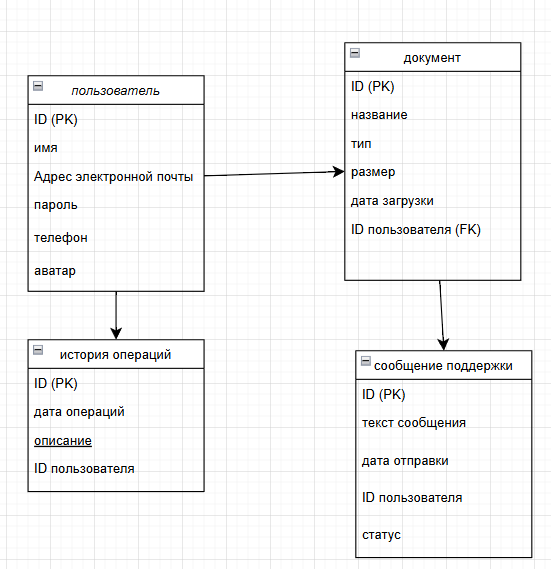
Задание 1

Техническое задание на разработку веб-приложения "Личный кабинет пользователя"  
  
1. **Введение**  
Цель этого проекта — разработать веб-приложение "Личный кабинет пользователя", которое позволит пользователям управлять своим профилем, просматривать данные и взаимодействовать с различными функциональными возможностями системы.  
  
 2. **Описание проекта**  
Веб-приложение будет предоставлять пользователям следующий функционал:  
- Регистрация и авторизация пользователей  
- Редактирование профиля пользователя  
- Просмотр истории операций  
- Возможность загрузки документов  
- Связь с службой поддержки  
  
 3. **Требования к функционалу**  
  
3.1. Регистрация и авторизация  
- Пользователь должен иметь возможность зарегистрироваться, указав имя, email и пароль.  
- После регистрации пользователю должно быть отправлено подтверждение по email.  
- Пользователь должен иметь возможность авторизоваться в системе, используя email и пароль.  
- Реализовать функцию восстановления пароля.  
  
3.2. Профиль пользователя  
- Пользователь должен иметь возможность редактировать свой профиль (имя, email, телефон, пароль).  
- Пользователь должен иметь возможность загрузить аватар.  
  
3.3. История операций  
- Реализовать раздел "История операций", где пользователь сможет видеть список своих действий (например, загрузка документов, изменения в профиле).  
- Возможность фильтрации и сортировки истории по дате.  
  
 3.4. Загрузка документов  
- Пользователь должен иметь возможность загружать документы (до 5 МБ на файл).  
- Поддерживаемые форматы: PDF, DOCX, JPG, PNG.  
  
 3.5. Служба поддержки  
- Реализовать контактную форму для связи со службой поддержки.  
- Пользователь должен иметь возможность оставлять сообщения и получать уведомления о статусе запроса.  
  
 4. **Требования к дизайну**  
- Дизайн интерфейса должен быть современным и адаптивным (подстраиваться под различные размеры экранов).  
- Использование цветовой палитры: [указать цвета].  
- Шрифты: [указать шрифты].  
  
 5. **Технические требования**  
- Язык разработки: Python (Django/Flask) или JavaScript (Node.js/React).  
- База данных: PostgreSQL или MySQL.  
- Разработка должна осуществляться по методологии Agile.  
  
 6. **Критерии приемки**  
- Функционал должен быть протестирован и соответствовать всем требованиям, указанным в ТЗ.  
- Приложение должно быть доступно на тестовом сервере для окончательной проверки перед запуском.  
  
 7. **Сроки**  
- Дата начала разработки: [указать дату].  
- Дата окончания разработки и тестирования: [указать дату].  
  
 8. **Ответственные лица**  
- Заказчик: [Имя, контактные данные]  
- Исполнитель: [Имя, контактные данные]

Задание 2



Здание 3

1. Языки программирования

a. JavaScript

- Почему: JavaScript является основным языком для веб-разработки, позволяя создавать интерактивные элементы на стороне клиента. Он поддерживается всеми современными браузерами и является языком, на котором строится много популярных библиотек и фреймворков.

b. Python

- Почему: Python является удобным и высокоуровневым языком, который широко используется для серверной разработки. Интуитивно понятный синтаксис и мощные библиотеки упрощают такие задачи, как обработка данных, работа с API и взаимодействие с базами данных.

2. Фреймворки

a. Для фронтенда:

- React.js

- Почему: React позволяет создавать динамичные пользовательские интерфейсы. Он поддерживает компонентный подход, что облегчает переиспользование кода и уменьшает количество ошибок. React также имеет большую экосистему и сообщество, что обеспечивает наличие множества библиотек и инструментов.

- Vue.js

- Почему: Vue проще в освоении и подходит для небольших и средних приложений. Его можно легко интегрировать в существующий проект и использовать для создания интуитивно понятного интерфейса.

b. Для бэкенда:

- Django (на Python)

- Почему: Django — это высокоуровневый фреймворк, который позволяет быстро разрабатывать веб-приложения. Он включает множество встроенных функций, таких как аутентификация пользователей, работа с базами данных и защита от распространенных веб-уязвимостей.

- Flask (на Python)

- Почему: Flask — это легковесный фреймворк, который предоставляет большую гибкость для разработчиков. Отлично подходит для небольших проектов, когда не требуется много встроенных функций.

- Node.js (JavaScript)

- Почему: Node.js позволяет использовать JavaScript на стороне сервера, что может привести к более однородному коду как на клиенте, так и на сервере. Это особенно полезно для приложений, которые требуют высокого уровня взаимодействия в реальном времени.

3. Базы данных

a. PostgreSQL

- Почему: PostgreSQL — это мощная объектно-реляционная база данных, которая поддерживает сложные запросы, транзакции и функции расширяемости. Она хорошо подходит для приложений с высокими требованиями к целостности данных и поддерживает работающие с большими объемами данных.

b. MySQL

- Почему: MySQL — это одна из самых популярных реляционных баз данных. Она проста в использовании, эффективна и имеет большое сообщество, что обеспечивает хорошую поддержку и документацию.

4. Дополнительные технологии

a. Docker

- Почему: Docker позволяет упаковать приложение и все его зависимости в контейнеры, что делает развертывание приложения более предсказуемым и удобным. Это также упрощает настройку среды разработки и тестирования.

b. RESTful API

- Почему: Использование RESTful API обеспечивает четкое разделение между клиентом и сервером. Это позволяет создавать масштабируемые приложения и облегчает взаимодействие с другими сервисами.

c. Git

- Почему: Git используется для контроля версий и совместной работы над кодом. Он позволяет отслеживать изменения в коде, управлять ветками разработки и обеспечивать безопасное хранение проекта.

Задание 4