**Содержание**

[**1. Управление документом 2**](#_efomwelj346k)

[1.1 Назначение документа 2](#_q2x798mcn09a)

[1.2 История изменений 3](#_km7ta53cuk73)

[1.3 Ограничения и допущения 4](#_lkc853wai47h)

[1.4 Цели и задачи 4](#_d74fb5b81hdz)

[**2. Описание автоматизируемых бизнес-процессов 5**](#_998iu4yb21pq)

[2.1 User Stories 5](#_kacwo8k74paa)

[2.2 Бизнес-процессы 7](#_jnfedgtgv3vo)

[2.2.1 Подача заявки на обучение 7](#_anzgkjnrsoo0)

[2.2.2 Управление процессом обработки заявок 9](#_qe9lu4e3px5d)

[**3. Описание решения 11**](#_nbuez5jlrf)

[3.1 Общее описание 11](#_c9lkeh8e8fu5)

[3.2 Функциональные требования 13](#_p6w8twmhseew)

[3.2.1 Список функциональных требований 13](#_5wqdwefp8if2)

[3.2.2 Требования к графическому интерфейсу 14](#_dg7xw1dn3rz8)

[3.2.2.1. Общие требования к интерфейсу: 14](#_k4gszfnpiz5z)

[3.2.2.2.Описание взаимодействий и API 15](#_4o1daem474r0)

[3.2.2.3.Макеты графического интерфейса 24](#_q4qvtligox4v)

[3.2.3 Варианты использования (Use Cases) 27](#_z3km7p1f1eom)

[3.2.3.1. UС-1 Подача заявки на обучение 27](#_1r6p7phmigjv)

[3.2.3.2.UС-2 Подача предзаявки на обучение 31](#_7x8vuaqek74)

[3.2.3.3.UС-3 Управление заявками в админ-панели 34](#_epb5kllyq920)

[3.2.3.4.UС-4 Настройка периодов активности кнопок 35](#_qpvzc7tyztqs)

[3.2.3.5.UС-5 Анализ заявок 36](#_j9i4emvr5mhw)

[3.3. Нефункциональные требования 37](#_ligd9416992t)

[3.4 Архитектурные требования 39](#_reekrgldz3ou)

[3.4.1.Схема архитектуры 39](#_tunmg9tq1cnz)

[3.4.2.Схема базы данных 40](#_ndpl3p8kh9d7)

[3.5. Спецификации для модулей 44](#_qixyolxdbig3)

[3.5.1. Модуль Telegram-бота 44](#_wm1sk9cxh6jq)

[3.5.2. Модуль базы данных 48](#_t6ioi9tz51kb)

[3.5.3. Модуль административной панели 51](#_yzbxg8elycjv)

[3.5.4. Микросервис №1 (База данных) 55](#_54ok3qlj488d)

[3.5.5. Микросервис №2 (API) 58](#_t2dx7xscraq4)

# 1. Управление документом

## 1.1 Назначение документа

Документ является техническим заданием для реализации решения команды Алхимики кода на Хакатоне компании Неофлекс. Он описывает требования, которые необходимо реализовать для успешного выполнения проекта. Документ предназначен для команды проекта: аналитиков, тестировщиков, разработчиков и дата инженера.

Участники команды:

Крюков Роман (капитан)  
Богдановский Михей

Волков Дмитрий

Воробьева Кристина

Кабачек Никита  
Крылов Сергей

Литвинов Андрей  
Малашок Полина

Мамонтов Кирилл

Попов Илья

Федосова Мила

**Технологии разработки:**

Frontend

* React.JS
* TypeScript
* Bootstrap
* Vite
* SCSS

Backend

* Spring Boot
* Spring Data
* Spring Security
* JWT
* Liquibase
* Swagger
* Postgresql

## 1.2 История изменений

| N | Дата | Авторы | Описание изменений |
| --- | --- | --- | --- |
| 1 | 26.01.2025 | Малашок Полина, Крылов Сергей, Федосова Мила | Первоначальная версия документа |

## 1.3 Ограничения и допущения

- Кнопка «Подать заявку» доступна только в строго определённые даты, указанные организатором.

- Кнопка «Оставить предзаявку» становится активной, если набор закрыт. Одновременная активация двух кнопок запрещена.

- Логика работы с заявками предполагает, что каждый пользователь может подать только одну заявку на одно направление.

- Изменение шаблонов сообщений доступно только через административный интерфейс, но не через Telegram.

- Бот рассчитан на обработку до 5 000 заявок на курс; при росте нагрузки потребуется доработка инфраструктуры.

- Редактирование данных заявки пользователя после ввода невозможно.

- Время отклика на пользовательский запрос в боте не должно превышать 3 секунд.

- Администраторы имеют доступ только к заявкам, но не к другим данным, введённым пользователями, кроме указанных в заявке.

## 1.4 Цели и задачи

Цели:

- Создание Telegram-бота для работы с заявками на обучение на Хакатоне компании Неофлекс командой Алхимики кода.

- Автоматизация процесса подачи заявок для упрощения взаимодействия с заявкой для пользователей, снижение временных затрат и минимизация ошибок при обработке заявок.

Задачи:

- Собрать бизнес-требования, определить функциональные и нефукциональные требования, включая роли пользователей, доступные функции и технические ограничения

- Разработка работающего прототипа Telegram-бота для обработки заявок с учетом разных сценариев (активные и неактивные наборы)

- Интеграция с системой обучения (реализация сценариев выдачи ссылки на страницу с входными испытаниями)

- Проектирование и реализация базы данных для хранения заявок, пользователей, курсов и логов

- Создание административной панели (отображение в панели списка заявок с возможностью фильтрации и сортировки)

- Разработка двух микросервисов (база данных и API) и настройка взаимодействия между ними

- Проектирование API контракта между Frontend и Backend

- Подключение аналитики и логирования (сбор данных о заявках, настройка логирования операций, интеграция с инструментами мониторинга)

- Разработка тестов для проверки функциональности (все сценарии, корректности работы кнопок, взаимодействия Frontend и Backend)

- Создание отчета о результатах тестирования в месте с путями их исправления.

# 2. Описание автоматизируемых бизнес-процессов

**Бизнес-требование:** Необходимо обеспечить инструмент для улучшения пользовательского опыта и оптимизации процесса управления наборами студентов, что позволит Компании-организатору повысить эффективность обработки заявок и лучше прогнозировать потоки обучения.

## 2.1 User Stories

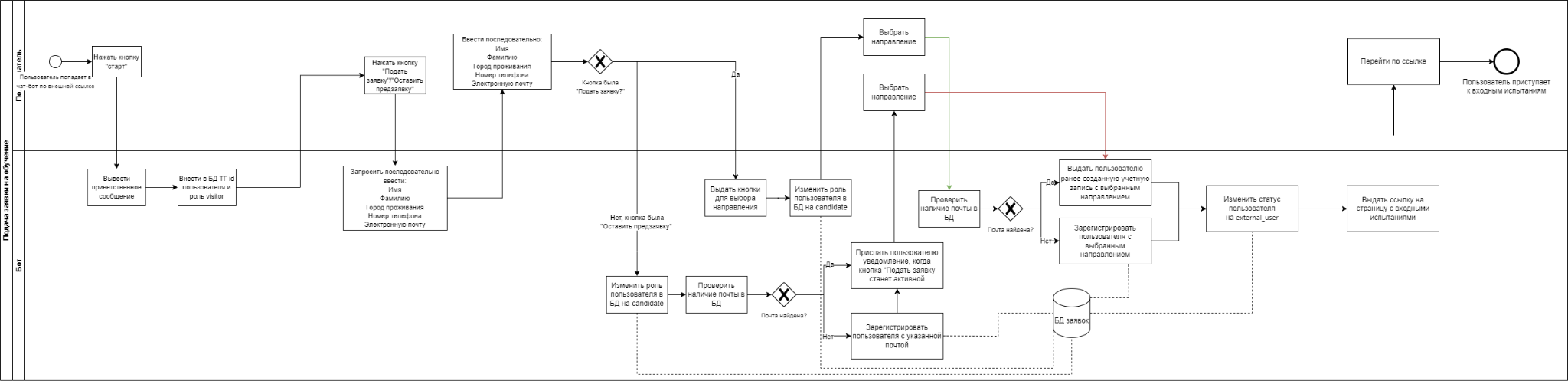
| Us\_id | User story | Роль |
| --- | --- | --- |
| 1 | Как пользователь, я хочу подать заявку через Telegram-бот, чтобы участвовать в обучении по выбранному направлению. | Пользователь |
| 2 | Как пользователь, я хочу оставить предзаявку, чтобы быть уведомленным о начале набора. | Пользователь |
| 3 | Как пользователь, я хочу выбирать направления обучения, чтобы записаться на интересующий меня курс (например, Java, DevOps, Frontend и др. ). | Пользователь |
| 4 | Как пользователь, я хочу получать уведомления в Telegram об открытии набора в учебный центр, чтобы оперативно зарегистрироваться. | Пользователь |
| 5 | Как пользователь, я хочу получить ссылку на страницу с входными испытаниями, чтобы продолжить процесс регистрации. | Пользователь |
| 6 | Как пользователь, я хочу получать подтверждение успешной подачи заявки в виде сообщения в Telegram, чтобы быть уверенным, что моя заявка зарегистрирована. | Пользователь |
| 7 | Как пользователь, я хочу видеть список доступных направлений обучения, чтобы выбрать наиболее подходящий курс. | Пользователь |
| 8 | Как администратор, я хочу видеть список всех заявок в админ-панели, чтобы управлять процессом набора. | Администратор |
| 9 | Как администратор, я хочу фильтровать и сортировать заявки, чтобы быстро находить нужных кандидатов. | Администратор |
| 10 | Как администратор, я хочу просматривать подробную информацию о каждой заявке, чтобы уточнять данные кандидатов. | Администратор |
| 11 | Как администратор, я хочу изменять статус заявки («В ожидании», «На рассмотрении», «Принята», «Отклонена»), чтобы управлять процессом набора. | Администратор |
| 12 | Как администратор, я хочу настраивать даты активности кнопок «Подать заявку» и «Оставить предзаявку», чтобы контролировать процесс подачи заявок. | Администратор |
| 13 | Как администратор, я хочу видеть витрину с количеством заявок по направлениям обучения (Big Data, бэкенд, тестирование и т.д.), чтобы анализировать популярность курсов. | Администратор |
| 14 | Как администратор, я хочу анализировать данные о заявках (количество предзаявок, количество зарегистрированных заявок), чтобы понимать конверсии и улучшать процесс набора. | Администратор |
| 15 | Как администратор, я хочу получать аналитические данные ежедневно, чтобы следить за процессом набора и улучшать его. | Администратор |
| 16 | Как администратор, я хочу получать отчеты о количестве пользователей, датах и выбранных направлениях обучения в формате Excel, чтобы анализировать данные набора и улучшать его процесс. | Администратор |
| 17 | Как администратор, я хочу видеть распределение пользователей по городам проживания, чтобы понимать географическое распределение кандидатов. | Администратор |

## 2.2 Бизнес-процессы

### 2.2.1 Подача заявки на обучение

**Цель:** Представить процесс подачи заявки на обучение пользователем с помощью бота с использованием нотации BPMN

Схема в развернутом виде доступна файлом [по ссылке](https://drive.google.com/file/d/1d30-feP6pInKcMGsi3F2Gpm9CU6M4UmI/view)



**Участники процесса:**

* Пользователь
* Бот

**Описание процесса:**

Пользователь:

1. Начинает процесс с попадания в чат-бот по внешней ссылке
2. Нажимает на кнопку “Старт”

Бот:

1. Выводит приветственное сообщение пользователю
2. Вносит в Базу Данных ТГ ID пользователя и присваивает ему роль visitor

Пользователь:

1. Нажимает кнопку Подать заявку/оставить предзаявку

Бот:

1. Запрашивает последовательный ввод данных: “Имя”, “Фамилия”, “Город проживания”, “Номер телефона”, “Электронная почта”

Пользователь:

1. Последовательно вводит данные: “Имя”, “Фамилия”, “Город проживания”, “Номер телефона”, “Электронная почта”

Если в шаге 5 Пользователь нажимает кнопку “Подать заявку”, Бот выдает пользователю кнопки для выбора направления

8.1) Бот меняет роль пользователя в БД на candidate

9.1) Пользователь выбирает направление обучения.

10.1) Бот проверяет наличие почты в Базе данных, если

11.1) Если почта обнаружена, Бот выдает пользователю ранее созданную учетную запись с выбранным направлением, если почта не обнаружена, Бот регистрирует пользователя с выбранным направлением

12.1) Бот меняет статус пользователя на external\_user

13.1) Бот выдает ссылку на страницу с входными испытаниями

14.1) Пользователь переходит по ссылке

Если в шаге 5 Пользователь нажимает кнопку “Оставить предзаявку”, Бот изменяет роль пользователя БД на candidate

8.2) Бот проверяет наличие почты в Базе данных

9.2) Если почта была найдена в базе данных, Бот присылает пользователю уведомление, когда кнопка “Подать заявку” станет активной, если почта не найдена в базе данных, Бот регистрирует пользователя с указанной почтой и после Бот присылает пользователю уведомление, когда кнопка “Подать заявку” станет активной

10.2) Пользователь выбирает направление

11.2) Бот выдает пользователю ранее созданную учетную запись с выбранным направлением

12.2) Бот меняет статус пользователя меняется на external\_user

13.2) Бот выдает ссылку на страницу с входными испытаниями

14.2) Пользователь переходит по ссылке

Итог:

Процесс заканчивается, когда пользователь приступает к входным испытаниям

### 2.2.2 Управление процессом обработки заявок

**Цель:** Представить процесс управления обработкой заявок администратором с использованием нотации BPMN

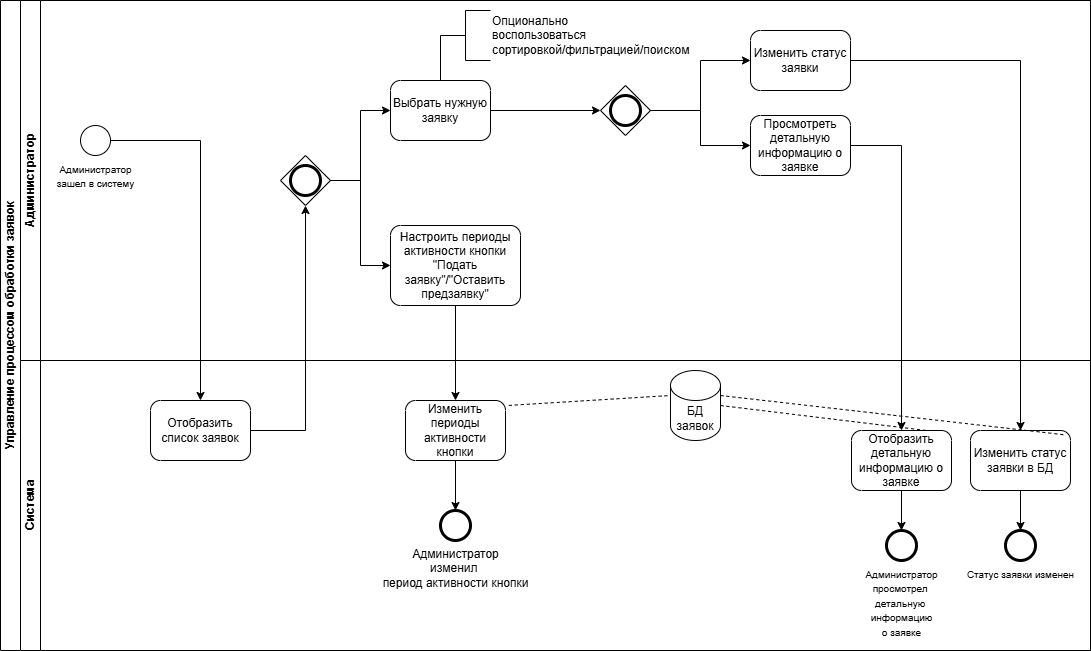


Схема в развернутом виде доступна [по ссылке](https://drive.google.com/file/d/1jQlrB0eBMq8skU4kl1j05tUpRYnmryCF/view)

**Участники процесса:**

* Администратор
* Система

**Описание процесса:**

Администратор:

1. Осуществляет вход в систему

Система:

1. Отображает список заявок

Администратор:

1. Если Администратор выбирает настройку периодов активности кнопки “Подать заявку”/”Оставить предзаявку”, Система меняет периоды активности кнопки, и после этого происходит выход из бизнес-процесса

4) Если Администратор выбирает нужную заявку, то он выбирает одно из следующих действий:

5) Если Администратор меняет статус заявки, то Система меняет статус заявки в Базе Данных и происходит выход из бизнес-процесса

6) Если Администратор выбирает просмотр детальной информации по заявке, система отображает детальную информацию по заявке и происходит выход из бизнес-процесса.

Итог:

Процесс заканчивается, когда происходит одно из следующих действий: Администратор изменил в Системе период активности кнопки, Администратор просмотрел в Системе детальную информацию по заявке или Администратор изменил в Системе статус заявки

# 3. Описание решения

## 3.1 Общее описание

Роли:

* пользователь
* администратор

Решение позволяет пользователям быстро и удобно регистрироваться на интересующие направления, а организаторам эффективно управлять заявками и анализировать данные для оптимизации набора.

Пользователь может выбрать направление обучения, заполнить личные данные (имя, город, номер телефона, email) и подать заявку через Telegram-бот.

После успешной подачи заявки пользователь получает подтверждение и ссылку на страницу с входными испытаниями.

Если набор еще не открыт, пользователь может оставить предзаявку. После активации набора он получит уведомление и возможность завершить процесс регистрации.

Пользователь может просмотреть список доступных курсов и выбрать подходящий.

Администратор имеет доступ к списку всех заявок через админ-панель.

Доступны инструменты для фильтрации и сортировки заявок по статусу, направлениям, и другим параметрам.

Администратор может управлять периодами активности кнопок «Подать заявку» и «Оставить предзаявку», что позволяет гибко планировать периоды набора.

Администратор имеет доступ к аналитике данных о заявках.

Система позволяет отслеживать статистику активных пользователей и их действия.

## 3.2 Функциональные требования

### 3.2.1 Список функциональных требований

| FR-id | Функциональное требование |
| --- | --- |
| 1 | Бот должен предоставлять пользователю возможность заполнить свои данные: (имя и фамилия, город проживания, номер телефона, электронная почта) |
| 2 | Данные должны быть проверены на корректность (формат email и номера телефона) |
| 3 | Бот должен позволять пользователю выбрать направление обучения из списка доступных курсов |
| 4 | Бот должен позволить пользователю нажать кнопку “Подать заявку” или “Оставить предзаявку”, в зависимости от того, какая кнопка активна в момент обращения пользователя |
| 5 | Бот должен зарегистрировать заявку в базе данных с указанием всех введённых данных, направления и текущей даты подачи |
| 6 | Бот должен отправлять пользователю уведомление в Telegram при: открытии набора (для предзаявок), успешной подаче заявки. |
| 7 | После успешной подачи заявки бот должен отправлять пользователю ссылку на страницу с входными испытаниями. |
| 8 | Административная панель должна отображать список всех поданных заявок с возможностью:фильтрации по направлениям, статусам и датам) и сортировки по полям (дата подачи, статус заявки). |
| 9 | Административная панель должна позволять Администратору открыть подробную информацию о каждой заявке |
| 10 | Административная панель должна позволять Администратору менять статус заявки на: «В ожидании», «На рассмотрении», «Принята», «Отклонена». |
| 11 | Административная панель должна позволять Администратору задавать периоды активности для кнопок «Подать заявку» и «Оставить предзаявку». |
| 12 | База данных должна содержать список доступных направлений обучения. |
| 13 | Каждая заявка в Базе Данных должна включать:личные данные пользователя (имя, город, email, телефон), выбранное направление, статус заявки, тип заявки («Заявка» или «Предзаявка»), дата подачи заявки. |
| 14 | Бот должен предоставлять возможность изменения шаблонов базовых сообщений (таких как приветственные сообщения, уведомления об отказе или успешной регистрации) без необходимости изменения исходного кода |
| 15 | Бот должен выполнять проверку наличия электронной почты пользователя в базе данных при подаче заявки или предзаявки |

### 3.2.2 Требования к графическому интерфейсу

#### 3.2.2.1. Общие требования к интерфейсу:

* Интерфейс должен быть максимально простым и понятным для пользователей любого уровня технической подготовки.
* Язык интерфейса - все сообщения и отображаемые элементы должны иметь надписи на русском языке
* Интерфейс административной панели должен быть удобен для работы на различных устройствах с любыми разрешениями экранов.
* Интерфейс должен быть минималистичным, без излишних элементов, чтобы администратор мог легко ориентироваться.
* Для обозначения статусов (например, принят, отказано, в ожидании) должны использоваться разные цвета.
* Должна быть возможность фильтрации данных по нескольким параметрам (направление, город, статус).
* Кнопки "Изменить", выпадающие списки, вкладки и графики должны быть интерактивными и отзывчивыми.
* Интерфейс должен быть легким и загружаться быстро.
* Панель должна эффективно обрабатывать большое количество заявок.

#### 3.2.2.2.Описание взаимодействий и API

* **Экран 1 - Вкладка Заявки**

Вкладка "Заявки" предоставляет пользователю возможность:

- Просматривать список заявок.

- Оценивать текущий статус заявок (принято, в ожидании, отказано) с помощью визуальных подсказок (цветовая кодировка).

- Фильтровать данные по заданным критериям.

- Получать информацию по каждой заявке в виде карточек.

**Элементы интерфейса и действия:**

1. Фильтры

1.Пользователь кликает на выпадающий список.

2.Выбирает необходимый параметр фильтрации (например, "Направление").

3.Список заявок обновляется и отображает только те заявки, которые соответствуют выбранному критерию.

4.После применения фильтра цветовое распределение карточек изменяется в зависимости от статусов отфильтрованных заявок.

1. Карточки заявок

1.Пользователь просматривает карточки, быстро оценивая статусы заявок благодаря цветовой кодировке.

2.Может кликнуть на карточку для более детального просмотра.

3.Использует визуальные маркеры (цвета) для быстрого анализа текущего состояния заявок.

1. Навигация

1.Пользователь может кликнуть на любую из вкладок для перехода в соответствующий раздел панели.

2.Текущая активная вкладка выделяется визуально.

**API-методы, которые используются на вкладке Заявки**

1.GET /statements

Описание: Получение списка заявок с возможностью фильтрации.

Параметры запроса:

statements\_status (принята, отказано, в ожидании, на рассмотрении) — фильтрация по статусу.

course — фильтрация по направлению.

2.GET /statement/{id}

Описание: Просмотр детальной информации о заявке с данными пользователя и направлением.

Параметры запроса:

id — уникальный идентификатор заявки.

3.PUT /data-storage/updateStatementStatus

Описание: Обновление статуса заявки и добавление его в историю через админ-панель.

4.GET /data-storage/user/{telegramId}

Описание: Проверка существования пользователя.

Параметры:

telegram\_Id — ID пользователя в Telegram

5.GET /data-storage/courses

Описание: Получение списка курсов.

6. POST /data-storage/insertCourse/{courseName}

Описание: Добавление нового направления обучения.

Параметры:

course — описание курса

7.POST /data-storage/user

Описание: Сохранение пользователя.

9. GET /data-storage/users

Описание: Получение списка всех пользователей.

9. GET /data-storage/users/without\_course

Описание: Получение списка всех пользователей без курса.

10. GET /data-storage/user\_email/{email}

Описание: Проверка существования почты.

Параметры:

email — электронная почта пользователя

11. PUT /admin/statement\_status

Описание: Обновление статуса заявки и добавление его в историю через админ-панель.

* **Экран 2 - Принятые заявки**

Вкладка "Принятые Заявки" предназначена для работы с заявками, статус которых установлен как "Принята". Пользователь может:

- Просматривать детальную информацию о принятых заявках.

- Редактировать данные заявок.

- Фильтровать заявки по различным параметрам.

- Искать конкретные заявки.

**Элементы интерфейса и действия:**

1. Фильтры

1.Пользователь кликает на выпадающий список фильтров.

2.Выбирает фильтр, например "Направление" или "Город".

3.После выбора фильтра система обновляет список заявок, показывая только те, которые соответствуют выбранным критериям.

1. Карточки заявок

1.Пользователь изучает основные сведения по каждой заявке.

2.Пользователь нажимает кнопку "Изменить".

3.Это действие открывает модальное окно, где администратор может изменить данные заявки, включая статус, направление или контактную информацию.

4.Пользователь может быстро получить представление обо всех принятых заявках, поскольку все карточки имеют одинаковый цвет.

1. Поиск

1.Пользователь вводит ключевые слова (например, имя, фамилию или направление).

2.Система обновляет отображаемый список, показывая только те заявки, которые соответствуют запросу.

1. Навигация

1.Пользователь может кликнуть на любую из вкладок для перехода в соответствующий раздел панели.

2.Текущая активная вкладка выделяется визуально.

**API-методы, которые используются на вкладке Принятые Заявки**

1.GET /accepted-statements

Описание: Получение списка принятых заявок с возможностью фильтрации.

Параметры запроса:

course — фильтрация по направлению.

2.GET /accepted-statements/{id}

Описание: Получение подробной информации по конкретной заявке.

Параметры пути:

id — уникальный идентификатор заявки.

3.PUT /accepted-statements/{id}

Описание: Изменение данных заявки.

Параметры пути:

id — уникальный идентификатор заявки.

4.PUT /admin/statement\_status

Описание: Обновление статуса заявки и добавление его в историю через админ-панель.

5.POST /admin/set/period

Описание: Задание периода подачи заявок (даты начала и конца периода).

* **Экран 3 - Система фильтрации**

На экране более развернуто представлены реализованные для административной панели фильтры. Взаимодействие с пользователем для этого экрана аналогично взаимодействию пользователя с вкладкой Заявки (Экран 1).

**API-методы на этого экрана аналогичны методам, использующимся на Экране 1**

Дополнительно:

1.GET /data-storage/getStatementsОписание: Получение всех заявок пользователя.

* **Экран 4 - Вкладка Предзаявки**

Вкладка "Предзаявки" предназначена для работы с предварительными заявками, которые еще не перешли в заявки и отображает тех, кто хотел оставить заявку, но набор в данный период времени не был открыт. Пользователь может:

- Просматривать детальную информацию о предзаявках.

- Редактировать данные заявок.

- Фильтровать заявки по различным параметрам.

**Элементы интерфейса и действия:**

1. Фильтры

1.Пользователь кликает на выпадающий список фильтров.

2.Выбирает критерий фильтрации (например, по городу или другому параметру).

3.После выбора фильтра система обновляет отображение списка предзаявок, оставляя только те, которые соответствуют заданному фильтру.

1. Карточки предзаявок

1.Пользователь может ознакомиться с основной информацией, представленной в карточке каждой предзаявки.

2.Пользователь может кликнуть на карточку для перехода к детальной информации или изменению данных.

1. Навигация

1.Пользователь может кликнуть на любую из вкладок для перехода в соответствующий раздел панели.

2.Текущая активная вкладка выделяется визуально.

API-методы, которые используются на вкладке Предзаявки

1.GET /prestatements

Описание: Получение списка предзаявок с возможностью применения фильтров.

Параметры запроса:

city — фильтрация по городу.

direction — фильтрация по направлению.

2.GET /prestatements/{id}

Описание: Получение подробной информации по конкретной предзаявке.

Параметры пути:

id — уникальный идентификатор предзаявки.

3.PUT /prestatements/{id}

Описание: Изменение данных предзаявки.

Параметры пути:

id — уникальный идентификатор предзаявки.

4.GET /data-storage/user\_email/{email}

Описание: Проверка существования почты.

Параметры:

email — Email пользователя.

5.GET /data-storage/user/{telegramId}

Описание: Проверка существования пользователя.

Параметры:

telegram\_id — ID пользователя в Telegram.

* **Экран 5 - Аналитика**

Вкладка "Аналитика" предназначена для анализа и визуализации данных о заявках, поступивших через Telegram-бот. Пользователь может:

- Просматривать общую статистику по заявкам.

- Формировать отчеты за определенные периоды.

- Анализировать распределение заявок по направлениям с помощью диаграммы.

**Элементы интерфейса и действия:**

1. Фильтры

1.Пользователь выбирает фильтр в выпадающем списке ("Нет фильтров").

2.Указывает диапазон дат, заполняя поля "Начало периода" и "Конец периода".

Нажимает кнопку "Сформировать" для обновления аналитических данных.

1. Блоки статистики

1.Пользователь просматривает данные, представленные в числовом формате, для быстрого понимания общего состояния заявок.

1. Круговая диаграмма

1.Пользователь анализирует визуальное распределение данных для оценки популярности направлений.

1. Кнопка "Выгрузить отчет"

1.После анализа данных пользователь нажимает кнопку "Выгрузить отчет".

2.Система предлагает сохранить отчет в формате Excel на устройство.

1. Навигация

1.Пользователь может кликнуть на любую из вкладок для перехода в соответствующий раздел панели.

2.Текущая активная вкладка выделяется визуально.

**API-методы, которые используются на вкладке Аналитика**

1.GET /analytics

Описание: Получение аналитических данных для отображения статистики и диаграмм.

Параметры запроса:

filter — выбранный фильтр (например, направление, город).

start\_date — начальная дата периода аналитики.

end\_date — конечная дата периода аналитики.

2.GET /analytics/directions

Описание: Получение данных для построения круговой диаграммы, отражающей распределение заявок по направлениям.

Параметры запроса:

start\_date — начальная дата периода аналитики.

end\_date — конечная дата периода аналитики.

3.POST /analytics/report

Описание: Выгрузка аналитических данных в файл Excel.

4.POST /user-actions

Описание: Сохранение действий пользователя, таких как применение фильтров или выгрузка отчета.

#### 3.2.2.3.Макеты графического интерфейса

Ниже представлены макеты графического интерфейса:

* Вкладка Заявки
* Вкладка Принятые заявки
* Система фильтрации
* Вкладка Предзаявки
* Вкладка Аналитика

|  |  |
| --- | --- |
| Вкладка Заявки | Вкладка Принятые заявки |
|  |  |
| Система Фильтрации | Вкладка Предзаявки |
|  |  |
| Вкладка Аналитика |  |

Макеты интерфейса доступны отдельным файлом [по ссылке](https://drive.google.com/file/d/19Ycs6ZvLxqJuCbnJyzZgGP8_E-PIAy-o/view).

### 3.2.3 Варианты использования (Use Cases)

#### 3.2.3.1. UС-1 Подача заявки на обучение

| № | Атрибут | Описание |
| --- | --- | --- |
| 1 | Участники | Пользователь |
| 2 | Предусловие | Пользователь попадает в чат-бот по внешней ссылке и получает роль visitor |
| 3 | Триггер | Пользователь нажал на «Подать заявку» |
| 4 | Основной сценарий | 1. Пользователь выбирает направление  2. Роль пользователя меняется на candidate  3. Пользователь заполняет поля формы:   * Имя.   Формат ввода: должно содержать только буквы и может включать символы дефиса или пробелы, от 2 до 30 символов.   * Фамилия.   Формат ввода: должно содержать только буквы и может включать символы дефиса или пробелы, от 2 до 30 символов.   * Город проживания:   Формат ввода: только название города, без уточнений, от 2 до 35 символов. Пример: "Пенза".   * Номер телефона.   Формат ввода: не менее 10 цифр, не более 15 символов. Пример: “+79123456789”   * Электронная почта.   Формат ввода: стандартный формат электронной почты (наличие символа @, корректный домен), до 64 символов  4. Происходит проверка наличия почты в БД  5. Если почта не найдена, то:  Телеграмм-бот регистрирует пользователя в системе с выбранным направлением  6. Роль пользователя меняется на external\_user  7. Пользователь получает ссылку на страницу с входными испытаниями |
| 5 | Альтернативный сценарий | 4. Если почта найдена, то:  Чат-бот выдаёт пользователю ранее созданную учётную запись с выбранным направлением  5. Роль пользователя меняется на external\_user  6.Пользователь получает ссылку на страницу с входными испытаниями |
| 6 | Исключительный сценарий | 1. Пользователь вводит некорректные данные:  Для имени: пустое поле, наличие символов, которые не являются буквами (например, цифры).  Для фамилии: пустое поле, наличие символов, которые не являются буквами (например, цифры).  Для города проживания: пустое поле или наличие некорректных символов.  Для номера телефона: несоответствие формату, наличие букв, неправильное количество цифр.  Для электронной почты: отсутствие символа @, некорректный домен.  2. Бот отправляет сообщение с инструкциями по корректному вводу данных.  3. Пользователь повторно вводит данные.  4. При повторной ошибке система завершает процесс подачи заявки. |
| 7 | Постусловие | Пользователь успешно переходит на страницу с входными испытаниями и начинает проходить испытания |

**Исключительный сценарий для UC-1 (Подача заявки на обучение)**

**1. Имя:**

* Валидация: Поле не должно быть пустым, должно содержать только буквы и может включать символы дефиса или пробелы, от 2 до 30 символов.
* Ошибка: Если имя содержит цифры, специальные символы (кроме дефиса) или поле пустое, бот отправляет сообщение:
* "Пожалуйста, укажите имя кириллицей без пробелов."

**2. Фамилия:**

* Валидация: Поле не должно быть пустым, должно содержать только буквы и может включать символы дефиса или пробелы, от 2 до 30 символов.
* Ошибка: Если фамилия содержит цифры, специальные символы (кроме дефиса) или поле пустое, бот отправляет сообщение:
* "Пожалуйста, укажите фамилию кириллицей без пробелов."

**3. Город проживания:**

* Валидация: Поле не должно быть пустым, должно содержать только буквы и может включать символы дефиса или пробелы, от 2 до 35 смволов.
* Ошибка: Если введено некорректное значение (например, цифры или специальные символы), бот отправляет сообщение:
* "Укажите корректный город проживания (только буквы, допустимы пробелы и дефис)."

**4. Номер телефона:**

* Должен быть в формате, соответствующем локальным стандартам страны (например, +79ХХХХХХХХХ, 89ХХХХХХХХХ, 9ХХХХХХХХХ), от 10 до 15 символов, поле не должно быть пустым, должно содержать только цифры и может включать символы плюса, дефиса или пробелы.
* Ошибка: Если номер телефона содержит буквы, некорректное количество цифр или иной формат, бот отправляет сообщение:
* "Пожалуйста, укажите номер телефона в формате например: +7ХХХХХХХХХХ"

**5. Электронная почта:**

* Валидация: Должна соответствовать формату электронной почты (наличие символа @, корректный домен), до 64 символов.
* Ошибка: Если введённая строка не соответствует формату, бот отправляет сообщение:
* "Пожалуйста, укажите существующую электронную почту, например: example@mail.com."

**6. Общий процесс обработки ошибок:**

* Если одно или несколько полей содержат ошибки, бот запрашивает повторный ввод только для неверно заполненных полей.
* После успешного ввода всех данных бот продолжает обработку заявки.

#### 3.2.3.2.UС-2 Подача предзаявки на обучение

| № | Атрибут | Описание |
| --- | --- | --- |
| 1 | Участники | Пользователь |
| 2 | Предусловие | Пользователь попадает в чат-бот по внешней ссылке и получает роль visitor |
| 3 | Триггер | Пользователь нажал на «Оставить предзаявку»: |
| 4 | Основной сценарий | 1. Роль пользователя меняется на candidate  Пользователь заполняет поля формы:   * Имя.   Формат ввода: должно содержать только буквы и может включать символы дефиса или пробелы, от 2 до 30 символов.   * Фамилия.   Формат ввода: должно содержать только буквы и может включать символы дефиса или пробелы, от 2 до 30 символов.   * Город проживания:   Формат ввода: только название города, без уточнений, от 2 до 35 символов. Пример: "Пенза".   * Номер телефона.   Формат ввода: не менее 10 цифр, не более 15 символов. Пример: “+79123456789”   * Электронная почта.   Формат ввода: стандартный формат электронной почты (наличие символа @, корректный домен), до 64 символов.  2. Происходит проверка наличия почты в БД  3. Если почта не найдена, то:  Телеграмм-бот регистрирует пользователя в системе с указанной почтой  4. После регистрации пользователю приходит уведомление, когда кнопка «Подать заявку» станет активной  5. Как только кнопка «Подать заявку» станет активной, пользователю приходит уведомление через телеграмм-бот об открытии нового набора  6. Пользователь выбирает направление  7. Роль пользователя меняется на external\_user  8. Пользователь получает ссылку на страницу с входными испытаниями |
| 5 | Альтернативный сценарий | 3. Если почта найдена, то:  Бот присылает пользователю уведомление, когда кнопка «Подать заявку» станет активной  4. Как только кнопка «Подать заявку» станет активной, пользователю приходит уведомление через телеграмм-бот об открытии нового набора  5. Пользователь выбирает направление  6. Чат-бот выдаёт пользователю ранее созданную учётную запись с выбранным направлением  7. Роль пользователя меняется на external\_user  8. Пользователь получает ссылку на страницу с входными испытаниями |
| 6 | Искючительный сценарий | 1. Пользователь вводит некорректные данные:  Для имени: пустое поле, наличие символов, которые не являются буквами (например, цифры).  Для фамилии: пустое поле, наличие символов, которые не являются буквами (например, цифры).  Для города проживания: пустое поле или наличие некорректных символов.  Для номера телефона: несоответствие формату, наличие букв, неправильное количество цифр.  Для электронной почты: отсутствие символа @, некорректный домен.  2. Бот отправляет сообщение с инструкциями по корректному вводу данных.  3. Пользователь повторно вводит данные.  4. При повторной ошибке система завершает процесс подачи заявки |
| 7 | Постусловие | Пользователь успешно переходит на страницу с входными испытаниями и начинает проходить испытания |

**Исключительный сценарий для UC-1 (Подача предзаявки на обучение)**

**1. Имя:**

* Валидация: Поле не должно быть пустым, должно содержать только буквы и может включать символы дефиса или пробелы, от 2 до 30 символов.
* Ошибка: Если имя содержит цифры, специальные символы (кроме дефиса) или поле пустое, бот отправляет сообщение:
* "Пожалуйста, укажите имя кириллицей без пробелов."

**2. Фамилия:**

* Валидация: Поле не должно быть пустым, должно содержать только буквы и может включать символы дефиса или пробелы, от 2 до 30 символов.
* Ошибка: Если фамилия содержит цифры, специальные символы (кроме дефиса) или поле пустое, бот отправляет сообщение:
* "Пожалуйста, укажите фамилию кириллицей без пробелов."

**3. Город проживания:**

* Валидация: Поле не должно быть пустым, должно содержать только буквы и может включать символы дефиса или пробелы, от 2 до 35 смволов.
* Ошибка: Если введено некорректное значение (например, цифры или специальные символы), бот отправляет сообщение:
* "Укажите корректный город проживания (только буквы, допустимы пробелы и дефис)."

**4. Номер телефона:**

* Должен быть в формате, соответствующем локальным стандартам страны (например, +79ХХХХХХХХХ, 89ХХХХХХХХХ, 9ХХХХХХХХХ), от 10 до 15 символов, поле не должно быть пустым, должно содержать только цифры и может включать символы плюса, дефиса или пробелы.
* Ошибка: Если номер телефона содержит буквы, некорректное количество цифр или иной формат, бот отправляет сообщение:
* "Пожалуйста, укажите номер телефона в формате например: +7ХХХХХХХХХХ"

**5. Электронная почта:**

* Валидация: Должна соответствовать формату электронной почты (наличие символа @, корректный домен), до 64 символов.
* Ошибка: Если введённая строка не соответствует формату, бот отправляет сообщение:
* "Пожалуйста, укажите существующую электронную почту, например: example@mail.com."

**6. Общий процесс обработки ошибок:**

* Если пользователь ввёл некорректные данные, бот уведомляет об ошибке и запрашивает уточнение.
* После ввода всех данных бот завершает процесс подачи предзаявки.

#### 3.2.3.3.UС-3 Управление заявками в админ-панели

| **№** | **Атрибут** | **Описание** |
| --- | --- | --- |
| 1 | Участники | Администратор  Система |
| 2 | Предусловие | Администратор авторизован в системе.  Список заявок доступен. |
| 3 | Триггер | Администратор заходит в админ-панель. |
| 4 | Основной сценарий: | 1. Администратор входит в админ-панель с учётной записью.  2. Система отображает список заявок с фильтрацией и сортировкой.  3. Администратор находит нужную заявку и открывает её детали.  4. Администратор изменяет статус заявки:   * "В ожидании" * "На рассмотрении". * "Принята". * "Отклонена".   5. Cистема обновляет статус заявки и отправляет уведомление пользователю. |
| 5 | Альтернативный сценарий | Администратор использует фильтры (по направлениям, статусам, датам подачи), чтобы быстро найти нужную заявку. |
| 6 | Исключительный сценарий | Ошибка подключения к базе данных.  Система выводит сообщение об ошибке и предлагает повторить операцию позже. |
| **7** | Постусловие | Постусловие: Заявка обновлена, пользователь получает уведомление о новом статусе. |

#### 3.2.3.4.UС-4 Настройка периодов активности кнопок

| № | Атрибут | Описание |
| --- | --- | --- |
| 1 | Участники | Администратор Система |
| 2 | Предусловие | Администратор авторизован в админ-панели.  Доступен раздел настройки активности кнопок. |
| 3 | Триггер | Администратор открывает раздел настройки активности кнопок. |
| 4 | Основной сценарий | 1. Администратор заходит в админ-панель.  2. Открывает раздел настроек активности кнопок.  3. Выбирает дату начала и окончания активности для кнопки "Подать заявку".  4. Устанавливает параметры активности для кнопки "Оставить предзаявку".  5. Система сохраняет изменения.  6. Бот обновляет статус кнопок в режиме реального времени. |
| 5 | Альтернативный сценарий | Администратор задаёт только дату начала активности, оставляя окончание открытым. |
| 6 | Исключительный сценарий | Введены некорректные даты (например, пересечение периодов активности кнопок).  Система выводит сообщение об ошибке и не позволяет сохранить изменения. |
| 7 | Постусловие | Активность кнопок настроена, бот реагирует в соответствии с заданными параметрами. |

#### 3.2.3.5.UС-5 Анализ заявок

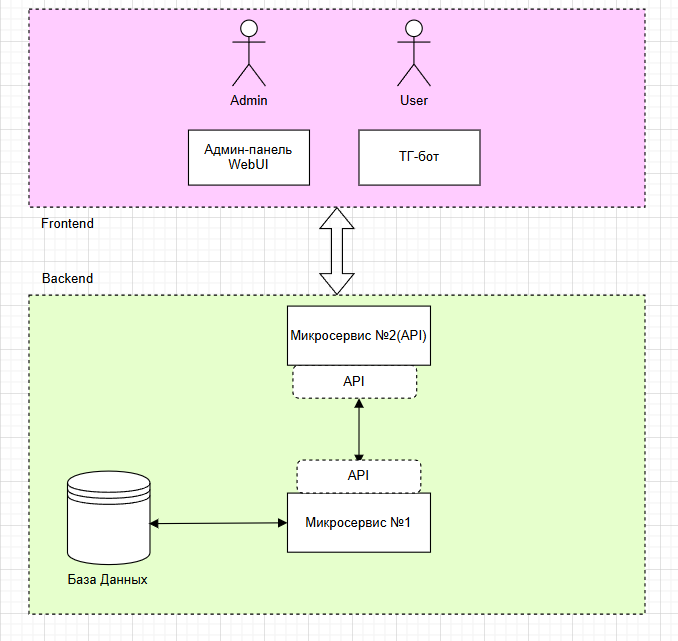
| № | Атрибут | Описание |
| --- | --- | --- |
| 1 | Участники | Администратор Система |
| 2 | Предусловие | Администратор авторизован в системе.  Доступен раздел аналитики. |
| 3 | Триггер | Администратор открывает раздел аналитики. |
| 4 | Основной сценарий | 1. Администратор заходит в админ-панель.  2. Открывает раздел аналитики.  3. Система отображает:   * Количество заявок по направлениям. * Статусы заявок (принятые, отклонённые, на рассмотрении, в ожидании). * Динамику подачи заявок по времени. * Географическое распределение заявок по городам.   4. Администратор анализирует данные и экспортирует их в Excel. |
| 5 | Альтернативный сценарий | Администратор использует фильтры для анализа (например, по направлениям или статусам). |
| 6 | Исключительный сценарий | Ошибка при построении витрин аналитики.  Система предлагает повторить запрос позже. |
| 7 | Постусловие | Аналитика доступна, администратор принимает управленческие решения на основе данных. |

## 3.3. Нефункциональные требования

| NFR-id | Нефункциональное требование |
| --- | --- |
| 1 | Язык интерфейса системы должен быть русским |
| 2 | Бот должен обрабатывать до 5 000 заявок на курс без ухудшения производительности. |
| 3 | Обработка данных в административной панели (например, фильтрация заявок) должна занимать не более 3 секунд. |
| 4 | Уровень доступности бота должен составлять не менее 95% в год. |
| 5 | Сообщения об ошибках должны быть информативными и содержать рекомендации по исправлению ошибок |
| 6 | Все ошибки должны логироваться и передаваться в систему мониторинга для оперативного устранения. |
| 7 | Бот должен работать во всех популярных мессенджер-клиентах Telegram (iOS, Android, Desktop, Web) |
| 8 | Административная панель должна корректно работать в современных браузерах, включая (Google Chrome, Firefox, Microsoft Edge). |

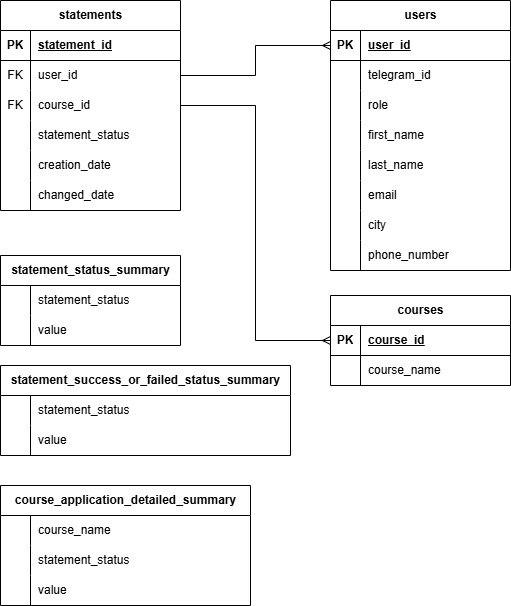
## 3.4 Архитектурные требования

### 3.4.1.Схема архитектуры

****

Также доступна отдельным файлом [по ссылке](https://drive.google.com/file/d/1aFm2ehElOevvBX2s8223Tts-CBgsM2b5/view)

### 3.4.2.Схема базы данных



Также доступна отдельным файлом [по ссылке](https://drive.google.com/file/d/1sUcitMvVXsXk856dE0IQL57nLrReME9Q/view)

**База данных: сущности и их атрибуты**

Таблица statements (заявки)

| Атрибут | Описание | Тип данных | Пример | Ключ | Обязательность |
| --- | --- | --- | --- | --- | --- |
| statement\_id | Уникальный идентификатор заявления | Integer | 101 | Первичный (PK) | Обязательный |
| user\_id | Идентификатор пользователя | Integer | 5 | Внешний (FK) | Обязательный |
| course\_id | Идентификатор курса | Integer | 2 | Внешний (FK) | Обязательный |
| statement\_status | Статус заявки | Varchar(30) | “На рассмотрении” | - | Обязательный |
| creation\_date | Дата создания заявки | Timestamp without timezone | 2025-01-25 14:30:00 | - | Обязательный |
| changed\_date | Дата последнего изменения | Timestamp without timezone | 2025-01-25 14:30:00 | - | Обязательный |

Таблица courses (направления обучения)

| Атрибут | Описание | Тип данных | Пример | Ключ | Обязательность |
| --- | --- | --- | --- | --- | --- |
| course\_id | Уникальный идентификатор курса | Integer | 1 | Первичный (РК) | Обязательный |
| course\_name | Название курса | Varchar(50) | “Java Development” | - | Обязательный |

Таблица users (пользователи)

| Атрибут | Описание | Тип данных | Пример | Ключ | Обязательность |
| --- | --- | --- | --- | --- | --- |
| user\_id | Уникальный идентификатор пользователя | Integer | 1 | Первичный ключ (PK) | Обязательный |
| telegram\_id | Идентификатор пользователя в Telegram | Bigint | 123456789 | - | Обязательный |
| role | Роль пользователя | Varchar (20) | "external\_user" | - | Обязательный |
| first\_name | Имя пользователя | Varchar (30) | “Михаил” | - | Обязательный |
| last\_name | Фамилия пользователя | Varchar (30) | “Иванов” | - | Обязательный |
| email | Email пользователя | Varchar (64) | “example@gmail.com” | - | Обязательный |
| city | Город проживания | Varchar (35) | “Пенза” | - | Обязательный |
| phone\_number | Номер телефона | Varchar (12) | "9535671023" | - | Обязательный |

Витрины данных

Витрина statement\_status\_summary - витрина с количеством заявок, сгруппированных по их статусам.

| Атрибут | Описание | Тип данных | Пример |
| --- | --- | --- | --- |
| statement\_status | Статус заявки (на рассмотрении, принято, отклонено) | VARCHAR(30) | На рассмотрении |
| value | Общее количество заявок в этом статусе | INTEGER | 25 |

Витрина statement\_success\_or\_failed\_status\_summary - отображение информации об общем числе принятых и отклонённых заявок.

| Атрибут | Описание | Тип данных | Пример |
| --- | --- | --- | --- |
| statement\_status | Статус заявки (на рассмотрении, принято, отклонено) | VARCHAR(30) | Принято |
| value | Количество заявок с указанным статусом | INTEGER | 15 |

Витрина course\_application\_detailed\_summary - детальная информация о заявках на курсы с учетом статуса.

| Атрибут | Описание | Тип данных | Пример |
| --- | --- | --- | --- |
| course\_name | Название курса | VARCHAR(50) | Аналитика |
| statement\_status | Статус заявки (на рассмотрении, принято, отклонено) | VARCHAR(30) | На рассмотрении |
| value | Общее количество заявок по курсу и статусу | INTEGER | 10 |

## 3.5. Спецификации для модулей

### 3.5.1. Модуль Telegram-бота

**Описание**: Этот модуль предоставляет интерфейс взаимодействия с пользователями через Telegram. Его цель — сбор данных, передача их в API, а также предоставление пользователю необходимой информации и ссылок.

**Функциональные требования:**

* Отправка приветственного сообщения новым пользователям.
* Пошаговый сбор данных (имя, фамилия, город проживания, номер телефона, email).
* Валидация введённых данных (имя, фамилия, город проживания, номер телефона, email).
* Предоставление списка доступных направлений для выбора.
* Поддержка кнопок “Подать заявку” и “Оставить предзаявку” с учетом периодов активности.
* Отправка уведомлений пользователю об успешной подаче заявки, об открытии нового набора.
* Предоставление ссылки на входные испытания после успешной подачи заявки.

**Методы:**

1. Отправка приветственного сообщения
   * Уведомление пользователя о возможностях бота.
2. Сбор данных пользователя
   * Последовательный ввод имени, фамилии, города проживания, телефона, электронной почты.
3. Валидация данных
   * Проверка корректности формата email и номера телефона.
4. Выбор направления обучения
   * Получение списка направлений с сервера.
5. Подача заявки
   * Отправка данных пользователя и выбранного направления на сервер.
6. Оставление предзаявки
   * Регистрирует пользователя для уведомления о начале набора.
7. Уведомления
   * Отправка уведомлений об успешной подаче заявки и начале набора.

**Алгоритм:**

1. Пользователь активирует бота.
2. Бот отправляет приветственное сообщение.
3. Пользователь выбирает действие («Подать заявку» или «Оставить предзаявку»).
4. Вводятся данные пользователя (проверяются на корректность).
5. Пользователь выбирает направление обучения.
6. Данные отправляются в API (POST запрос на /apply или /preapply).
7. Сервер возвращает результат: успех или ошибка.
8. Бот уведомляет пользователя о результате, отправляет ссылку на входные испытания.

**Входящие данные:**

* Имя, фамилия, город, email, номер телефона.
* Выбранное направление обучения.
* Тип действия: заявка или предзаявка.

**Исходящие данные:**

* Запросы к API микросервиса №2:
* POST /apply: регистрация заявки.
* POST /preapply: регистрация предзаявки.
* GET /courses: получение списка направлений.
* Уведомления пользователю о результате.

**Исключения:**

* Неверный формат email или номера телефона

Сообщение: «Введите данные в корректном формате».

* Ошибка сервера

Сообщение: «Сервер временно недоступен. Попробуйте позже».

* Недоступность направления

Сообщение: «Выбранное направление недоступно».

**База данных:**

Модуль использует:

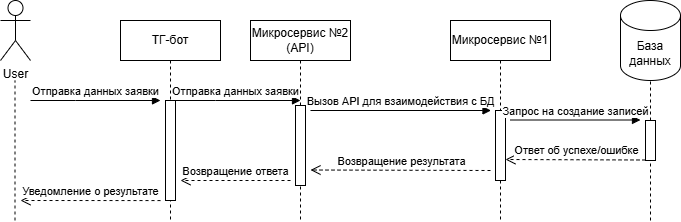
* Таблицу users для хранения данных пользователя.
* Таблицу statements для регистрации заявок.

Атрибуты связаны через user\_id и statement\_id.

**Нефункциональные требования:**

* Быстродействие: время отклика менее 1 секунды.
* Надежность: обработка ошибок при взаимодействии с API.
* Логи: запись всех операций (info/debug/error).

**Описание функциональности:**

****

### 3.5.2. Модуль базы данных

**Описание:** Модуль базы данных предназначен для хранения и обработки информации о заявках, пользователях, курсах, статусах заявок и логах операций. Он обеспечивает надежное хранение данных и поддерживает взаимодействие с другими модулями через API.

**Функциональные требования:**

* Поддержка операций добавления, чтения, обновления и удаления данных (Create, Read, Update, Delete).
* Хранение шаблонов сообщений (приветствия, уведомления) с возможностью изменения через админ-панель.

**Методы:**

1. Добавление данных: Создание новых записей (заявки, пользователи, курсы).
2. Чтение данных: Получение данных о заявках, пользователях и курсах.
3. Обновление данных: Обновление информации о заявках (например, изменение статуса).
4. Удаление данных: Удаление записей из базы (например, удаление старых заявок).

**Алгоритм:**

1. При получении запроса на создание или обновление данных проверяются переданные данные на соответствие требованиям.
2. Выполняется операция записи или обновления в базу данных.
3. В случае ошибок возвращается сообщение об ошибке.
4. Для операций чтения данные фильтруются по заданным параметрам, после чего возвращаются клиенту.

**Входящие данные:**

- Добавление заявки:

* user\_id: Идентификатор пользователя.
* course\_id: Идентификатор курса.
* statement\_status: Статус заявки (например, "На рассмотрении").
* creation\_date: Дата подачи заявки.
* changed\_date: Дата последнего изменения заявки.

- Добавление пользователя:

* telegram\_id: Идентификатор Telegram.
* role: Роль пользователя.
* first\_name, last\_name: Имя и фамилия.
* email: Электронная почта.
* city: Город проживания.
* phone\_number: Номер телефона.

**Исходящие данные:**

* Данные о заявках (фильтрация по статусу, курсу, дате).
* Данные о пользователях (по идентификатору или роли).
* Список курсов

**Исключения:**

1. Ошибка записи: Неверные или отсутствующие данные (например, некорректный формат email).
2. Ошибка чтения: Данные не найдены по заданным параметрам.
3. Ошибка обновления: Попытка обновления несуществующей записи.

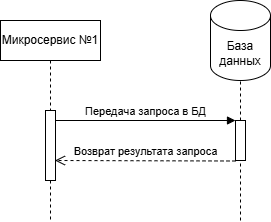
**Структура данных:**

* Таблица statements:
* Поля: statement\_id, user\_id, course\_id, statement\_status, creation\_date, changed\_date
* Таблица users:
* Поля: user\_id, telegram\_id, role, first\_name, last\_name, email, city, phone\_number
* Таблица courses:
* Поля: course\_id, course\_name.

**Нефункциональные требования:**

* Масштабируемость для увеличения объемов данных.
* Время выполнения операций с данными не должно превышать 1 секунды.
* Все операции с данными должны быть транзакционными для обеспечения целостности.

**Описание функциональности:**

****

### 3.5.3. Модуль административной панели

**Описание**: Модуль административной панели предоставляет интерфейс для администраторов, позволяющий управлять заявками, настраивать периоды активности кнопок, а также анализировать данные и выгружать отчёты.

**Функциональные требования:**

* Просмотр списка заявок с фильтрацией и сортировкой (по дате, статусу, курсу).
* Изменение статуса заявки (“В ожидании”, “На рассмотрении”, “Принята”, “Отклонена”).
* Настройка периодов активности кнопок “Подать заявку” и “Оставить предзаявку”.
* Отображение аналитики:
  + Количество заявок по направлениям.
  + Динамика подачи заявок.
  + Географическое распределение пользователей.

**Методы:**

1. **Просмотр заявок:**
   * Получение списка заявок с фильтрацией по статусу и направлению обучения.
   * Получение списка предзаявок с возможностью фильтрации по статусу.
2. **Изменение статуса заявки:**
   * Обновление статуса заявки (например, "На рассмотрении", "Принята", "Отклонена", “В ожидании”).
3. **Аналитика:**
   * Генерация аналитических данных (количество заявок по направлениям, динамика подачи заявок, географическое распределение пользователей).
   * Выгрузка отчётов по заявкам за указанный период.
4. **Настройка кнопок:**
   * Задание и изменение периодов активности кнопок "Подать заявку" и "Оставить предзаявку".

**Алгоритм:**

1. **Просмотр заявок:**
   * Администратор выбирает параметры фильтрации.
   * Запрос отправляется к API.
   * Результаты отображаются в интерфейсе.
2. **Изменение статуса заявки:**
   * Администратор выбирает заявку.
   * Выбирает новый статус.
   * Запрос отправляется к API для обновления данных.
   * Состояние заявки обновляется.
3. **Генерация аналитики:**
   * Администратор выбирает диапазон дат.
   * Запрос отправляется к API.
   * Возвращаются аналитические данные ( JSON).
4. **Настройка кнопок:**
   * Администратор задаёт периоды активности кнопок.
   * Данные отправляются в API.
   * Настройки обновляются в базе данных.

**Входящие данные:**

- Для просмотра заявок:

* Фильтры (например, статус заявки, направление обучения).

- Для изменения статуса:

* ID заявки.
* Новый статус.

- Для аналитики:

* Дата начала и окончания периода.

- Для настройки кнопок:

* Даты начала и окончания активности кнопок.

**Исходящие данные:**

* Подтверждения успешных операций (изменение статуса, обновление настроек).
* Обновления периодов активности кнопок.
* Списки заявок и предзаявок с информацией (пользователь, статус, курс, дата подачи).
* Аналитические данные (графики, сводки, числа).

**Исключения**

* Ошибка в формате даты или других входных данных.
* Сбой связи с API.
* Ошибка сохранения данных в базе данных.

**База данных:**

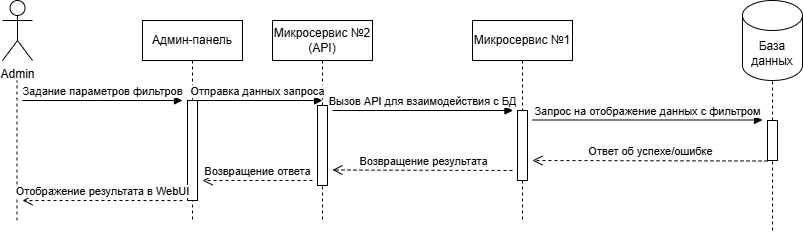
Модуль использует:

* Таблицу users для отображения информации о пользователях, связанных с заявками, а также использует ее данные для аналитики
* Таблицу statements для отображения списка заявок с фильтрацией по статусу, курсу, дате подачи, обновления статуса заявки, а также для подготовки данных для аналитики
* Таблицу courses для отображения названий курсов при просмотре заявок, фильтрации заявок по курсу, анализа популярности направлений.

**Нефункциональные требования:**

* Удобство интерфейса.
* Поддержка большого объема данных без потери производительности.
* Обработка данных и отображение интерфейса менее чем за 2 секунды.

**Описание функциональности:**

****

### 3.5.4. Микросервис №1 (База данных)

**Описание:** Микросервис №1 отвечает за управление всеми данными в базе данных, предоставление интерфейсов для работы с данными о пользователях, заявках и курсах, а также обработку операций чтения, создания, обновления и удаления данных. Он используется микросервисом №2 для взаимодействия с базой.

**Функциональные требования:**

* Обработка запросов на добавление, чтение, обновление данных.
* Предоставление интерфейсов для получения данных о заявках, курсах, статусах.

**Методы**

* GET /statements

Получение списка заявок.

* POST /apply

Создание новой заявки.

* PUT /statements/{id}

Обновление данных заявки по ID.

* GET /courses

Получение списка курсов.

**Алгоритм:**

1. При получении запроса микросервис обрабатывает запрос через контроллер.
2. Контроллер передаёт данные сервису для выполнения бизнес-логики.
3. Сервис обращается к репозиторию для взаимодействия с базой данных.
4. Репозиторий выполняет запросы к базе данных (чтение, создание, обновление, удаление).
5. Полученные данные передаются в ответе на запрос или подтверждается успешное выполнение операции.

**Входящие данные**

* Создание заявки:
  + Имя пользователя.
  + Фамилия пользователя.
  + Город проживания.
  + Номер телефона.
  + Электронная почта.
  + Идентификатор курса.
* Обновление заявки :
  + Новый статус заявки.

**Исходящие данные**

* Получение заявок:
  + Список заявок (ID, пользователь, курс, статус, дата создания и изменения).
* Создание заявки:
  + Подтверждение успешного создания заявки.
* Обновление заявки:
  + Подтверждение успешного обновления заявки.
* Получение списка курсов:
  + Список доступных курсов.

**Исключения**

1. Ошибка соединения с базой данных:
   * Код ошибки: 500.
   * Описание: Не удалось подключиться к базе данных.
2. Некорректный запрос:
   * Код ошибки: 400.
   * Описание: Ошибка валидации входных данных.
3. Данные не найдены:
   * Код ошибки: 404.
   * Описание: Указанный ID заявки или курс не существует.

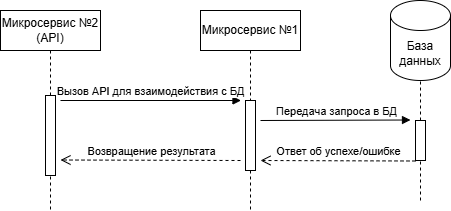
**База данных:**

* Таблица statements:
* Поля: statement\_id, user\_id, course\_id, statement\_status, creation\_date, changed\_date
* Таблица users:
* Поля: user\_id, telegram\_id, role, first\_name, last\_name, email, city, phone\_number
* Таблица courses:
* Поля: course\_id, course\_name.

**Нефункциональные требования:**

* Высокая скорость обработки запросов (до 1000 запросов/сек).
* Поддержка горизонтального масштабирования
* Логирование всех операций (запросы, ошибки) с уровнями debug, info, error.

**Описание функциональности:**

****

### 3.5.5. Микросервис №2 (API)

**Описание:** Микросервис №2 предоставляет API-интерфейс для взаимодействия Telegram-бота и административной панели с системой. Основные задачи включают обработку заявок, предзаявок, управление статусами и предоставление аналитики.

**Функциональные требования:**

* Получение данных о заявках, курсах и статусах из микросервиса базы данных.
* Обработка запросов от Telegram-бота на регистрацию заявки.
* Обработка запросов от административной панели на изменения статусов заявок.

**Методы:**

* POST /apply: Регистрация заявки.
* POST /preapply: Регистрация предзаявки.
* GET /courses: Получение списка направлений обучения.
* GET /statements: Получение списка заявок.
* GET /prestatements: Получение списка предзаявок.
* GET /analytics: Получение аналитики по заявкам.
* POST /analytics/report: Генерация и выгрузка отчета.
* PUT /settings\_button: Установка периодов активности кнопок.
* PUT /statement/{id}: Изменение статуса заявки.

**Алгоритм**

1. Принимает запрос от Telegram-бота или административной панели.
2. Проводит валидацию входных данных.
3. Направляет запрос к микросервису №1 для выполнения операций с базой данных.
4. Обрабатывает ответ от микросервиса №1.
5. Возвращает результат клиенту (Telegram-боту или админ-панели).

**Входящие данные:**

* Создание пользователя:
* Имя пользователя (firstName).
* Фамилия пользователя (lastName).
* Город проживания пользователя (city).
* Номер телефона пользователя (phone\_number).
* Электронная почта пользователя (email).
* Роль пользователя (role).
* Выбор курса
* ID выбранного курса (course\_id).
* Создание заявки
* Статус заявки (statement\_status).
* Активность кнопок
* Периоды активности кнопок (start\_date, end\_date).

**Исходящие данные:**

* Ссылка на тестовые испытания.
* Список заявок (с данными о пользователе, курсе, статусе, дате подачи).
* Список предзаявок (с данными о пользователе и курсе).
* Список доступных курсов.
* Информация об аналитике:
  + Общее количество заявок.
  + Количество заявок по статусам (принятые, отклоненные, в ожидании).
  + Динамика подачи заявок.
  + Географическое распределение пользователей.
* Подтверждение успешного изменения статуса заявки.
* Подтверждение изменения периодов активности кнопок.

**Исключения:**

* Ошибка соединения с базой данных:
  + Код ошибки: 500.
  + Описание: Не удалось подключиться к базе данных.
* Некорректный запрос:
  + Код ошибки: 400.
  + Описание: Ошибка валидации входных данных.
* Данные не найдены:
  + Код ошибки: 404.
  + Описание: Указанный ID заявки или курс не существует.

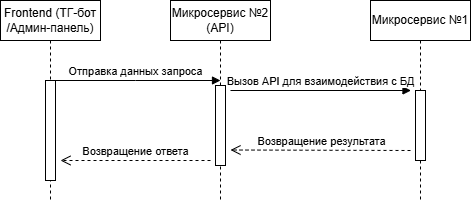
**База данных:**

* Таблица statements:
* Поля: statement\_id, user\_id, course\_id, statement\_status, creation\_date, changed\_date
* Таблица users:
* Поля: user\_id, telegram\_id, role, first\_name, last\_name, email, city, phone\_number
* Таблица courses:
* Поля: course\_id, course\_name.

**Нефункциональные требования:**

* Поддержка высокого количества одновременных соединений.
* Безопасность (авторизация и аутентификация для административной панели).
* Логи всех входящих и исходящих запросов.
* Поддержка увеличения нагрузки.

**Описание функциональности:**

****

**Спецификация API:** в отдельном файле **API\_TG-bot.yaml**