Problem Set

Проектирование веб-приложения

Этапы разработки веб-приложения на примере проекта Krug

Ответьте в свободной форме

Дедлайн: 29 августа 17:00

Summer Freestyle Workshop

1 Идея и постановка задачи

1.1 Контекст

Проект Krug — социальная сеть выпускников университета с функциями наставничества и AI-ассистентом. Выпускники могут:

- находить менторов и подопечных (mentees);
- общаться в реальном времени;
- получать карьерные советы от AI-ассистента;
- участвовать в событиях (вебинары, встречи).

1.2 Задание 1.1. Проблема и цель

Любое приложение начинается с проблемы, которую оно решает.

Какую проблему решает Krug? Сформулируйте своими словами, для кого это приложение и почему оно нужно.

1.3 Задание 1.2. Ваша идея

Представьте, что вы хотите создать простое веб-приложение для своей области (экономика, финансы, ML, данные).

Какую проблему из вашей работы/учебы вы хотели бы решить с помощью веб-приложения? Опишите идею в 2–3 предложениях.

2 Функциональные требования (FR)

Функциональные требования (Functional Requirements) описывают, что система должна делать. Это конкретные, проверяемые утверждения.

2.1 Пример: реальные FR из проекта Krug

ID	Требование
FR-001	Система должна предоставлять адаптивный SPA
FR-002	Система должна управлять аутентификацией через JWT
FR-003	Система должна предоставлять интерфейс профиля
FR-004	Система должна управлять наставничеством
FR-005	Система должна реализовывать обмен сообщениями
FR-006	Система должна отображать события
FR-007	Система должна предоставлять чат AI-ассистента

2.2 Задание 2.1. Составьте FR для своей идеи

Для вашего приложения из Задания 1.2 сформулируйте 4–5 функциональных требований. Каждое должно быть конкретным и проверяемым (начинается с «Система должна...»).

2.3 Задание 2.2. Проверка FR

Посмотрите на свои FR. Для каждого ответьте: как вы проверите, что это требование выполнено?

3 Нефункциональные требования (NFR)

Нефункциональные требования (Non-Functional Requirements) описывают как система должна работать: производительность, безопасность, масштабируемость, удобство.

3.1 Пример: NFR из проекта Krug

ID	Требование
NFR-001	Время загрузки страницы < 2 секунд
NFR-002	Время ответа AI-ассистента < 5 секунд
NFR-003	Доставка сообщения < 500 мс
NFR-004	Поддержка до 1000 пользователей
NFR-005	Доступность $\geq 99\%$
NFR-006	Компонентная архитектура

3.2 Задание 3.1. Составьте NFR для своей системы

Сформулируйте 3–4 нефункциональных требования для вашего приложения. Подумайте о: скорости, количестве пользователей, безопасности, удобстве.

3.3 Задание 3.2. Разница FR и NFR

Объясните своими словами: чем функциональные требования отличаются от нефункциональных? Приведите по одному примеру каждого из вашего приложения.

4 Модель данных

Прежде чем писать код, нужно понять, какие данные будут храниться и как они связаны.

4.1 Пример: сущности Krug

Сущность	Ключевые поля	Связи
User	id, login, email, balance	участвует в Mentorship
Mentorship	id, mentor_id, mentee_id, status	имеет много Task
Task	id, mentorship_id, title, status	принадлежит Mentorship
Message	id, sender_id, receiver_id	принадлежит User
Event	id, title, description, datetime	создан User
Interview	id, title, transcript, indexed	для AI-ассистента

4.2 Задание 4.1. Сущности вашего приложения

Для вашего приложения из Задания 1.2 определите 3–4 сущности (таблицы) и 2–3 ключевых поля для каждой.

4.3 Задание 4.2. Связи между сущностями

Опишите связи между вашими сущностями. Например: «Один User может иметь много Project», «Один Project принадлежит одному User».

5 Архитектурные решения (ADR)

Architecture Decision Record (ADR) — это документ, который фиксирует важное архитектурное решение и объясняет, почему выбрали именно так.

5.1 Пример: ADR из проекта Krug

ADR-001: Vue 3 с Composition API

- Контекст: фронтенд требует современный фреймворк с хорошей поддержкой TypeScript.
- Решение: использовать Vue 3 с Composition API.
- Последствия: лучше поддержка TypeScript, но есть кривая обучения.

ADR-003: DeepSeek V4 для AI-ассистента

- Контекст: AI-ассистент обрабатывает 50–100 запросов в день.
- Решение: использовать DeepSeek V4 Pro API.
- Последствия: лучший баланс цена/качество, но зависимость от внешнего сервиса.

5.2 Задание 5.1. Примите архитектурное решение

Для вашего приложения выберите одно важное архитектурное решение (например: какой фреймворк, какую базу данных, как хранить данные). Опишите его по формату ADR.

5.3 Задание 5.2. Почему ADR полезны?

Объясните своими словами: зачем фиксировать архитектурные решения в письменном виде? Что это даёт команде через полгода?

6 Этапы разработки

Теперь, когда у вас есть требования и архитектура, посмотрим на этапы разработки.

6.1 Этапы разработки веб-приложения (на примере Krug)

Этап	Что делается	Пример из Krug
1. Идея	Определяем проблему	Соцсеть для alumni
2. Требования	Формулируем FR и NFR	FR-001...FR-007
3. Модель данных	Проектируем сущности	User, Mentorship, Task
4. Архитектура	Принимаем ADR	Vue 3, FastAPI, Qdrant
5. Бэкенд	Пишем API	FastAPI-роутеры
6. Фронтенд	Строим интерфейс	Vue 3, Pinia, Axios
7. Real-time	Мгновенные обновления	Socket.IO
8. AI-интеграция	Подключаем LLM	RAG-ассистент
9. Тестирование	Проверяем качество	113 unit + 71 integration
10. Деплой	Запускаем в продакшн	Docker Compose

6.2 Задание 6.1. Расставьте этапы для вашего приложения

Для вашего приложения из Задания 1.2 расположите этапы в правильном порядке и кратко опишите, что будет на каждом.

6.3 Задание 6.2. Что было бы самым сложным?

Какой этап разработки вашего приложения был бы для вас самым сложным и почему?

7 Итоговое задание

7.1 Задание 7.1. Полный план вашего приложения

Соберите всё вместе. Для вашего приложения из Задания 1.2 напишите краткий план:

1. Проблема (1 предложение)
2. Целевая аудитория (кто будет пользоваться)
3. 3 функциональных требования
4. 2 нефункциональных требования
5. 3 сущности данных
6. 1 архитектурное решение (ADR)

7.2 Задание 7.2. Рефлексия

Что нового вы узнали о процессе разработки веб-приложений? Что из этого пригодится вам в вашей работе (экономика, финансы, ML, данные)?

Критерии сдачи

Problem set считается сданным, если:

☐ Заполнены все поля для ответов

☐ Ответы написаны своими словами (не скопированы)

☐ Понятно, чем FR отличается от NFR

☐ Понятно, зачем нужны ADR

☐ Есть осмысленный план своего приложения

Ключевые выводы

После этого problem set вы:

1. Поняли этапы разработки веб-приложения (от идеи до деплоя)
2. Научились формулировать функциональные и нефункциональные требования
3. Поняли, как проектировать модель данных (сущности и связи)

4. Узнали, что такое ADR и зачем фиксировать архитектурные решения
5. Спроектировали собственное приложение для своей области

Главное: разработка начинается не с кода, а с мышления.