**Техническое задание для первого этапа хакатона "Smart ЖКХ"**

**1. Описание проекта**

**1.1 О проекте**

**Smart ЖКХ** - это современная цифровая платформа для взаимодействия жителей с управляющими компаниями и поставщиками услуг ЖКХ.

**1.2 Цель первого этапа**

Разработать MVP веб-сервиса с базовым функционалом для работы жителей со своими лицевыми счетами, просмотра начислений и оплаты коммунальных услуг.

**1.3 Ключевые пользователи**

* **Жители** - основные пользователи системы, которые оплачивают коммунальные услуги
* **Система** - автоматизированные процессы обработки данных

**2. Функциональные требования**

**2.1 Аутентификация и авторизация**

Система использует **Keycloak** для аутентификации и авторизации пользователей.



**Требования:**

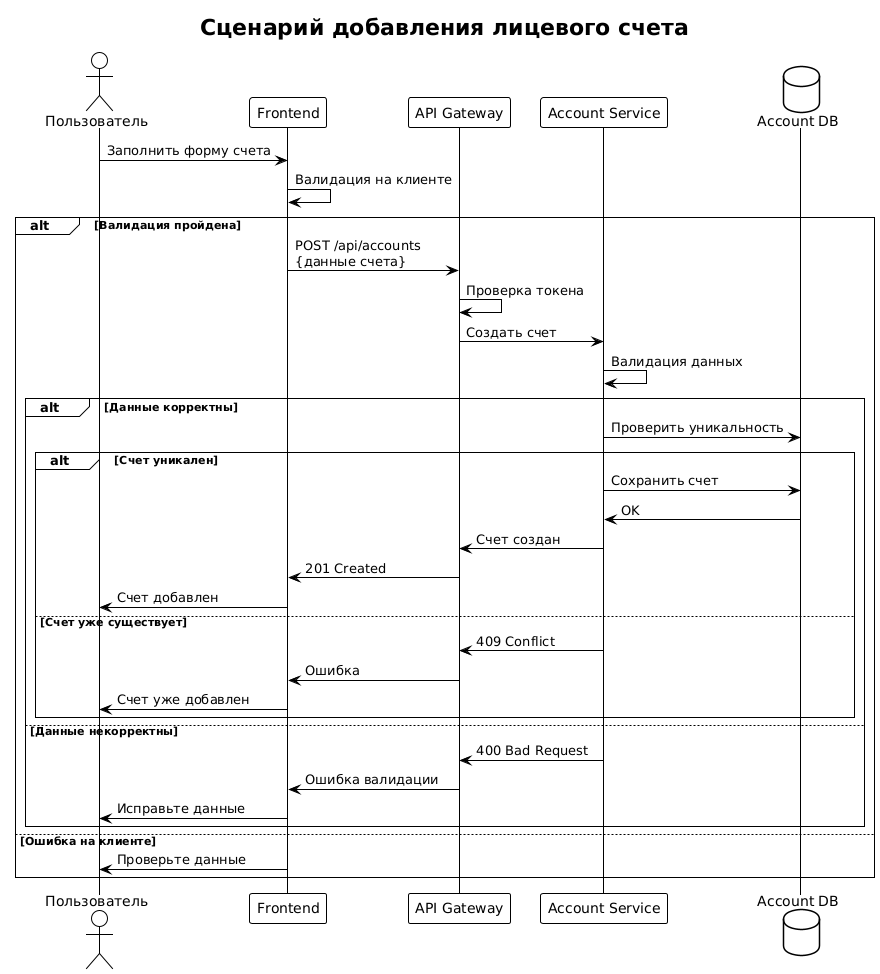
* Интеграция с предоставленным Keycloak
* Защита данных пользователей

**2.2 Управление лицевыми счетами**

Пользователь может управлять своими лицевыми счетами для оплаты услуг ЖКХ.

**Функции:**

* Добавление лицевого счета к профилю
* Просмотр списка своих счетов
* Удаление счета из профиля
* Выбор активного счета



**Данные лицевого счета:**

* Номер лицевого счета (10 цифр)
* Адрес помещения
* ФИО владельца
* Площадь помещения
* Количество проживающих
* Управляющая компания

**2.3 Просмотр начислений**

Пользователь может просматривать начисления за коммунальные услуги.



**Функции:**

* Просмотр начислений за выбранный период
* История начислений (минимум 6 месяцев)
* Детализация по видам услуг

**Виды услуг:**

* Холодное водоснабжение
* Горячее водоснабжение
* Электроэнергия
* Отопление
* Водоотведение
* Содержание жилья
* Капитальный ремонт

**2.4 Оплата услуг**

Система должна обеспечивать возможность оплаты коммунальных услуг.



**Функции:**

* Полная оплата всех услуг одним платежом
* История платежей
* Имитация процесса оплаты (без реальной интеграции с банками)

**2.5 Формирование квитанций**

Система должна позволять скачивать квитанции для оплаты.



**Требования:**

* Генерация квитанции в формате PDF
* Квитанция содержит все необходимые данные для оплаты
* Возможность скачать квитанцию за любой период

**Содержание квитанции:**

* Данные плательщика
* Адрес помещения
* Период оплаты
* Детализация по услугам
* Итоговая сумма
* QR-код для оплаты (опционально)

**3. Технические требования**

**3.1 Архитектура системы**

**Микросервисная архитектура**

Система должна быть реализована как набор микросервисов.



**Предоставляется организаторами:**

* Docker контейнеры с базовой структурой проекта
* Настроенный Keycloak для авторизации

**3.2 Требования к реализации**

**Обязательные требования:**

* **Язык программирования**: любой на выбор команды
* **База данных**: реляционная (PostgreSQL, MySQL, SQLite) или NoSQL
* **API**: RESTful архитектура
* **Безопасность**: валидация данных
* **Документация**: описание API и инструкция по запуску
* **Docker**: использование предоставленных контейнеров

**Рекомендуемые технологии:**

* **Backend**: Java/Spring, Python/Django, C/.NET, Node.js, Go
* **Frontend**: React, Vue, Angular или серверный рендеринг

**4. Нефункциональные требования**

**4.1 Производительность**

* Время ответа API не более 2 секунд
* Поддержка минимум 10 одновременных пользователей

**4.2 Безопасность**

* Валидация всех входных данных

**4.3 Удобство использования**

* Интуитивно понятный интерфейс
* Корректное отображение на desktop и mobile
* Понятные сообщения об ошибках

**5. Тестовые данные**

Для демонстрации работы системы необходимо подготовить:

* Минимум 3 тестовых пользователя
* По 1-2 лицевых счета на каждого
* История начислений за 6 месяцев
* Примеры оплаченных и неоплаченных периодов

**6. Критерии оценки**

**6.1 Полнота реализации**

* Реализованы все обязательные функции
* Система работает без критических ошибок
* Есть тестовые данные для демонстрации

**6.2 Качество кода**

* Читаемость и структурированность
* Следование принципам разработки
* Наличие комментариев в сложных местах

**6.3 Удобство использования**

* Понятный интерфейс
* Адаптивность под разные устройства
* Информативные сообщения

**6.4 Дополнительные баллы**

* Автоматические тесты
* Дополнительный функционал

**7. Что должно быть сдано**

**7.1 Требования к репозиторию**

Каждая команда должна предоставить **публичный Git репозиторий** (GitHub, GitLab, Bitbucket) с понятной структурой.

**7.2 Обязательное содержание README.md**

Smart ЖКХ - [Название команды]

Описание решения

Краткое описание вашей реализации (3-5 предложений)

Стек технологий

- Backend:

- Frontend:

- Database:

- Дополнительно:

Запуск приложения

Требования

- Docker и Docker Compose

- [Другие зависимости]

Установка

```bash

Команды для установки зависимостей

**Запуск**

Команды для запуска

**Реализованный функционал**

Указать какой функционал реализован

### 7.3 Демонстрация

- Работающее приложение

- Готовность продемонстрировать все функции

### 7.4 Процесс сдачи

1. До дедлайна команда должна:

- Завершить разработку

- Загрузить весь код в репозиторий

- Убедиться, что репозиторий публичный

- Проверить, что приложение запускается по инструкции

2. Отправка ссылки:

- Заполнить форму

- Указать ссылку на репозиторий

- Указать контакты команды

- Подтвердить готовность к проверке

3. После отправки:

- Не вносить изменения в ветку main/master

- Исправления загружать только в отдельные ветки

- Быть готовыми ответить на вопросы жюри

### 7.5 Рекомендации

- Проверьте работоспособность: попросите коллегу склонировать репозиторий и запустить по вашей инструкции

- Используйте .gitignore: не загружайте в репозиторий node\_modules, виртуальные окружения, файлы IDE

- Версионирование: используйте осмысленные commit-сообщения

- Документация: лучше простая, но понятная документация, чем сложная и запутанная

### 7.6 Техническая поддержка

При проблемах со сдачей работы:

- Написать в официальный чат хакатона в Telegram

- Обратиться к организаторам до истечения дедлайна

- Сохранить скриншоты в случае технических проблем

## 8. Ограничения

- НЕ требуется реальная интеграция с платежными системами

- НЕ требуется интеграция с внешними API управляющих компаний

- НЕ требуется разработка мобильного приложения

- НЕ требуется сложная бизнес-логика расчета тарифов

Желаем удачи всем участникам!

Помните: лучше сделать базовый функционал качественно, чем пытаться реализовать все возможности с ошибками.