Правительство Санкт-Петербурга

Комитет по науке и высшей школе

Санкт-Петербургское государственное бюджетное

профессиональное образовательное учреждение

«Политехнический колледж городского хозяйства»

**КУРСОВОЙ ПРОЕКТ**

**по профессиональному модулю ПМ 11**

**«Разработка, администрирование и защита баз данных»**

**(МДК 11.01 «Технология разработки и защиты баз данных»)**

Специальность 09.02.07 Информационные системы и программирование

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ИС Автозаправочные станции\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*тема курсового проекта (работы)*

**Студент** **группы** \_ИП-19-7К\_\_\_

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

Ф.И.О.

**Руководитель**

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

Ф.И.О.

**Председатель П(Ц)К**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(**Л.В. Левит)\_

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

«\_\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2022 г.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*подпись*   *расшифровка*

Санкт-Петербург

2022 г.

Правительство Санкт-Петербурга

Комитет по науке и высшей школе

Санкт-Петербургское государственное бюджетное профессиональное образовательное учреждение

«Политехнический колледж городского хозяйства»

**ОТЗЫВ**

**НА КУРСОВОЙ ПРОЕКТ (РАБОТУ)**

**по профессиональному модулю ПМ 11**

**«Разработка, администрирование и защита баз данных»**

**(МДК 11.01 «Технология разработки и защиты баз данных»)**

**Студент** Шумилов Александр Дмитриевич

**Отделение** ИТ

**Специальность** 09.02.07 Информационные системы и программирование

**Наименование темы:** ИС Автозаправочные станции

**Руководитель** Левит Л.В.

**ШКАЛА ОЦЕНКИ КУРСОВОГО ПРОЕКТА (РАБОТЫ)**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **№** | **Показатели** | **Оценка** | | | |
| **5** | **4** | **3** | **0\*** |
| **Справочно-информационные** | 1 | Соответствие представленного материала техническому заданию |  |  |  |  |
| 2 | Раскрытие актуальности тематики работы |  |  |  |  |
| 3 | Степень полноты обзора состояния вопроса |  |  |  |  |
| 4 | Уровень и корректность использования в работе методов исследований, математического моделирования |  |  |  |  |
| 5 | Степень комплексности работы, применение в ней знаний естественнонаучных, социально-гуманитарных, экономических, обще- профессиональных и специальных дисциплин |  |  |  |  |
| 6 | Использование информационных ресурсов Internet |  |  |  |  |
| 7 | Использование современных пакетов компьютерных программ и технологий |  |  |  |  |
| **Творческие** | 8 | Оригинальность и новизна полученных результатов, научных, конструкторских и технологических решений |  |  |  |  |
| 9 | Ясность, четкость, последовательность и обоснованность изложения |  |  |  |  |
| **Оформление** | 10 | Уровень оформления пояснительной записки: |  |  |  |  |
| - общий уровень грамотности |  |  |  |  |
| - стиль изложения |  |  |  |  |
| - качество иллюстраций |  |  |  |  |
| - соответствие требованиям стандарта |  |  |  |  |
| 11 | Объем и качество выполнения графического материала, его соответствие тексту записки и стандартам (ЕСКД; ЕСТД). |  |  |  |  |
| **ИТОГОВАЯ ОЦЕНКА** | | |  | | | |

\* - не оценивается (трудно оценить)

**В ходе работы над курсовым проектом (работой)** студент продемонстрировал умения, знания и навыки в соответствии с профессиональными компетенциями, представленными

в *МДК 11.01* *ПМ 11* ФГОС СПО

**Заключение руководителя**: Считаю, что представленный курсовой проект (работа) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*соответствует / не соответствует*

заявленной теме и заданию, и заслуживает оценки \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*подпись* *ФИО*

«\_\_\_\_» «\_\_\_\_\_\_\_\_\_\_\_\_\_\_» 2022 г.

**СОГЛАСОВАНО**

предметной (цикловой) комиссией 09.02.07

Протокол №2 от 12.12.2022г

Председатель П(Ц)К Л.В. Левит

**ЗАДАНИЕ НА КУРСОВОЙ ПРОЕКТ (РАБОТУ)**

**по профессиональному модулю ПМ 11**

**«Разработка, администрирование и защита баз данных»**

**(МДК 11.01 «Технология разработки и защиты баз данных»)**

Специальность\_\_09.02.07 Информационные системы и программирование

*№ и наименование специальности*

Тема \_\_\_\_\_\_\_\_ ИС Автозаправочные станции\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Студент \_\_\_\_\_\_\_\_\_\_\_Шумилов Александр Дмитриевич\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*ФИО*

Группа \_\_\_\_\_ИП-19-7К\_\_\_ Курс \_\_\_\_\_4 курс\_\_\_\_\_\_ Форма обучения: очная

Руководитель\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*подпись ФИО*

Время выполнения проекта с « 26 » сентября 2022г. по « 23 » декабря 2022г.

Дата выдачи задания на курсовой проект/работу «\_26\_» \_сентября\_\_2022г.

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Шумилов Александр Дмитриевич

*подпись*   *ФИО*

**ПЕРЕЧЕНЬ ВОПРОСОВ, ПОДЛЕЖАЩИХ РАЗРАБОТКЕ**

| **Содержание задания по профилирующим разделам курсового проекта** | **ПМ\*** | **МДК\*** | **ПК\*** | **ОК\*** |
| --- | --- | --- | --- | --- |
| **1. Техническое задание**  Введение | ПМ 11 | МДК 11.01 | .  11.1-11.5 | 1-9 |
| 1.1. Основания для разработки |
| 1.2. Назначение разработки |
| 1.3.Требования к программе:Требования к функциональным характеристикамТребования к надёжностиУсловия эксплуатацииТребования к составу и параметрам технических средствТребования к информационной и программной совместимости |
| 1.4.Требования к программной документации |
| 1.5. Стадии и этапы разработки |
| 1.6. Перечень графических материалов |
| 1. **Пояснительная записка**   Аннотация  Содержание  Введение | ПМ 11 | МДК 11.01 | 11.1-11.5 | 1-9 |
| * 1. Назначение и область применения | 11.1-11.3 |
| * 1. Постановка задачи |
| * 1. Описание программы |
| * 1. Программа и методика испытаний |
| * 1. Руководство пользователя |
| 6. Мероприятия по информационной безопасности |
| Заключение  Приложения. Текст программы  Источники, используемые при разработке | 11.1-11.5 |

**Графическая часть**

1. Оформить все рисунки и таблицы в соответствии с ГОСТами;
2. В руководстве оператора все действия продублировать иллюстрациями.

**Заключение**

1. Прописать выводы и предложения;
2. Отразить результаты практической значимости;
3. Прописать предложения по совершенствованию и модернизации программного продукта в дальнейшем.

**Примечание:**

**ПМ –** профессиональный модуль;

**МДК** – междисциплинарный курс;

**ПК** – профессиональные компетенции;

**ОК** – общие компетенции.

**\* -** смотреть Приложение А.

Дополнительные указания представлены в Приложение Б.

Тема курсового проекта рассмотрена на заседании П(Ц)К спец. 09.02.07

Время выполнения проекта с « 26 » сентября 2022г. по « 23 » декабря 2022г.

Руководитель курсового проекта: \_\_\_\_\_\_\_\_\_\_

Правительство Санкт-Петербурга

Комитет по науке и высшей школе

Санкт-Петербургское государственное бюджетное

профессиональное образовательное учреждение

«Политехнический колледж городского хозяйства»

УТВЕРЖДЕН

ПКГХ 09.02.07 ИП-19-7К .089-19 - ЛУ

# КУРСОВОЙ ПРОЕКТ

ИС Автозаправочные станции

## Техническое задание

ПКГХ 09.02.07 ИП-19-7К. 089-19 ТЗ

Листов 14

СОДЕРЖАНИЕ

[Введение 3](#_Toc116908249)

[1. Основания для разработки 4](#_Toc116908250)

[2. Назначение разработки 5](#_Toc116908251)

[3. Требования к программе или программному изделию 6](#_Toc116908252)

[3.1. Требования к функциональным характеристикам 6](#_Toc116908253)

[3.2. Требования к надёжности 7](#_Toc116908254)

[3.3. Условия эксплуатации 8](#_Toc116908255)

[3.4. Требования к составу и параметрам технических средств 9](#_Toc116908256)

[3.5. Требования к информационной и программной совместимости 10](#_Toc116908257)

[3.6. Требования к программной документации 11](#_Toc116908258)

[4. СТАДИИ И ЭТАПЫ РАЗРАБОТКИ 12](#_Toc116908259)

[4.1. Стадия разработки «Технический проект» 12](#_Toc116908260)

[4.2. Стадия разработки «Рабочий проект» 12](#_Toc116908261)

[4.3. Стадия разработки «Внедрение» 12](#_Toc116908262)

[5. Мероприятия по информационной безопасности 13](#_Toc116908263)

[6. Порядок контроля и приемки 1](#_Toc116908264)3

[7. Перечень графических материалов 1](#_Toc116908265)4

* 1. Введение

Любое автозаправочное учреждение сталкивается с проблемой привлечения новых клиентов и распространением информации об расположении своих станций между клиентами. Руководствам автозаправочных станций нужно тратить огромные деньги на оплату труда маркетологов для распространения информации. Это сказывается на ежегодной выручке компаний.

Использование информационной системы поможет распространению информации об расположении станций и новостей, а также привлечению новых клиентов путём использования различных скидок и акций в личном кабинете.

В настоящее время большинство информационных систем в России являются оригинальными разработками различных компаний, методика работы с такими системами может сильно различаться. Такие системы часто бывают специфичны в использовании, имеют уникальный интерфейс, что затрудняет поиск информации, а также – плохо интегрируемы друг с другом.

Для решения вышеизложенной проблемы предлагается разработать мобильную систему электронного онлайн мониторинга автозаправочных станций.

1. Основания для разработки

Программа создается на основе задания по предмету МДК 11.01, утверждённого преподавателем Левит Л.В. 17 октября 2022 года.

1. Назначение разработки

Программа «ИС Автозаправочные станции» предназначено для следующих задач:

1. Создание/ввод, систематизация, использование и хранение данных о ходе автозаправочных станциях и их расположений.
2. Просмотр всех автозаправочных станций на карте.
3. Показ новостей.
4. Просмотр цен на топливо и список доступного топлива на заправках.
5. Требования к программе или программному изделию
   1. Требования к функциональным характеристикам

Программа должна соответствовать следующему макету:



Рисунок 1 – Карта

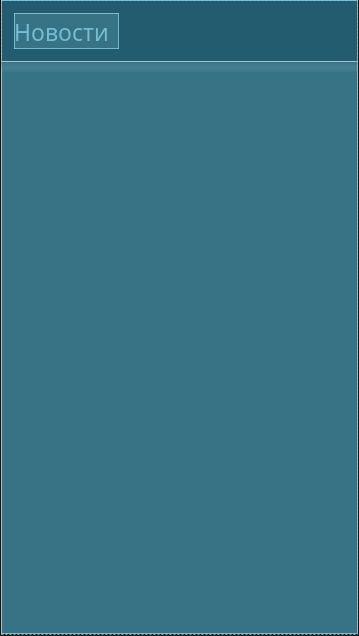


Рисунок 2 – Новости

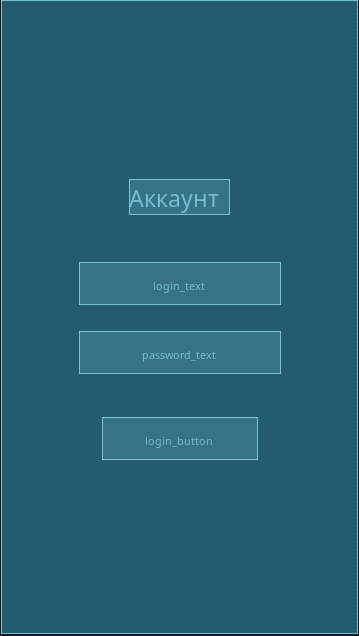


Рисунок 3 – Вход в аккаунт



Рисунок 4 – Информация об автозаправочной станции

Требования к функциональным характеристикам, следующие:

1. Программа должна осуществлять следующие функции: вход в учетную запись, показ карты, показ информации о станциях, показ новостей, показ цен на топливо.
2. Программа должна реализовывать возможность отображения станций на интерактивной карте.
3. Входными данными является авторизация пользователя.
4. Выходными данными является информация из электронного журнала
   1. Требования к надёжности

В случае возникновения каких-либо ошибок пользователю должно выводиться соответствующее сообщение на экран телефона.

Осуществить возможность редактирования данных только администратору.

* 1. Условия эксплуатации

Программа будет прекрасно работать от минус 30 до плюс 45 °C при относительной влажности 90 % и атмосферном давлении 462 мм.рт.ст., поскольку такие условия приблизительно соответствуют условиям эксплуатации современных мобильных телефонов.

* 1. Требования к составу программных и технических средств

Для корректной работы программного обеспечения необходимо наличие следующих технических средств:

1. Операционная система Android 10 и выше;
2. Объем ОЗУ – 2 ГБ и более;
3. Объём памяти – 8 ГБ и более;

Хостинг, на котором находятся файлы программного обеспечения, должен отвечать следующим требованиям:

* Защита от DDoS атак;
* Возможность получения выделенного IP-адреса;
* Объём памяти не менее 20 ГБ;
* Объём ОЗУ не менее 2 ГБ;
* Установленная Docker система;
* Запущенные базы данных MongoDB и PostgreSQL
  1. Требования к информационной и программной совместимости

Программное обеспечение требуется разработать с помощью:

* Программного языка Kotlin;
* Интегрированной среды разработки Android Studio и IntelliJ IDEA;

База данных на сервере должна быть реализована в среде PostgreSQL и MongoDB.

Для связи серверной базы данных с программой должен быть реализован REST API.

В состав программного обеспечения должны входить:

* Операционная система Android 10 и выше;
* Для работоспособности программного обеспечения необходим доступ в интернет.
  1. Требования к программной документации

Должна разработана пояснительная записка (с приложениями).

1. СТАДИИ И ЭТАПЫ РАЗРАБОТКИ
   1. Стадия разработки «Технический проект»

* Разработка структуры и определение формы представления входных и выходных данных;
* Разработка алгоритма решения задачи;
* Разработка структуры программы;
* Окончательное определение конфигурации технических средств;
* Разработка пояснительной записки;
* Согласование и утверждение технического проекта.
  1. Стадия разработки «Рабочий проект»
* Программирование и отладка программы;
* Разработка программных документов в соответствии с требованиями ГОСТ 19.101 – 77;
* Разработка и согласование программы и методики испытаний;
* Корректировка программы и программной документации по результатам испытаний.

1. Мероприятия по информационной безопасности

В целях обеспечения информационной безопасности необходимо обеспечить:

* Разграничение доступа к информации серверной БД;
* Сохранение доступности информации, предназначенной для пользователя.

1. Перечень графических материалов

Для данной разработки следует подготовить следующие графические материалы:

* Диаграмму классов программы;
* Схему базы данных.

Правительство Санкт-Петербурга

Комитет по науке и высшей школе

Санкт-Петербургское государственное бюджетное

профессиональное образовательное учреждение

«Политехнический колледж городского хозяйства»

УТВЕРЖДЕН

ПКГХ 09.02.07 ИП-19-7К .089-19 - ЛУ

# КУРСОВОЙ ПРОЕКТ

ИС Автозаправочные станции

## Пояснительная записка

ПКГХ 09.02.07 ИП-19-7К. 089-19.81

Листов 26

## Аннотация

Программа «ИС Автозаправочные станции» предназначено для следующих задач:

1. Вход в учетную запись, показ карты, показ информации о станциях, показ новостей, показ цен на топливо.
2. Возможность отображения станций на интерактивной карте.

## 

**Содержание**

[Введение 5](#_Toc121142203)

[1. Назначение и область применения 6](#_Toc121142204)

[2. Постановка задачи 7](#_Toc121142205)

[3. Описание программы 8](#_Toc121142206)

[**3.1 Общие сведения** 8](#_Toc121142207)

[**3.2 Функциональное назначение** 9](#_Toc121142208)

[**3.3 Описание логической структуры** 10](#_Toc121142209)

[**3.4 Используемые технические средства** 14](#_Toc121142210)

[**3.5 Вызов и загрузка** 15](#_Toc121142211)

[**3.6 Входные и выходные данные** 16](#_Toc121142212)

[4. Программа и методика испытаний 17](#_Toc121142213)

[**4.1 Объекты испытаний** 17](#_Toc121142214)

[**4.2 Цель испытаний** 17](#_Toc121142215)

[**4.3 Требования к программе** 17](#_Toc121142216)

[**4.4 Методы испытаний** 17](#_Toc121142217)

[**4.5 Тестовый пример** 18](#_Toc121142218)

[5. Руководство оператора 19](#_Toc121142219)

[**5.1 Выполнение программы** 19](#_Toc121142220)

[**5.2 Сообщение оператору** 2](#_Toc121142221)3

[6. Мероприятия по информационной безопасности 2](#_Toc121142222)4

[Приложения 2](#_Toc121142223)5

[Источники, использованные при разработке 2](#_Toc121142224)6

**Введение**

Программа «ИС Автозаправочные станции» является простым и наглядной информационной системой и включает в себя множество полезных функций для просмотра информации о автозаправочных станций.

Разработка программы «ИС Автозаправочные станции» проводилось на основе следующих документов:

1) Техническое задание.

**1. Назначение и область применения**

Программа предназначена для:

1. Просмотра автомобильных заправок;
2. Просмотр новостей компании;
3. Просмотра цен на топливо на каждой заправке.

Использование информационной системы позволяет ускорить и упростить поиск новых автомобильных заправок в неизвестном городе или месте.

Существующими аналогами приложения являются:

* Яндекс заправки [— Просмотр автозаправок (zapravki.yandex.ru)](https://dnevnik2.petersburgedu.ru/login)

**2. Постановка задачи**

Необходимо разработать программу, которая должна соответствовать следующим требованиям:

* В программе должна присутствовать возможность авторизации;
* Все данные из базы данных должны получаться через REST запрос с API;
* На главной странице должна отображаться карта, на которой отмечены все автомобильные заправки;
* Должна присутствовать кнопка, которая переводит пользователя на страницу новостей;
* При нажатии на автомобильную заправку на карте должна появляться информация о ней.

**3.**

**Описание программы**

**3.1 Общие сведения**

Наименование программы «ИС Автозаправочные станции».

Программа «ИС Автозаправочные станции» написано на программном языке Kotlin с помощью программ Android Studio и IntelliJ IDEA.

Локальная БД разработана с помощью PostgreSQL и MongoDB.

Для полной работоспособности программы необходимо разместить БД на сервере.

Все данные в программу из базы данных должны передаваться через REST API.

**3.2 Функциональное назначение**

Программа предназначена для:

1. Просмотр новостей компании;
2. Просмотра цен на топливо на каждой заправке.
3. Просмотра автомобильных заправок;

Использование информационной системы позволяет ускорить и упростить поиск новых автомобильных заправок в неизвестном городе или месте.

**3.3 Описание логической структуры**

Ниже предоставлен алгоритм взаимодействия программы с локальным сервером.



Рисунок 5 - Взаимодействие с сервером

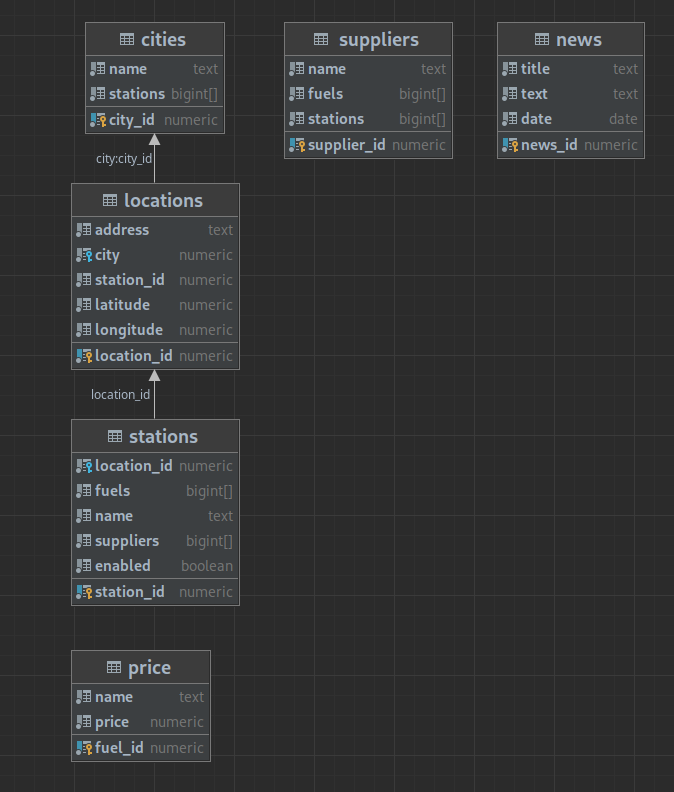


Рисунок 6 – Схема базы данных

Диаграмма прецедентов:

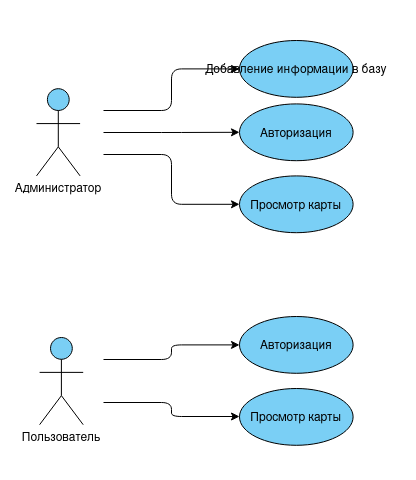


Рисунок 7 – Диаграмма прецедентов

**3.4 Используемые технические средства**

Для использования программы необходимо иметь телефон с операционной системой Android 10 и выше.

Для работоспособности программы необходим доступ в Интернет.

**3.5 Вызов и загрузка**

Для запуска программы необходимо нажать на иконку программы в меню приложений.

**3.6 Входные и выходные данные**

Входными данными в программе являются:

* Данные авторизации;
* Данные с REST запроса;

**4. Программа и методика испытаний**

**4.1 Объекты испытаний**

Объектом для испытаний является программа «ИС Автозаправочные станции» и его функционал.

**4.2 Цель испытаний**

Испытания проводятся с целью проверки соответствия проекта требованиям, указанным в техническом задании.

**4.3 Требования к программе**

Соответствие программного продукта требованиям, указанным в техническом задании.

**4.4 Методы испытаний**

Испытания проводятся в следующем порядке:

* Тестирование главной страницы (Карты) – Главная страница должна быть исправна, все элементы, расположенные на ней, должны корректно отвечать на пользовательские команды;
* Тестирование страницы новостей – Новостная страница должна быть исправна, все элементы, расположенные на ней, должны корректно отвечать на пользовательские команды;
* Тестирование страницы с информацией о автомобильной заправки – Информационная страница должна быть исправна, все элементы, расположенные на ней, должны корректно отвечать на пользовательские команды;
* Внешний вид главного меню – Главное меню должно быть легко читаемым и интуитивно понятным;

**4.5 Тестовый пример**

|  |  |  |
| --- | --- | --- |
| **Проверяемые требования** | **Сообщения программы и**  **вводимые значения** | **Результаты** |
| Кнопка “Вход” | Логин и пароль  “Вы успешно вошли” | Успешно |
| Кнопка “Вход” | Логин и пароль  “Проверьте правильность написания логина и пароля” | Успешно |
| Кнопка “Выход” | Выход | Успешно |
| Карта | Все метки на карте отображаются | Успешно |
| Нажатие на метку | Открывается меню с информацией, карта приближается к метке и строится маршрут до неё от пользователя | Успешно |
| Кнопка «Новости» | Открывается новостная страница | Успешно |
| Кнопка «Карта» | Все страницы закрываются и открывается карта | Успешно |
| Кнопка «Назад» | Закрывается текущая страница | Успешно |

Заключение тестирования.

В ходе прохождения тестирования никаких неисправностей выявлено не было. Были протестированы все основные функции программы. В ходе тестирования все элементы работали исправно и никаких ошибок не происходило.

**5. Руководство оператора**

**5.1 Выполнение программы**

Для запуска программы необходимо открыть нажать на иконку приложения в меню приложений.

После открытие программы появится карта, на которой отмечены все автомобильные заправки (Рисунок 8).

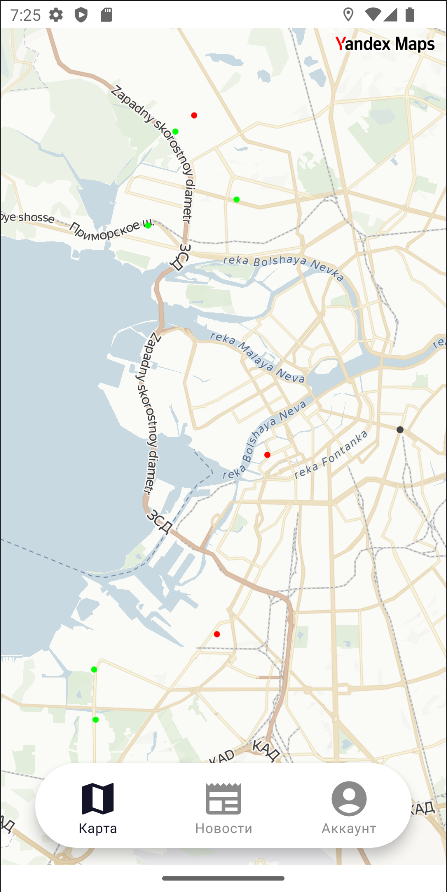


Рисунок 8 – Карта

При нажатии на метку – строится маршрут от пользователя (При условии выдачи прав на просмотр локации пользователя приложению) и открывается информационная панель с сведениями об данной автомобильной заправки (Рисунок 9). На этой форме расположены следующие элементы:

* Название автомобильной заправки
* Кнопка «Назад»
* Адрес автомобильной заправки
* Список топлива на данной заправке и его цена
* Список поставщиков



Рисунок 9 –Информация об автомобильной заправке

**5.2 Сообщение оператору**

В случае ввода неправильного логина или пароль или же отсутствие значение в соответствующих поля то пользователю будет выведено сообщение (Рисунок 12).

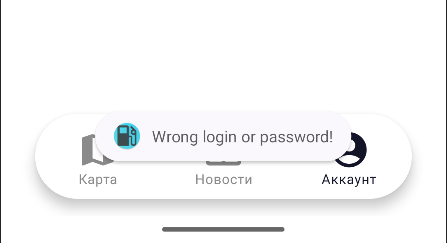


Рисунок 12 – Неверный логин или пароль

**6. Мероприятия по информационной безопасности**

Угрозами информационной безопасности для программы «ИС Автозаправочные станции» являются:

* Угрозы конфиденциальности;
* Угрозы целостности;
* Угрозы доступности;

Угроза нарушения конфиденциальности заключается в том, что информация становится известной тому, кто не располагает полномочиями доступа к ней.

К примеру, пользователь программы получит доступ к информации другого пользователя без наличия разрешения.

Данная угроза нейтрализуется с помощью ролей, которые существуют в программном коде, за счет которых каждый пользователь будет переходить только на ту форму, к которой он привязан.

Угрозы нарушения целостности — это угрозы, связанные с вероятностью модификации той или иной информации, хранящейся в информационной системе.

Для обеспечения целостности информации программы доступ к базе данных защищен с помощью обхода прямого подключения к БД через REST API (REST расшифровывается как Representational State Transfer. Отличительной особенностью сервисов REST является то, что они позволяют наилучшим образом использовать протокол HTTP).

Сам домен защищен с помощью криптографического протокола SSL.

SSL (англ. Secure Sockets Layer — уровень защищенных сокетов) — криптографический протокол, который подразумевает более безопасную связь. Он использует асимметричную криптографию для аутентификации ключей обмена, симметричное шифрование для сохранения конфиденциальности, коды аутентификации сообщений для целостности сообщений.

Нарушение доступности представляет собой создание таких условий, при которых доступ к услуге или информации будет заблокирован.

Данная угроза нейтрализуется благодаря размещению базы данных на хостинге. Так, с помощью выделенного общедоступного домена пользователь сможет получить доступ к информации в любое время.

На стороне клиента (мобильного приложения) необходимо обеспечить целостность и доступность данных для пользователя.

Целостность – свойство, при наличии которого информация сохраняет заранее определенные вид и качество.

Обеспечение целостности данных реализовано с помощью указанных ранее пунктов (сертификата SSL и аутентификации пользователя). Эти два пункта обеспечивают защиту информации от изменений, подмены или уничтожения в процессе их передачи пользователю.

**Заключение**

Программный продукт удовлетворяет всем требованиям, изложенным в техническом задании. Таким образом, задача, поставленная при проектировании программы «ИС Автозаправочные станции», выполнена.

Реализация данного программного обеспечения была произведена с помощью программы Android Studio от компании Google и JetBrains, а так же программы IntelliJ IDEA от компании JetBrains.

Программа написана на программном языке Kotlin. База данных была реализована с помощью среды PostgreSQL и в некоторых частях с помощью среды MongoDB.

Данная программа является актуальным в нынешнее время т.к. все большее и большее количество компаний, связанных с автомобильной топливной промышленностью, заинтересованы во владении своего приложения для отображения автомобильных заправок и дополнительной информации, связанной с этим.

**Текст программы**

Ниже приведен класс общения с сервером:

**object** RESTUtils {  
  
 **fun** getStations(): Set<Station> {  
 **if** (Build.VERSION.SDK\_INT > 8) {  
 **val** policy = StrictMode.ThreadPolicy.Builder()  
 .permitAll().build()  
 StrictMode.setThreadPolicy(policy)  
 **try** {  
 **val** url = URL("https://api.kiinse.me/stations/all")  
 **val** connection: HttpURLConnection = url.openConnection() **as** HttpURLConnection  
 connection.setRequestProperty("Content-Type", "application/json")  
 connection.requestMethod = "GET"  
 connection.connectTimeout = 10000  
 **if** (connection.responseCode != HttpURLConnection.HTTP\_OK)  
 **throw** IOException("Unexpected code **${**connection.responseCode**}**")  
 **val** stations = HashSet<Station>()  
 **val** array = JSONObject(AppUtils.inputStreamToString(connection.inputStream)).getJSONArray("data")  
 **for** (i **in** 0 until array.length()) {  
 **val** json = JSONObject(array[i].toString())  
 stations.add(Station(json.getInt("id"),  
 json.getBoolean("enabled"),  
 json.getString("name"),  
 getFuels(json.getJSONArray("fuels")),  
 getLocation(json.getJSONObject("location")),  
 getSuppliers(json.getJSONArray("suppliers"))))  
 }  
 **return** stations  
 } **catch** (e: Exception) {  
 AppUtils.sendErrorDialog(MainActivity.context!!, e.message)  
 }  
 }  
 **return** HashSet()  
 }  
  
 **fun** getNews(): Set<NewsData> {  
 **if** (Build.VERSION.SDK\_INT > 8) {  
 **val** policy = StrictMode.ThreadPolicy.Builder()  
 .permitAll().build()  
 StrictMode.setThreadPolicy(policy)  
 **try** {  
 **val** url = URL("https://api.kiinse.me/news/all")  
 **val** connection: HttpURLConnection = url.openConnection() **as** HttpURLConnection  
 connection.setRequestProperty("Content-Type", "application/json")  
 connection.requestMethod = "GET"  
 connection.connectTimeout = 10000  
 **if** (connection.responseCode != HttpURLConnection.HTTP\_OK)  
 **throw** IOException("Unexpected code **${**connection.responseCode**}**")  
 **val** news = HashSet<NewsData>()  
 **val** array = JSONObject(AppUtils.inputStreamToString(connection.inputStream)).getJSONArray("data")  
 **for** (i **in** 0 until array.length()) {  
 **val** json = JSONObject(array[i].toString())  
 news.add(NewsData(json.getInt("id"),  
 json.getString("title"),  
 json.getString("text"),  
 Date(json.getLong("date"))))  
 }  
 **return** news  
 } **catch** (e: Exception) {  
 AppUtils.sendErrorDialog(MainActivity.context!!, e.message)  
 }  
 }  
 **return** HashSet()  
 }  
  
 **fun** login(credentials: JSONObject): Account? {  
 **if** (Build.VERSION.SDK\_INT > 8) {  
 **val** policy = StrictMode.ThreadPolicy.Builder()  
 .permitAll().build()  
 StrictMode.setThreadPolicy(policy)  
 **val** url = URL("https://api.kiinse.me/account/register")  
 **val** connection: HttpURLConnection = url.openConnection() **as** HttpURLConnection  
 connection.setRequestProperty("Content-Type", "application/json")  
 connection.setRequestProperty("No-timeout", "kitsune")  
 connection.requestMethod = "POST"  
 connection.connectTimeout = 10000  
 AppUtils.writeToStream(connection.outputStream, credentials)  
  
 **val** responseCode = connection.responseCode  
 **if** (responseCode != HttpURLConnection.HTTP\_OK) {  
 **when** (responseCode) {  
 401 -> {  
 **throw** Exception(JSONObject(AppUtils.inputStreamToString(connection.errorStream)).getJSONObject("data").getString("message"))  
 }  
 **else** -> {  
 **val** json = JSONObject(AppUtils.inputStreamToString(connection.errorStream))  
 **if** (json.has("data")) {  
 **throw** Exception(json.getJSONObject("data").getString("message"))  
 }  
 **throw** Exception("Unexpected response code **$**responseCode")  
 }  
 }  
 }  
 **val** json = JSONObject(AppUtils.inputStreamToString(connection.inputStream)).getJSONObject("data")  
 **return** Account(credentials.getString("login"), json.getString("jwt"))  
 }  
 **return null**  
}  
  
 **fun** removeAccount(credentials: JSONObject) {  
 **if** (Build.VERSION.SDK\_INT > 8) {  
 **val** policy = StrictMode.ThreadPolicy.Builder()  
 .permitAll().build()  
 StrictMode.setThreadPolicy(policy)  
 **val** url = URL("https://api.kiinse.me/account/remove")  
 **val** connection: HttpURLConnection = url.openConnection() **as** HttpURLConnection  
 connection.setRequestProperty("Content-Type", "application/json")  
 connection.setRequestProperty("No-timeout", "kitsune")  
 connection.requestMethod = "POST"  
 connection.connectTimeout = 10000  
 AppUtils.writeToStream(connection.outputStream, credentials)  
 **if** (connection.responseCode != HttpURLConnection.HTTP\_OK) {  
 **val** json: JSONObject?  
 **try** {  
 json = JSONObject(AppUtils.inputStreamToString(connection.errorStream)).getJSONObject("data")  
 } **catch** (e: Exception) {  
 **throw** Exception("Unexpected code **${**connection.responseCode**}**")  
 }  
 **if** (json != **null**) {  
 **throw** Exception(json.getString("message"))  
 }  
 **throw** Exception("Unexpected code **${**connection.responseCode**}**")  
 }  
 }  
 }  
}

Ниже приведен метод создания панели c информацией о станции:

**override fun** onViewCreated(view: View, savedInstanceState: Bundle?) {  
 **super**.onViewCreated(view, savedInstanceState)  
 **val** exitButton = view.findViewById<ImageButton>(R.id.back\_button)  
 exitButton.setOnClickListener **{**  
exitButton.startAnimation(AnimationUtils.loadAnimation(MainActivity.context, R.anim.alpha\_animator))  
 MainActivity.instance?.removeAllFragments()  
 **}**   
 **val** station = data  
 **if** (station != **null**) {  
 **val** context = view.context  
  
 view.findViewById<RelativeLayout>(R.id.station\_enabled).background = **when** (station.enabled) {  
 **true** -> ContextCompat.getDrawable(context, R.drawable.station\_enabled)  
 **else** -> ContextCompat.getDrawable(context, R.drawable.station\_disabled)  
 }  
  
 view.findViewById<TextView>(R.id.station\_name).text = station.name  
 **val** location = station.location  
 view.findViewById<TextView>(R.id.station\_address).text = "**${**location.city.name**}**, **${**location.address**}**"  
  
 **var** layout = view.findViewById<LinearLayout>(R.id.fuel\_list)  
 **var** drawable = ContextCompat.getDrawable(context, R.drawable.rounded\_all\_corners\_fuel)  
 **for** (fuel **in** station.fuels) {  
 AppUtils.createFuelLayout(context, fuel, layout, drawable, ContextCompat.getColor(context, R.color.gray))  
 }  
  
 layout = view.findViewById(R.id.suppliers\_list)  
 drawable = ContextCompat.getDrawable(context, R.drawable.rounded\_all\_corners\_suppliers)  
 **for** (supplier **in** station.suppliers) {  
 AppUtils.createButton(context, " **${**supplier.name**}** ", layout, drawable, AppUtils.getThemeTextColor(context))  
 }  
 }  
}

Ниже приведен класс отправки информации с сервера при запросе:

@Controller("/stations")  
open class StationsController {  
  
 **private val** stationFactory: StationFactory = StationFactory()  
  
 @Get  
 **fun** index(): HttpResponse<Response> {  
 **val** json = JSONObject()  
 json.put("publicKey", ApiRSA.get().getPublicKeyJson())  
 **return** ResponseFactory.create(HttpStatus.OK, json)  
 }  
  
 @Get("/all")  
 @Authentication(timeout = 1)  
 **open fun** all(request: HttpRequest<String?>): HttpResponse<Response> {  
 **return** ResponseFactory.create(HttpStatus.OK, stationFactory.getAllStations())  
 }  
  
 @Post("/byLocation")  
 @Authentication(timeout = 1)  
 **open fun** byLocation(request: HttpRequest<String?>, @Body location: StationLocation): HttpResponse<Response> {  
 **val** latitude = location.latitude  
 **val** longitude = location.longitude  
 **if** (latitude == **null** || longitude == **null**) **return** ResponseFactory.create(HttpStatus.NOT\_ACCEPTABLE)  
 **return** ResponseFactory.create(HttpStatus.OK, stationFactory.getStation(latitude, longitude))  
 }  
  
 @Get("/get")  
 @Authentication(timeout = 1)  
 **open fun** get(request: HttpRequest<String?>, id: Long?): HttpResponse<Response> {  
 **if** (id == **null**) **return** ResponseFactory.create(HttpStatus.NOT\_ACCEPTABLE)  
 **return** ResponseFactory.create(HttpStatus.OK, stationFactory.getStation(BigDecimal.valueOf(id)))  
 }  
}

Ниже приведен класс общения с базой данных:

**class** StationFactory {  
  
 **private val** context: DSLContext = PostgresDb.getContext()!!  
  
 **fun** getAllStations(): Set<Station> {  
 **val** set = HashSet<Station>()  
 **for** (station **in** context.select().from(Stations.STATIONS).fetch()) {  
 set.add(convertToStation(station))  
 }  
 **return** set  
 }  
  
 **fun** getAllFuels(): Set<Fuel> {  
 **val** set = HashSet<Fuel>()  
 **for** (fuel **in** context.select().from(Price.PRICE).fetch()) {  
 set.add(convertToFuel(fuel))  
 }  
 **return** set  
 }  
  
 **fun** getAllNews(): Set<NewsData> {  
 **val** set = HashSet<NewsData>()  
 **for** (news **in** context.select().from(News.NEWS).fetch()) {  
 set.add(convertToNews(news))  
 }  
 **return** set  
 }  
  
 **fun** getAllSuppliers(): Set<Supplier> {  
 **val** set = HashSet<Supplier>()  
 **for** (supplier **in** context.select().from(Suppliers.SUPPLIERS).fetch()) {  
 set.add(convertToSupplier(supplier))  
 }  
 **return** set  
 }  
  
 **fun** getAllLocations(): Set<Location> {  
 **val** set = HashSet<Location>()  
 **for** (location **in** context.select().from(Locations.LOCATIONS).fetch()) {  
 set.add(convertToLocation(location))  
 }  
 **return** set  
 }  
  
 **fun** getAllCities(): Set<City> {  
 **val** set = HashSet<City>()  
 **for** (city **in** context.select().from(Cities.CITIES).fetch()) {  
 set.add(convertToCity(city))  
 }  
 **return** set  
 }  
  
 **fun** getStation(id: BigDecimal): Station? {  
 **val** sqlStation = context.select()  
 .from(Stations.STATIONS)  
 .where(Stations.STATIONS.STATION\_ID.equal(id))  
 .fetch()  
 **if** (sqlStation.isEmpty()) **return null**  
 **return** convertToStation(sqlStation[0])  
 }  
  
 **fun** getStation(latitude: Double, longitude: Double): Station? {  
 **val** sqlLocation = context.select()  
 .from(Locations.LOCATIONS)  
 .where(Locations.LOCATIONS.LATITUDE.equal(BigDecimal.valueOf(latitude)))  
 .and(Locations.LOCATIONS.LONGITUDE.equal(BigDecimal.valueOf(longitude)))  
 .fetch()  
 **if** (sqlLocation.isEmpty()) **return null**  
 **return** getStation(BigDecimal.valueOf(convertToLocation(sqlLocation[0]).stationId.toLong()))  
 }  
  
 **fun** getLocation(id: BigDecimal): Location? {  
 **val** sqlStation = context.select()  
 .from(Locations.LOCATIONS)  
 .where(Locations.LOCATIONS.LOCATION\_ID.equal(id))  
 .fetch()  
 **if** (sqlStation.isEmpty()) **return null**  
 **return** convertToLocation(sqlStation[0])  
 }  
  
 **fun** getNews(id: BigDecimal): NewsData? {  
 **val** sqlNews = context.select()  
 .from(News.NEWS)  
 .where(News.NEWS.NEWS\_ID.equal(id))  
 .fetch()  
 **if** (sqlNews.isEmpty()) **return null**  
 **return** convertToNews(sqlNews[0])  
 }  
  
 **fun** getSupplier(id: BigDecimal): Supplier? {  
 **val** sqlStation = context.select()  
 .from(Suppliers.SUPPLIERS)  
 .where(Suppliers.SUPPLIERS.SUPPLIER\_ID.equal(id))  
 .fetch()  
 **if** (sqlStation.isEmpty()) **return null**  
 **return** convertToSupplier(sqlStation[0])  
 }  
  
 **fun** getFuel(id: BigDecimal): Fuel? {  
 **val** sqlStation = context.select()  
 .from(Price.PRICE)  
 .where(Price.PRICE.FUEL\_ID.equal(id))  
 .fetch()  
 **if** (sqlStation.isEmpty()) **return null**  
 **return** convertToFuel(sqlStation[0])  
 }  
  
 **fun** getCity(id: BigDecimal): City? {  
 **val** sqlCity = context.select()  
 .from(Cities.CITIES)  
 .where(Cities.CITIES.CITY\_ID.equal(id))  
 .fetch()  
 **if** (sqlCity.isEmpty()) **return null**  
 **return** convertToCity(sqlCity[0])  
 }  
  
 **private fun** getFuels(value: Array<Long>): Set<Fuel> {  
 **val** set = HashSet<Fuel>()  
 **for** (id **in** value) {  
 **val** query = context.select()  
 .from(Price.PRICE)  
 .where(Price.PRICE.FUEL\_ID.equal(BigDecimal.valueOf(id)))  
 .fetch()  
 **if** (!query.isEmpty()) set.add(convertToFuel(query[0]))  
 }  
 **return** set  
 }  
 **private fun** getSuppliers(value: Array<Long>): Set<Supplier> {  
 **val** set = HashSet<Supplier>()  
 **for** (id **in** value) {  
 **val** query = context.select()  
 .from(Suppliers.SUPPLIERS)  
 .where(Suppliers.SUPPLIERS.SUPPLIER\_ID.equal(BigDecimal.valueOf(id)))  
 .fetch()  
 **if** (!query.isEmpty()) set.add(convertToSupplier(query[0]))  
 }  
 **return** set  
 }  
  
 **private fun** convertToStation(data: Record): Station {  
 **return** Station(data.getValue(Stations.STATIONS.STATION\_ID).intValueExact(),  
 data.getValue(Stations.STATIONS.ENABLED),  
 data.getValue(Stations.STATIONS.NAME),  
 getFuels(data.getValue(Stations.STATIONS.FUELS)),  
 getLocation(data.getValue(Stations.STATIONS.LOCATION\_ID))!!,  
 getSuppliers(data.getValue(Stations.STATIONS.SUPPLIERS)))  
 }  
  
 **private fun** convertToFuel(data: Record): Fuel {  
 **return** Fuel(data.getValue(Price.PRICE.FUEL\_ID).intValueExact(),  
 data.getValue(Price.PRICE.NAME),  
 data.getValue(Price.PRICE.PRICE\_))  
 }  
  
 **private fun** convertToSupplier(data: Record): Supplier {  
 **return** Supplier(data.getValue(Suppliers.SUPPLIERS.SUPPLIER\_ID).intValueExact(),  
 data.getValue(Suppliers.SUPPLIERS.NAME),  
 data.getValue(Suppliers.SUPPLIERS.FUELS),  
 data.getValue(Suppliers.SUPPLIERS.STATIONS))  
 }  
  
 **private fun** convertToCity(data: Record): City {  
 **return** City(data.getValue(Cities.CITIES.CITY\_ID).intValueExact(),  
 data.getValue(Cities.CITIES.NAME),  
 data.getValue(Cities.CITIES.STATIONS))  
 }  
  
 **private fun** convertToNews(data: Record): NewsData {  
 **return** NewsData(data.getValue(News.NEWS.NEWS\_ID).intValueExact(),  
 data.getValue(News.NEWS.TITLE),  
 data.getValue(News.NEWS.TEXT),  
 data.getValue(News.NEWS.DATE))  
 }  
  
 **private fun** convertToLocation(data: Record): Location {  
 **return** Location(data.getValue(Locations.LOCATIONS.LOCATION\_ID).intValueExact(),  
 data.getValue(Locations.LOCATIONS.STATION\_ID).intValueExact(),  
 getCity(data.getValue(Locations.LOCATIONS.CITY))!!,  
 data.getValue(Locations.LOCATIONS.ADDRESS),  
 Coordinates(data.getValue(Locations.LOCATIONS.LATITUDE),  
 data.getValue(Locations.LOCATIONS.LONGITUDE)))  
 }  
}

Ниже приведен класс авторизации на сервере при запросе:

@Singleton  
@InterceptorBean(Authentication::**class**)  
class AuthInterceptor : MethodInterceptor<Any, Any> {  
  
 @Inject **private var** authService: AuthService? = **null**  
  
@Suppress("UNCHECKED\_CAST")  
 **override fun** intercept(context: MethodInvocationContext<Any, Any>?): Any? {  
 **try** {  
 **if** (context != **null**) {  
 **for** (value **in** context.parameterValueMap.values) {  
 **if** (value **is** HttpRequest<\*>) {  
 **val** annotation = context.getAnnotation(Authentication::**class**.java)  
 **if** (annotation != **null**) {  
 **val** request = value **as** HttpRequest<String?>  
 **if** (annotation.values["permissions"] == **null** && checkTimeOut(request, annotation, "anonymous", context.methodName, value.remoteAddress)) {  
 **return** context.proceed()  
 }  
 **val** account = authService?.login(  
 RequestUtils.getBearer(request)  
 ?: **throw** AuthException(HttpStatus.UNAUTHORIZED, "Need Authorization token!")  
 )  
 **if** (account != **null** && checkPermissions(annotation.values["permissions"] **as** Array<String>, account.permissions)  
 && checkTimeOut(request, annotation, account.userName, context.methodName, value.remoteAddress)) {  
 **return** context.proceed()  
 }  
 }  
 }  
 }  
 }  
 **return** ResponseFactory.create(HttpStatus.UNAUTHORIZED)  
 } **catch** (exception: Exception) {  
 **return when** (exception) {  
 **is** HttpStatusException -> ResponseFactory.create(exception)  
 **else** -> ResponseFactory.create(HttpStatus.INTERNAL\_SERVER\_ERROR, exception)  
 }  
 }  
 }  
  
 @Throws(ManyRequestsException::**class**)  
 **private fun** checkTimeOut(request: HttpRequest<String?>, annotation: AnnotationValue<Authentication>,  
 accountName: String, methodName: String, remoteAddress: InetSocketAddress): Boolean {  
 **val** timeoutHeader = RequestUtils.getHeader(request, "No-timeout")  
 **if** (timeoutHeader != **null** && timeoutHeader == "kitsune") **return true**  
 **val** timeout = annotation.values["timeout"].toString().toInt()  
 **if** (AuthTimeout.isTimeOuted("**$**methodName-**$**accountName-**${**remoteAddress.hostString**}**", timeout))  
 **throw** ManyRequestsException(HttpStatus.TOO\_MANY\_REQUESTS, "Too many requests! Please send a request no more than once every **$**timeout seconds!")  
 **return true**  
}  
  
 @Throws(AccountException::**class**)  
 **private fun** checkPermissions(permissions: Array<String>, accountPermissions: Set<AccountPermission>): Boolean {  
 **if** (accountPermissions.contains(AccountPermission.ADMIN)) {  
 **return true**  
}  
 **if** (accountPermissions.contains(AccountPermission.USER) && !permissions.contains(AccountPermission.ADMIN.toString())) {  
 **return true**  
}  
 **for** (permission **in** permissions) {  
 **if** (accountPermissions.contains(AccountPermission.valueOf(permission))) {  
 **return true**  
}  
 }  
 **throw** AccountException(HttpStatus.UNAUTHORIZED, "You are not authorized for this request! You must have one of **${**  
permissions.joinToString(prefix = "[", postfix = "]", separator = ", ")  
 **}** rights!")  
 }  
}

**Источники, использованные при разработке**

1. Документация Google [электронный ресурс] URL: <https://developer/android.com/docs> (дата обращения: 20.08.2022)
2. Документация JetBrains [электронный ресурс] URL: <https://kotlinlang.org/docs/home.html> (дата обращения: 22.08.2022)
3. Документация Yandex [электронный ресурс] URL: <https://yandex.ru/dev/maps/mapkit?from=mapsapi> (дата обращения: 26.08.2022)
4. Документация Micronaut [электронный ресурс] URL: <https://docs.micronauit.io/index.html> (дата обращения: 03.09.2022)