Министерство науки и высшего образования Российской Федерации

НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ им. Р.Е.АЛЕКСЕЕВА

Кафедра прикладной математики

Планировщик полетов

По дисциплине Технологии программирования

Выполнили:

Корноухов Илья

Смородин Дмитрий

Кабанов Дамир

Субботин Николай

Нижний Новгород

2024

**Содержание**

[Содержание 2](#_Задача:)

[Цели и задачи 3](#_Задача:)

[Описание и идея проекта 4](#_Теоретическая_справка)

[Архитектура приложения 10](#_Реализация)

[Тестирование приложения 12](#_Результат_работы_программы)

**Цели и задачи**

Цели:

* Придумать идею
* Разработать приложения для управления полетами самолетов
* Распределить задачи по написанию программы
* Провести код review
* Разработать документацию
* Разработать автоматизированные тесты

Задачи:

* Разработать архитектуру приложения
* Реализовать программы на языке с++ с использованием cmake согласно архитектуре
* Реализовать автоматизированные тесты с помощью ctest
* Разработать документацию

**Описание и идея проекта**

**Название проекта:** Планировщик полетов.

**Идея проекта:** клиент-серверное приложение для управления авиарейсами.

**Возможности пользователя:**

1. Просмотр, создание, редактирование и удаление аэропортов
2. Просмотр, создание, редактирование и удаление самолетов
3. Просмотр и редактирование диспетчеров
4. Просмотр и создание полетов
5. Пропуск и просмотр текущего времени

**Реализация приложения**

Серверная часть приложения состоит из 4 сервисов

* database-service – предоставляет доступ к базе данных по протоколу http. Написан на языке Java с использованием фреймворка Spring, системы сборки Gradle, библиотеки log4j (логгирование), swagger2 (генерация документации API), flyway (миграции базы данных) и прочие.
* identity-service – предоставляет возможность регистрации и авторизации по JWT. Написан на языке Java с использованием фреймворка Spring, системы сборки Gradle, библиотеки log4j (логгирование), swagger2 (генерация документации API), jsonwebtoken (JWT) и прочие.
* plane-service – содержит логику работы программы. Написан на языке C++ с использованием, системы сборки CMake, библиотеки json (конвертация json в объекты), cpp-httplib (прием и отправление http запросов).
* gateway – перенаправляет запросы к сервисам. Написан на языке Java с использованием фреймворка Spring, системы сборки Gradle, библиотеки log4j (логгирование) и прочие.

В качестве базы данных использована PostgreSQL.

Сборка и запуск осуществляется с помощью Docker compose.

Клиент написан на языке C++ с использованием, системы сборки CMake, библиотеки json (конвертация json в объекты), cpp-httplib (отправление http запросов).

Состояния программы:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Состояние** | **Пользователь** | **Клиент** | **Сервер** | **Полученное состояние** |
| Стартовое | Ввод данных для авторизации | Получение токена и сохранение его в файл | Генерация токена | Выбор группы действий |
| Выбор группы действий | «Работа с аэропортами» | Вывод действий с аэропортами | - | Выбор действий с аэропортами |
| «Работа с самолетами» | Вывод действий с самолетами | - | Выбор действий с самолетами |
| «Работа с полетами» | Вывод действий с полетами | - | Выбор действий с полетами |
| «Работа с диспетчерами» | Вывод действий с диспетчерами | - | Выбор действий с диспетчерами |
| Выбор действий с аэропортами | «Получить» | Вывод всех аэропортов | Получение всех аэропортов | Выбор группы действий |
| «Создать» | Получение от пользователя данных для создания | Сохранение нового аэропорта | Выбор группы действий |
| «Редактировать» | Получение от пользователя данных для редактирования | Изменение аэропорта | Выбор группы действий |
| «Удалить» | Получение id аэропорта для удаления | Удаление аэропорта | Выбор группы действий |
| Выбор действий с самолетами | «Получить» | Вывод всех самолетов | Получение всех самолетов | Выбор группы действий |
| «Создать» | Получение от пользователя данных для создания | Сохранение нового самолета | Выбор группы действий |
| «Редактировать» | Получение от пользователя данных для редактирования | Изменение самолета | Выбор группы действий |
| «Удалить» | Получение id аэропорта для удаления | Удаление самолета | Выбор группы действий |
| Выбор действий с полетами | «Получить» | Вывод всех полетов | Получение всех полетов | Выбор группы действий |
| «Создать» | Получение от пользователя данных для создания | Сохранение нового самолета | Выбор группы действий |
| Выбор действий с диспетчерами | «Получить» | Вывод всех диспетчеров | Получение всех диспетчеров | Выбор группы действий |
| «Получить по id» | Вывод диспетчера с нужным id | Получение диспетчера с нужным id | Выбор группы действий |
| «Редактировать» | Получение от пользователя данных для редактирования | Изменение диспетчера | Выбор группы действий |
| Выбор действий с временем | «Получить» | Вывод текущего времени | Получение текущего | Выбор группы действий |
| «Пропустить» | Получение от пользователя времени для пропуска | Добавление указанного времени к текущему | Выбор группы действий |
| Выход | - | Завершение выполнения программы | - | - |

**Тестирование приложения**

Для тестирования приложения составлен автоматический тест проверяющий работоспособность слоя репозиториев в клиенте.

**Автоматический тест**

Для автоматического тестирования была написана программа, использующая классы репозиториев для отправления запросов и проверки на ошибки.

**Заключение**

В ходе выполнения работы было создано приложение для управления полетами самолетов, изучена система сборки CMake и система автоматического тестирования CTest, получены практические навыки командной разработки ПО.