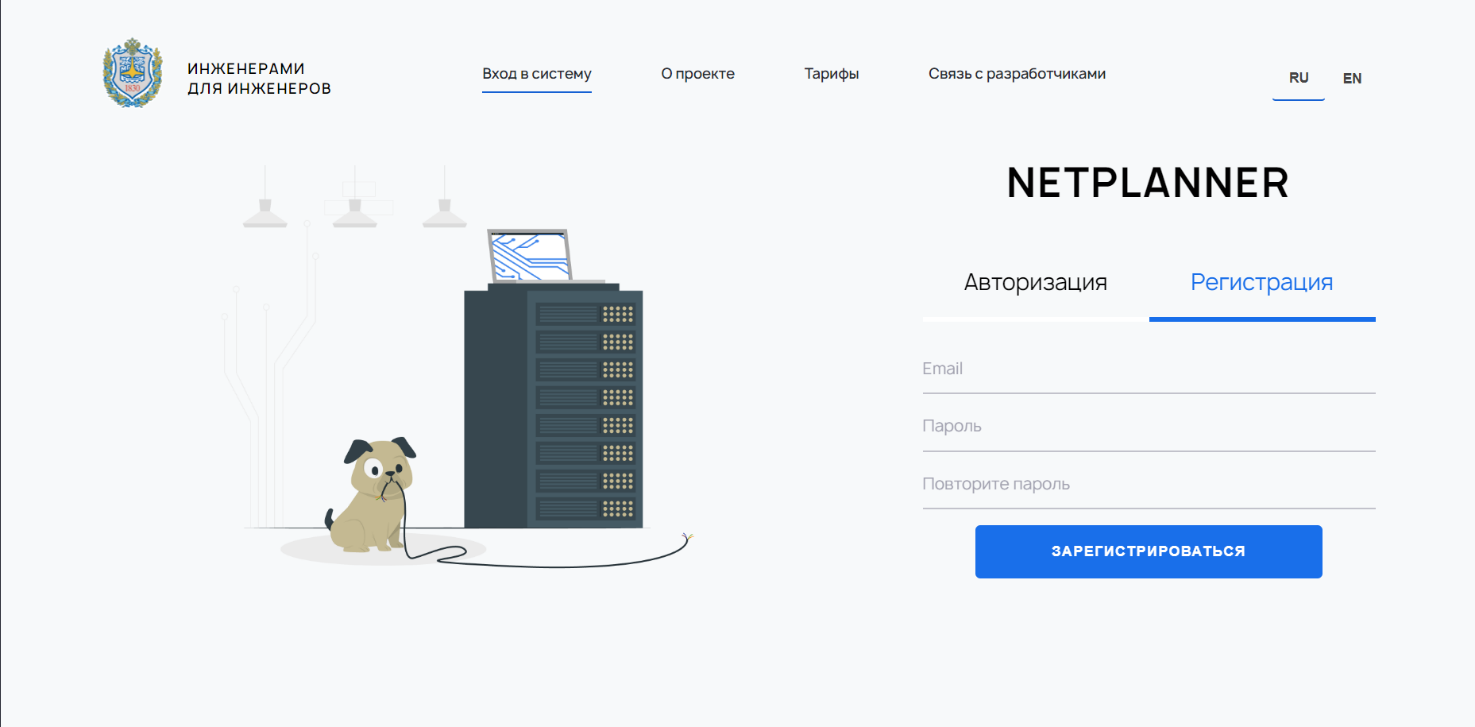


Рис. 17. Окно регистрации

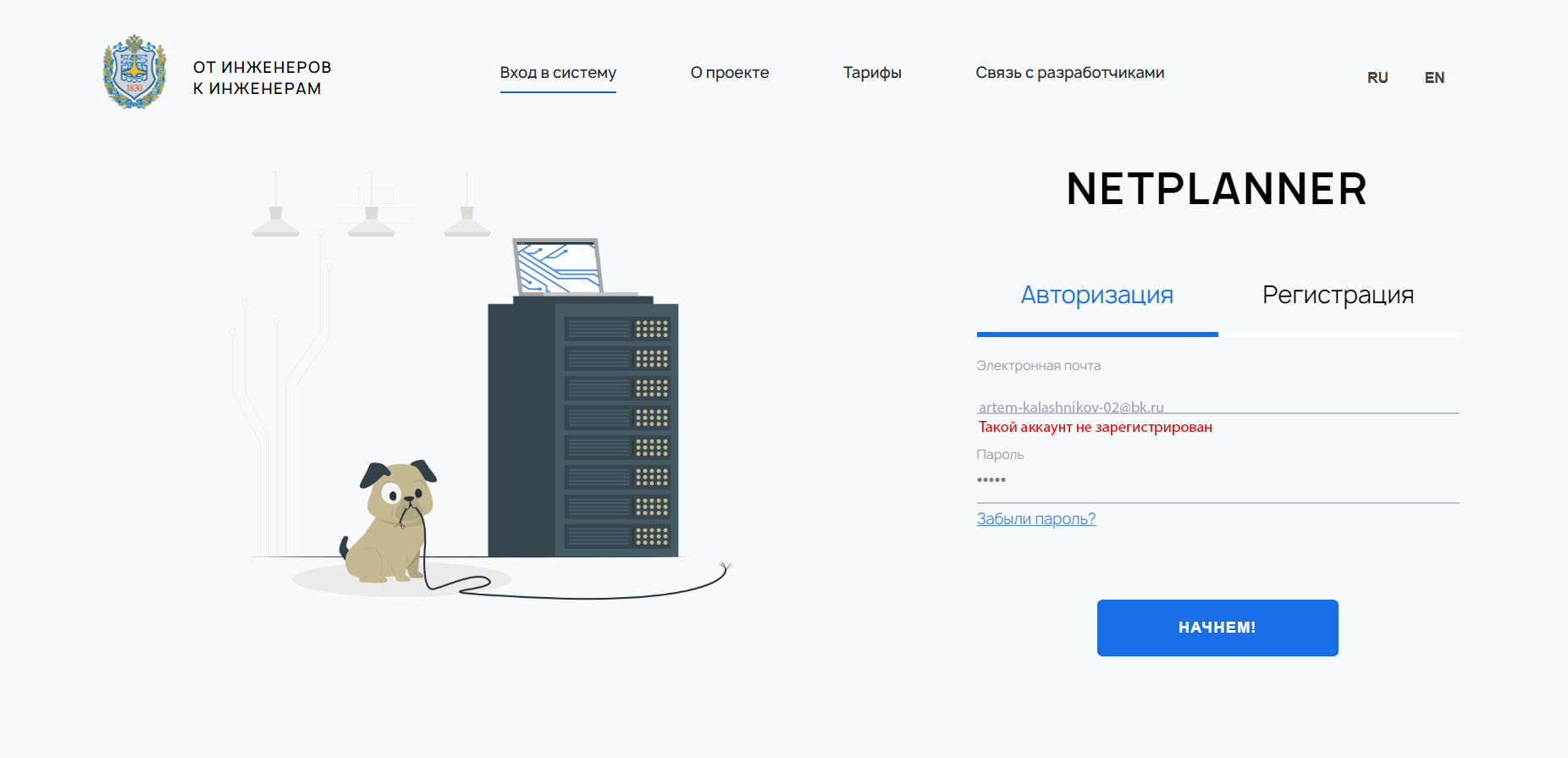
После выполнения регистрации в целях безопасности и проверки почты на почту будет отправлено сообщение с ссылкой на прохождение верификации (см. рис. 18).



Рис. 18. Сообщение о подтверждение почты

Ссылка будет действительна два дня, если же за два дня ссылка так и не будет использована, то пользователь так и не будет зарегистрирован.

И при попытке попробовать авторизоваться выдаст ошибку. Также будет выводить ошибку при попытке авторизоваться с неправильным (см. рис. 19).

Рис. 19. Ошибка

Также выполняется проверка email и пароля (см. рис. 20).

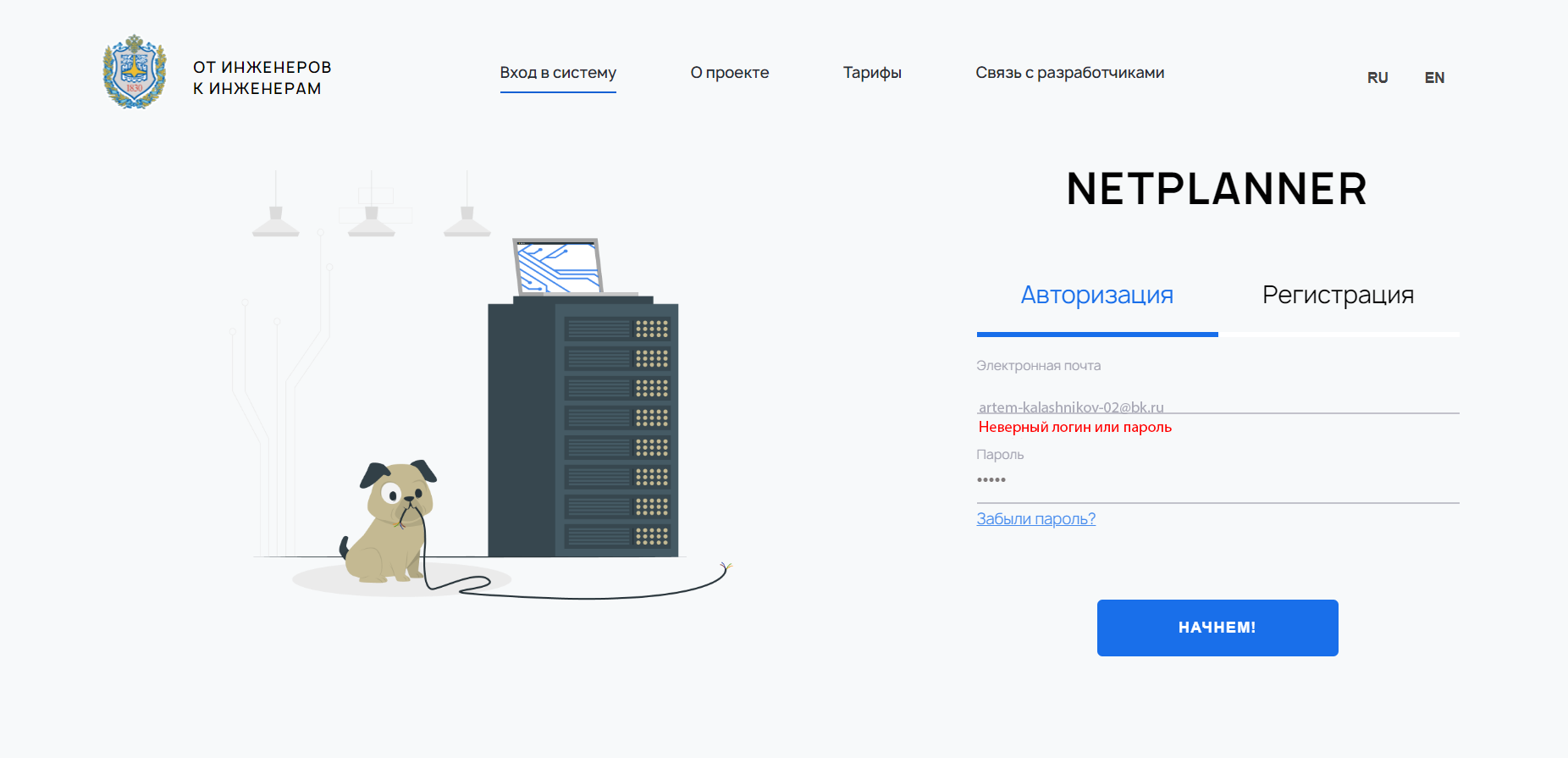


Рис. 21. Ошибка. Неверный логин

Добавлена проверка валидации полей при некорректном email выводит ошибку (см. рис. 22).

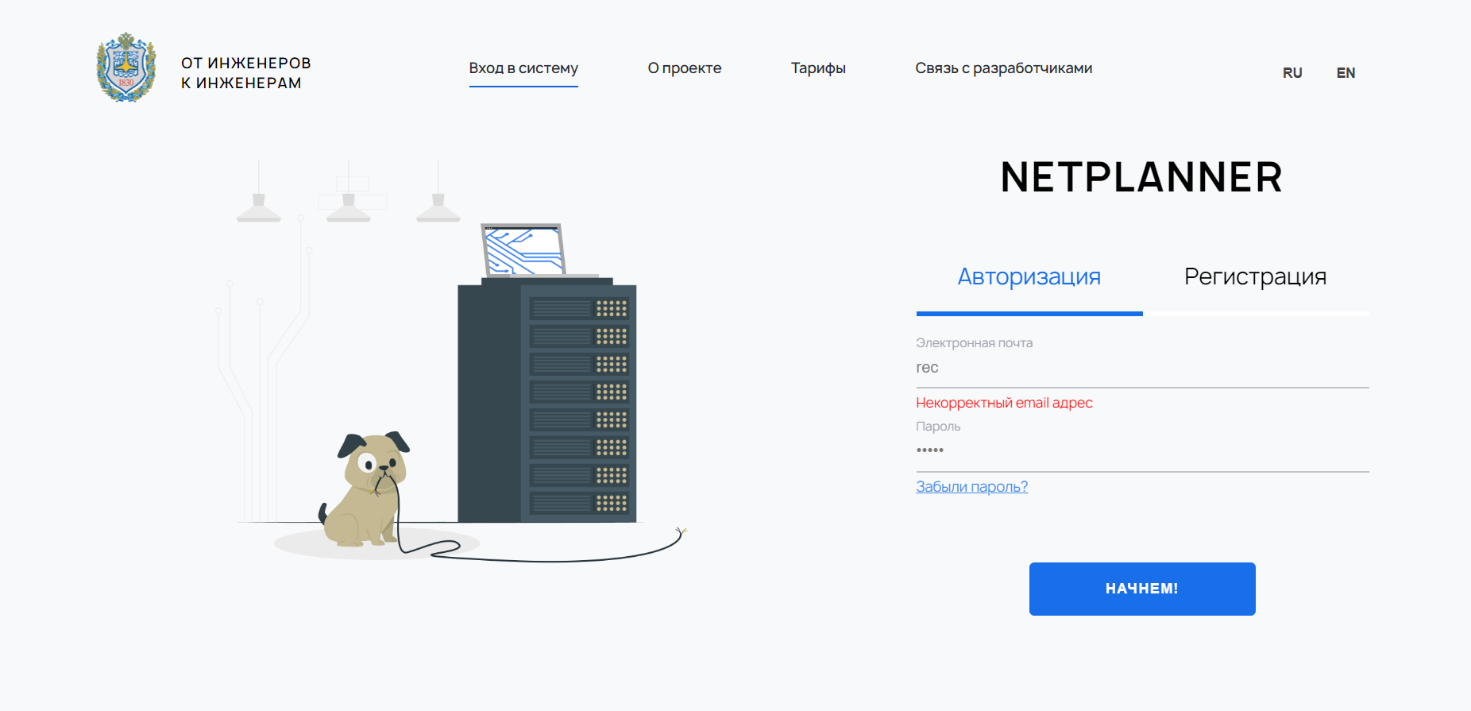


Рис. 19. Ошибка. Неверный логин

Для переключения языка, в правом верхнем углу нужно выбрать язык, который нам подходит и выполниться перевод (см. рис. 22).

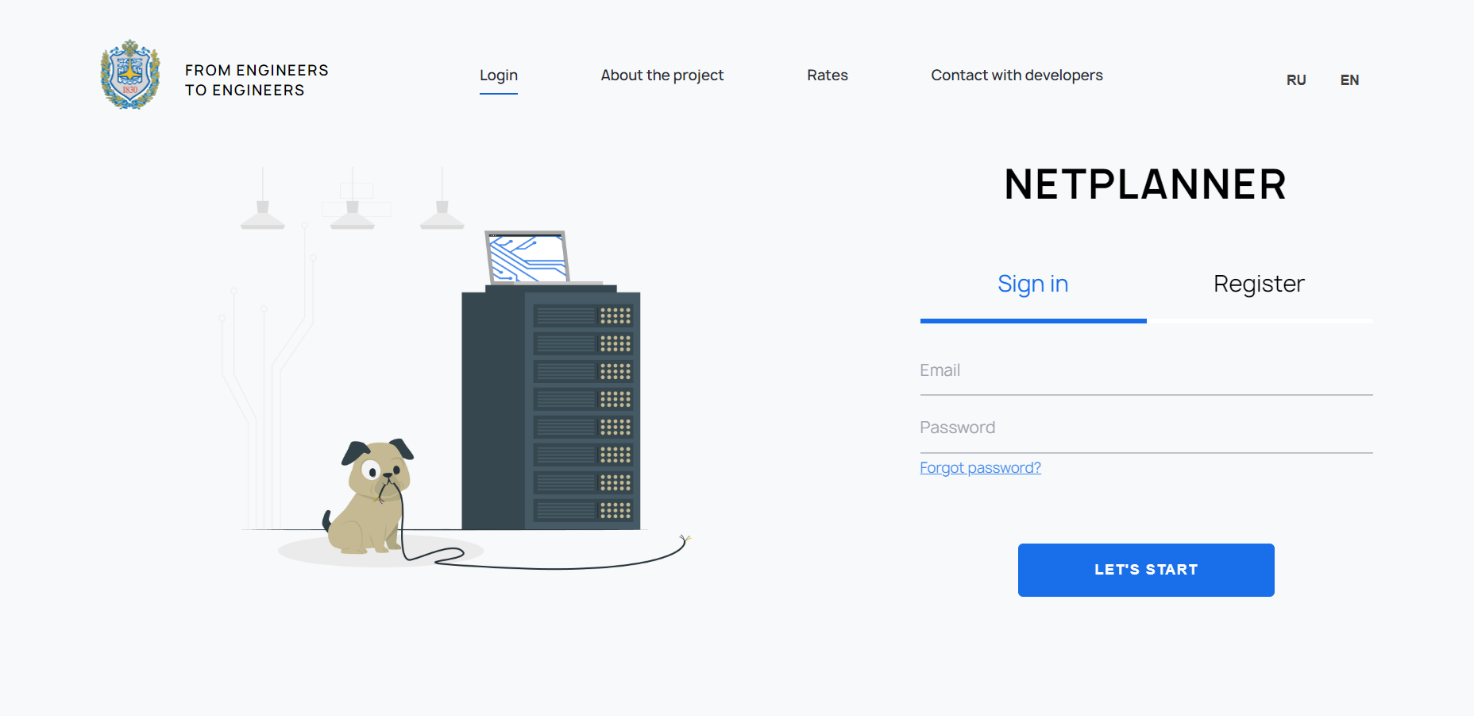
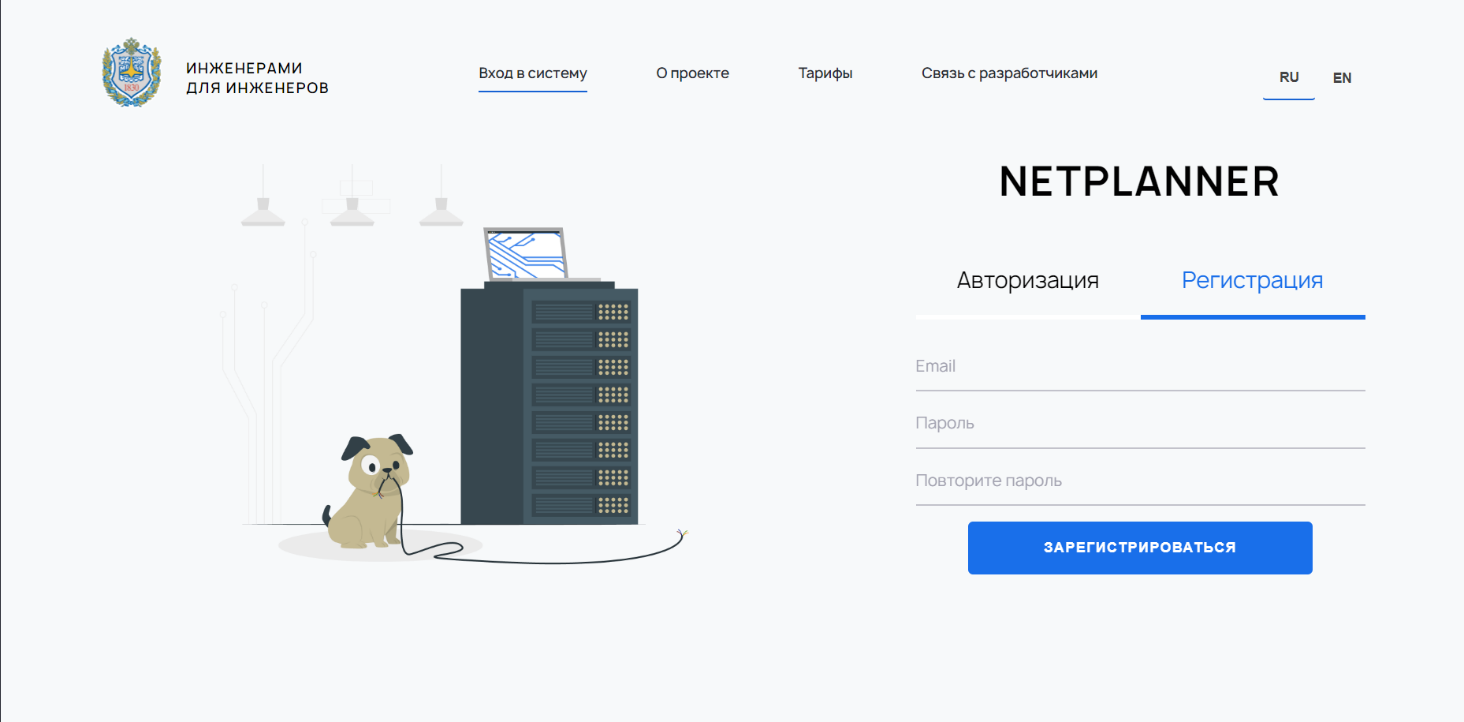
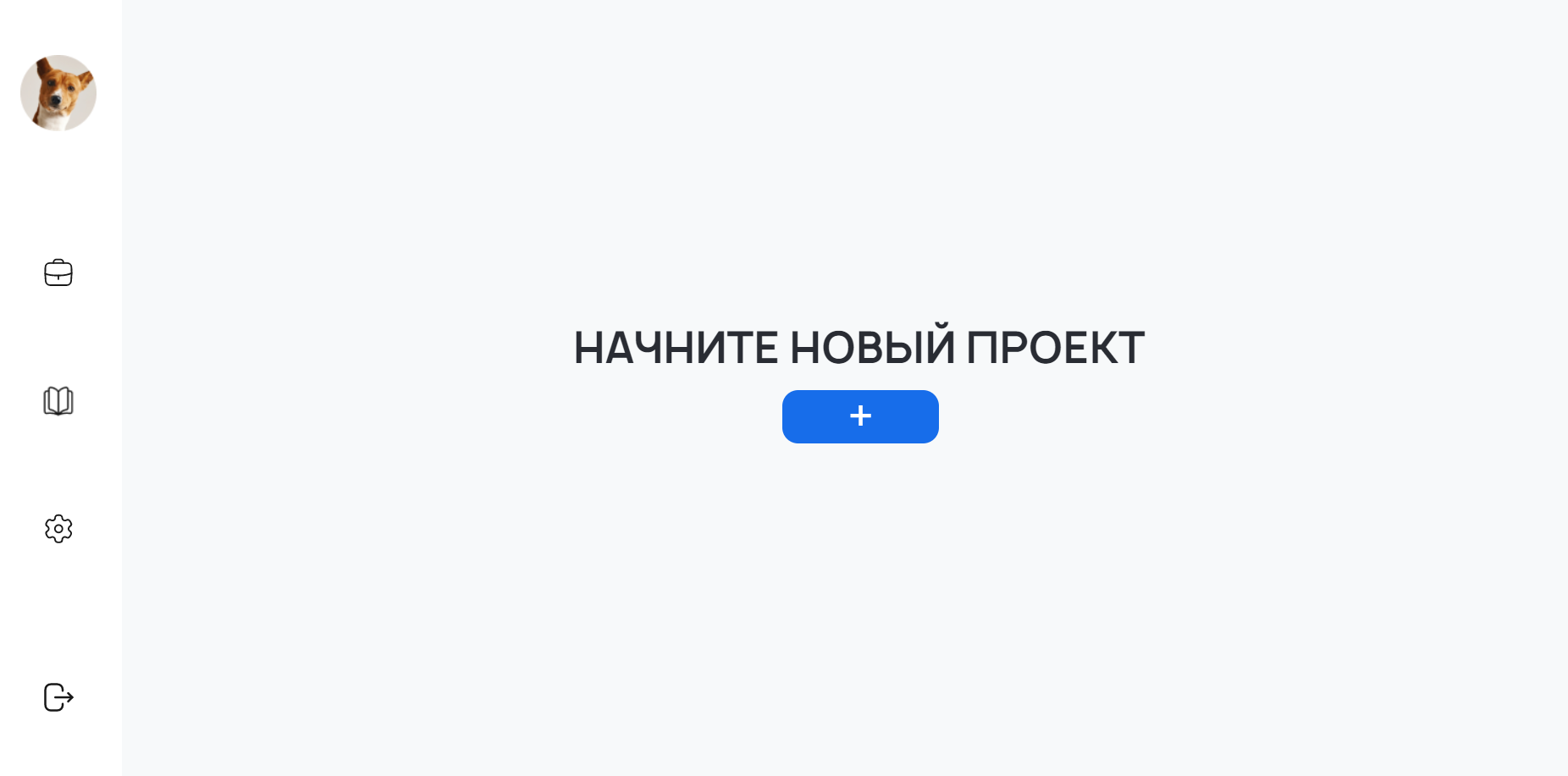


Рис. 22. Выполнение перевода

Как только мы пройдем авторизацию, мы переходим в главное окно нашего веб-приложения (см. рис. 23)

 Рис. 23. Главное окно

Для изменения данных пользователя мы перейдем в настройки где можем поправить все наши данные (см. рис. 24)



Рис. 24 Настройки пользователя

ЗАКЛЮЧЕНИЕ

В ходе выполнения курсового проекта были получены обширные навыки в сфере разработки веб-приложений, работы с ООП, проектирования и реализации продукта.

Результатом выполнения курсовой работы была создана система, позволяющая производить регистрацию и авторизацию пользователей, производить настройки аккаунта. Благодаря jwt и RSA система в дальнейшем может взаимодействовать с любыми микросервисами. Результат соответствует всем заранее определенным требованиям технического задания. Внедрение данной системы в приложение позволит увеличить безопасность и предотвращает несанкционированный доступ к ресурсам.

Поскольку поставленная задача была выполнена, в дальнейшем планируется перейти к использованиям баз данных.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Джон Скит. C#: программирование для профессионалов, 2-е издание = C# in Depth, 2nd Edition. — М.: «Вильямс», 2011. — 544 с.
2. Герберт Шилдт. C# 4.0: полное руководство = C# 4.0 The Complete Reference. — М.: «Вильямс», 2010. — С. 1056.
3. Кристиан Нейгел, Карли Уотсон и др. Visual C# 2010: полный курс = Beginning Microsoft Visual C# 2010. — М.: Диалектика, 2010.
4. Эндрю Троелсен. Язык программирования C# 2010 и платформа .NET 4.0 = Pro C# 2010 and the .NET 4.0 Platform. — 5-е изд. — М.: Вильямс, 2010. — С. 1392.
5. http://msdn.microsoft.com/ru-ru/library

Дополнительная литература

1. Мейер, Б. Объектно-ориентированное программирование и программная инженерия [Электронный ресурс] / Б. Мейер. — 3-е изд. — Электрон. текстовые данные. — М. : Интернет-Университет Информационных Технологий (ИНТУИТ), Ай Пи Эр Медиа, 2019. — 285 c. — 978-5-4486-0513-0. — Режим доступа: <http://www.iprbookshop.ru/79706.html>
2. Кратчен Ф. Введение в Rational Unified Process. 2-е изд.: Пер. с англ. – М.: Вильямс, 2002. – Глава 9. Режим доступа: https://www.iprbookshop.ru/125191
3. Вендров А. М. Проектирование программного обеспечения экономических информационных систем. 2-е изд. – М.: Финансы и статистика, 2005. https://www.iprbookshop.ru/125718
4. Рамбо Дж., Блаха М. UML 2.0. Объектно-ориентированное моделирование и разработка. 2-е изд.: Пер. с англ. – СПб.: Питер, 2007 https://www.iprbookshop.ru/125136
5. Боггс У., Боггс М., UML и Rational Rose 2002, ЛОРИ, 2004 https://www.iprbookshop.ru/102400
6. Логанов, С. В. Объектно-ориентированное программирование : учебное пособие для СПО / С. В. Логанов, С. Л. Моругин. — Саратов, Москва : Профобразование, Ай Пи Ар Медиа, 2022. — 215 c. — Режим доступа: <https://www.iprbookshop.ru/118969>
7. Букунов, С. В. Объектно ориентированное программирование на языке Python : учебное пособие / С. В. Букунов, О. В. Букунова. — Санкт-Петербург : Санкт-Петербургский государственный архитектурно-строительный университет, ЭБС АСВ, 2020. — 119 c. — Режим доступа: <https://www.iprbookshop.ru/117194>
8. Зыков, С. В. Введение в теорию программирования. Объектно-ориентированный подход : учебное пособие / С. В. Зыков. — 3-е изд. — Москва : Интернет-Университет Информационных Технологий (ИНТУИТ), Ай Пи Ар Медиа, 2021. — 187 c. — Режим доступа: <https://www.iprbookshop.ru/102007>
9. Розенберг, Д. Применение объектного моделирования с использованием UML и анализ прецедентов на примере книжного Internet-магазина / Д. Розенберг, К. Скотт ; перевод А. А. Слинкина. — 2-е изд. — Москва : ДМК Пресс, 2022. — 159 c.
10. Носова, Л. С. Case-технологии и язык UML : учебно-методическое пособие / Л. С. Носова. — 2-е изд. — Челябинск, Саратов : Южно-Уральский институт управления и экономики, Ай Пи Эр Медиа, 2019. — 67 c. — Режим доступа: https://www.iprbookshop.ru/81479
11. Дейт, К. Дж. SQL и реляционная теория/ К. Дж. Дейт. – М.: Издательский дом «Вильямс», 2015. – 474 с.
12. Visual Studio 2017 [Электронный ресурс]. – 2003-2017. - Электрон дан. -– Режим доступа: https://www.visualstudio.com/ru-ru/visual-studio-homepage-vs.aspx
13. Вирт, Никлаус. Алгоритмы и структуры данных/ Н. Вирт. – СПб.: БХВ-Петербург, 2015. – 274 с. Режим доступа: https://www.iprbookshop.ru/92343
14. Кватрани Т. RationalRose 2000 и UML. Визуальное моделирование/ Т. Кватрани. – М.: Издательский дом «Вильямс», 2013. – 583 с. Режим доступа: https://www.iprbookshop.ru/567
15. Кумар, В. NET Сетевое программирование/ В. Кумар, Э. Кровчик, Н. Лагари. – М.: Лори, 2014. – 500 с.: ил. Режим доступа: https://www.iprbookshop.ru/4433
16. Рихтер, Дж. Программирование на платформе Microsoft .NET Framework 4.0 на языке C#/ Дж. Рихтер. – 3-е издание. – СПб.: Питер, 2014. – 400 с.: ил.
17. Микелсен К. Язык программирования C#. Лекции и упражнения/ К. Микелсен. – М.: Издательский дом «ДиаСофт», 2015. – 656 с. Режим доступа: https://www.iprbookshop.ru/8123
18. Петзольд Ч. Программирование для Microsoft Windows на С#. В 2 томах/ Ч. Петзольд. – СПб.: Русская редакция, 2013. – 536 с. Режим доступа: https://www.iprbookshop.ru/81472