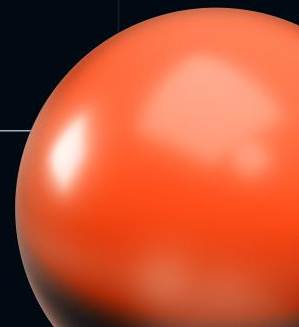
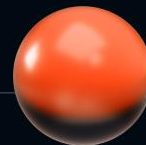


Что ChatGPT знает про анализ и проектирование ИТ-систем



**Юрий
Куприянов**

Systems.Education



Systems • Education

12 лет учим создавать успешные ИТ-продукты

+ New chat

New chat

Система цифровых расшире

Online Shop Order Process

Traffic funnel for course conver

User Stories Overview.

ChatGPT for Software Dev

Online school system.

New chat

Increasing Revenue for Analyst

Clear conversations

Upgrade to Plus

NEW

Dark mode

Updates & FAQ

Log out

ChatGPT



Examples

"Explain quantum computing in simple terms" →

"Got any creative ideas for a 10 year old's birthday?" →

"How do I make an HTTP request in Javascript?" →



Capabilities

Remembers what user said earlier in the conversation

Allows user to provide follow-up corrections

Trained to decline inappropriate requests



Limitations

May occasionally generate incorrect information

May occasionally produce harmful instructions or biased content

Limited knowledge of world and events after 2021





Что нужно знать про ChatGPT

2017: Google предложил архитектуру нейросетей “Трансформер”

2018: OpenAI создал и обучил **GPT – Generative Pre-trained Transformer.**

2019: OpenAI “не выпустила” **GPT-2** с **1,5** млрд. параметров.

2020: появилась **GPT-3** с **175** млрд. параметров (davinci).

2021: **GPT-3** натренировали несколько моделей под разные задачи – **GPT-3.5.**

2022: **ChatGPT** – это та же **GPT-3.5**, “зафайнтюненная” под ведение диалога и написание кода + интерфейс через чат, а не через **API.**

1 млн. пользователей ChatGPT набрал за 5 дней.



Что нужно знать про ChatGPT

GPT анализирует весь диалог и вставляет слово, наиболее подходящее по `<s>смыслу</s>` статистике, так генерируется всё предложение или код.

“Валюта” GPT – токены. **1000 токенов** – это примерно **750 слов** (англ.)

Русских слов в обучающей выборке GPT-3 – **0.18%**. Английских – **92%**.

У ChatGPT *окно контекста*, то есть память диалога – **8000** токенов. У GPT-4 контекст может быть до **32k**.

ChatGPT ничего не знает о событиях после Q4 2021 года.

ChatGPT не умеет ходить в Интернет за новой информацией.

29 марта Илон Маск, Стив Возняк и ещё более 1000 предпринимателей и ученых призвали [остановить исследования в области ИИ](#).



Как разговаривать с ChatGPT

- **Просто разговаривать**, как с человеком, у которого вы что-то спрашиваете или просите сделать.
- **Роль**: попросить **выступить из какой-то роли** – бизнес-аналитика, архитектора или менеджера проекта (иногда от роли зависит качество ответа!)
- **Цепочка запросов**: не спрашивать сразу о чем-то, а сначала наводящими вопросами подвести к мысли.
- Попросить **объяснить логику ответа** шаг за шагом.
- Можно **давать пример**, например – нотации, и просить сделать, как в примере.



Как разговаривать с ChatGPT

- В запросе нужно указывать, **в каком виде вы ожидаете ответ** (в виде таблицы, в виде кода, как набор вопросов к вам и т.п.)
- **Язык: английский.** Быстрее и длиннее ответы получаются (больше база, меньше токенов тратится).
- Если что-то не так – можно **просить исправить**.
- **Разбивать сложную задачу на несколько**, и просить улучшать постепенно, отдельными запросами.
- Коллекция промптов: <https://github.com/f/awesome-chatgpt-prompts>

Начнем с идеи приложения



Я хочу создать приложение, в котором люди будут делиться рецептами. Предложи функции, которые могут быть в этом приложении.



Конечно, я могу предложить некоторые функции, которые могут быть полезными для вашего приложения для обмена рецептами:



1. Регистрация и профиль: позвольте пользователям создать профиль и сохранять свои любимые рецепты, а также устанавливать предпочтения в отношении кухни и ингредиентов.
2. Поиск и фильтрация: разрешите пользователям искать рецепты по названию, ингредиентам или типу кухни. Функция фильтрации может помочь пользователям находить рецепты в соответствии с их диетическими потребностями, такими как веганские или безглютеновые.
3. Добавление рецептов: дайте пользователям возможность добавлять свои собственные рецепты с фотографиями и описанием, а также могут присоединить теги для легкого поиска.
4. Оценки и комментарии: позвольте пользователям оценивать и комментировать рецепты, а также видеть, сколько раз рецепт был добавлен в избранное.
5. Подписки на пользователей: дайте пользователям возможность подписаться на других пользователей и получать уведомления о новых рецептах, добавленных ими.
6. Рекомендации: используйте алгоритмы рекомендаций, чтобы предлагать пользователям рецепты, которые могут быть им интересны.

Начнем с идеи приложения

1. Регистрация и профиль: позвольте пользователям создать профиль и сохранять свои любимые рецепты, а также устанавливать предпочтения в отношении кухни и ингредиентов.
2. Поиск и фильтрация: разрешите пользователям искать рецепты по названию, ингредиентам или типу кухни. Функция фильтрации может помочь пользователям находить рецепты в соответствии с их диетическими потребностями, такими как веганские или безглютеновые.
3. Добавление рецептов: дайте пользователям возможность добавлять свои собственные рецепты с фотографиями и описанием, а также могут присоединить теги для легкого поиска.
4. Оценки и комментарии: позвольте пользователям оценивать и комментировать рецепты, а также видеть, сколько раз рецепт был добавлен в избранное.
5. Подписки на пользователей: дайте пользователям возможность подписаться на других пользователей и получать уведомления о новых рецептах, добавленных ими.
6. Рекомендации: используйте алгоритмы рекомендаций, чтобы предлагать пользователям рецепты, которые могут быть им интересны.
7. Категории: создайте различные категории, чтобы пользователи могли легко найти рецепты, такие как завтрак, обед, ужин, десерты, закуски и т.д.
8. Интерактивные инструкции: добавьте возможность создания интерактивных инструкций по готовке с помощью Regenerate response и или аудио, чтобы пользователи могли легко следовать рецепту.



Просим задать вопросы из роли



Я хочу, чтобы ты действовал как бизнес-аналитик и задал мне вопросы про приложение с рецептами, важные для успеха и реализации проекта.



Конечно, я могу помочь вам сформулировать важные вопросы, которые могут повлиять на успех и реализацию проекта приложения с рецептами. Некоторые из таких вопросов могут быть:



1. Какова целевая аудитория приложения с рецептами и какие потребности эта аудитория имеет в отношении приложения?
2. Какие конкуренты уже существуют на рынке приложений с рецептами, и как наше приложение будет отличаться от них?
3. Каковы будут функциональные и нефункциональные требования к приложению с рецептами?
4. Каков бюджет на разработку приложения и какие ресурсы (людские, технические, финансовые) будут выделены для реализации проекта?
5. Как будут проводиться маркетинговые и рекламные кампании для привлечения пользователей к нашему приложению?
6. Как будет организован процесс тестирования приложения и обеспечения его безопасности?
7. Как будет организована техническая поддержка пользователей приложения и какие каналы связи с ними будут использоваться?
8. Какие метрики будут использоваться для оценки эффективности приложения и как будет проводиться анализ этих метрик?

Сужаем скуп задач



Давай стартуем с минимального набора функций. Что бы это могло быть?



Конечно, мы можем начать с минимального набора функций. Некоторые функции, которые могут быть полезными для вашего приложения для обмена рецептами:



1. Регистрация и профиль: позвольте пользователям создать профиль, указав свое имя и адрес электронной почты.
2. Добавление рецептов: дайте пользователям возможность добавлять свои собственные рецепты с фотографиями и описанием.
3. Поиск: позвольте пользователям искать рецепты по названию.
4. Комментарии: позвольте пользователям комментировать рецепты.
5. Избранное: дайте пользователям возможность добавлять рецепты в избранное.
6. Авторизация: защитите доступ к редактированию и удалению рецептов через авторизацию.
7. Удаление: дайте пользователю возможность удалять свои собственные рецепты.

Это минимальный набор функций, который позволит пользователям создавать и просматривать рецепты, а также добавлять комментарии и избранное. Вы можете добавить больше функций по мере необходимости, чтобы улучшить и расширить функциональность вашего приложения в будущем.

Запрашиваем ответ в определенном формате



Оформи этот набор функций в виде пользовательских историй и сведи в таблицу



Пользовательские истории:

1. Как **пользователь**, я хочу **создать профиль**, чтобы сохранять свои рецепты и добавлять комментарии.
2. Как **пользователь**, я хочу **добавить новый рецепт** с названием, ингредиентами, описанием и фотографией, чтобы другие пользователи могли видеть и использовать мой рецепт.
3. Как **пользователь**, я хочу **искать рецепты по названию**, чтобы быстро найти нужный рецепт.
4. Как **пользователь**, я хочу **оставлять комментарии к рецептам**, чтобы поделиться своим опытом и обратной связью.
5. Как **пользователь**, я хочу **добавлять рецепты в избранное**, чтобы легко найти их в будущем.
6. Как **пользователь**, я хочу **авторизоваться**, чтобы иметь возможность редактировать или удалять свои собственные рецепты.
7. Как **пользователь**, я хочу иметь **возможность удалить свои собственные рецепты**, если я больше не хочу, чтобы они были доступны другим пользователям.



Запрашиваем ответ в определенном формате

Сводная таблица:

Функция	Краткое описание
Регистрация и профиль	Пользователь может создать профиль, указав имя и адрес электронной почты.
Добавление рецептов	Пользователь может добавить новый рецепт с названием, ингредиентами, описанием и фотографией.
Поиск	Пользователь может искать рецепты по названию.
Комментарии	Пользователь может оставлять комментарии к рецептам.
Избранное	Пользователь может добавлять рецепты в избранное.
Авторизация	Пользователь должен авторизоваться, чтобы иметь возможность редактировать или удалять свои собственные рецепты.
Удаление	Пользователь может удалить свои собственные рецепты.

Надеюсь, что это поможет вам лучше понять, какие функции включить в ваше приложение для обмена рецептами.

Можно вставить в Miro или импортировать в Trello.

Можно было бы поработать с бэклогом дальше: уточнить функции, детализировать, приоритизировать и т.д.

Регистрация
и профиль

Авторизация

Добавление
рецептов

Поиск

Комментарии

Избранное

Удаление

Пользователь может создать профиль, указав имя и адрес электронной почты.

Пользователь должен авторизоваться, чтобы иметь возможность редактировать или удалять свои собственные рецепты.

Пользователь может добавить новый рецепт с названием, ингредиентами, описанием и фотографией.

Пользователь может искать рецепты по названию.

Пользователь может оставлять комментарии к рецептам.

Пользователь может добавлять рецепты в избранное.

Пользователь может удалить свои собственные рецепты.

Уточняем роли пользователей



Если у нас есть комментарии, должен быть модератор. А какие вообще роли пользователей будут у нас в приложении?



Хороший вопрос! Действительно, если в приложении есть комментарии, то необходим модератор, чтобы управлять контентом и удалять нежелательные или оскорбительные комментарии. Кроме того, в зависимости от функционала, которые мы добавим, может понадобиться и другие роли пользователей.



Вот несколько потенциальных ролей пользователей для нашего приложения:

1. Гость: пользователь, который не зарегистрирован в приложении, но может просматривать доступные рецепты и комментарии.
2. Зарегистрированный пользователь: пользователь, который зарегистрировался в приложении и имеет свой профиль, может добавлять рецепты, оставлять комментарии и добавлять рецепты в избранное.
3. Модератор: пользователь, который может управлять контентом, удалять комментарии и редактировать нежелательный контент.
4. Администратор: пользователь, который имеет полный доступ к управлению приложением, может добавлять и удалять других пользователей, назначать роли и изменять настройки приложения.

Конечно, это только примеры ролей, и конкретные роли будут зависеть от потребностей вашего приложения и функционала, который вы реализуете.

Просим придумать архитектуру



Давай вернемся к общей архитектуре. Можешь изобразить общую архитектуру приложения с рецептами в нотации mermaid.js?



Конечно, вот пример общей архитектуры приложения с использованием Mermaid.js:



mermaid

Copy code

```
graph TD;
  User((Пользователь))
  subgraph "Мобильное приложение"
    RecipeView((Просмотр рецептов))
    RecipeCreate((Создание рецептов))
    RecipeUpdate((Редактирование рецептов))
    RecipeDelete((Удаление рецептов))
  end
  subgraph "Веб-интерфейс"
    RecipeListView((Список рецептов))
    RecipeDetailView((Просмотр рецепта))
    RecipeCreateView((Создание рецепта))
    RecipeUpdateView((Редактирование рецепта))
    RecipeDeleteView((Удаление рецепта))
  end
  RecipeAPI((API рецептов))
  DB((База данных))
  User --> RecipeView
  User --> RecipeCreate
  User --> RecipeUpdate
  User --> RecipeDelete
  RecipeView --> RecipeAPI
```

Ошибся! Но
попробовал ещё
раз. Получилось.



Mermaid говорит - ошибка парсинга в 3 строке



Извините за ошибку. Попробуйте следующий код:



mermaid

Copy code

```
graph LR
  User((Пользователь))
  API(API)
  DB((База данных))



  User -- Просмотр рецептов --> API
  User -- Добавление рецептов --> API
  User -- Поиск рецептов --> API
  User -- Комментирование рецептов --> API
  User -- Лайки --> API
  API -- Получение рецептов --> DB
  API -- Добавление рецептов --> DB
  API -- Поиск рецептов --> DB
  API -- Комментирование рецептов --> DB
  API -- Лайки --> DB
```

Данный код создаст граф с тремя блоками: Пользователь, API и База данных. Между пользователями и API нарисованы стрелки, обозначающие функции приложения, доступные для пользователя. Между API и Базой данных также нарисованы стрелки, обозначающие какие операции с базой данных API может выполнять.




Да, так получилось. Давай теперь в этом формате всю архитектуру



Конечно, вот пример общей архитектуры приложения с рецептами в нотации Mermaid:  

mermaid

 Copy code

```
graph TD
  subgraph User Interface
    A1(Web) -- HTTP Request --> B(App Server)
    A2(Mobile) -- HTTP Request --> B(App Server)
  end

  subgraph App Server
    B -- Request to search recipes --> C(Elasticsearch)
    B -- Request to add, edit or delete recipes --> D(RDBMS)
    B -- Request to retrieve recipe details --> D
    B -- Request to add, edit or delete comments --> E(RDBMS)
    B -- Request to retrieve comments --> E
    B -- Request to authenticate user --> F(Auth Server)
  end

  subgraph Auth Server
    F -- Authenticate User --> G1(JWT Tokens)
  end
```

Mermaid </> Code



Auto sync



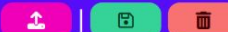
DOCS

```

1 graph TD
2   subgraph User Interface
3     A1(Web) -- HTTP Request --> B(App Server)
4     A2(Mobile) -- HTTP Request --> B(App Server)
5   end
6
7   subgraph App Server
8     B -- Request to search recipes --> C(Elasticsearch)
9     B -- Request to add, edit or delete recipes -->
10    B -- Request to retrieve recipe details --> D
11    B -- Request to add, edit or delete comments -->
12    B -- Request to retrieve comments --> E
13    B -- Request to authenticate user --> F(Auth Server)
14  end
15
16  subgraph Auth Server
17    F -- Authenticate User --> G1(JWT Tokens)
18  end
19
20  subgraph Data Storage
21    C(Elasticsearch) --> |Recipe Data| H1
22    D(RDBMS) --> |Recipe Data| H1
  
```

> Sample Diagrams

> History



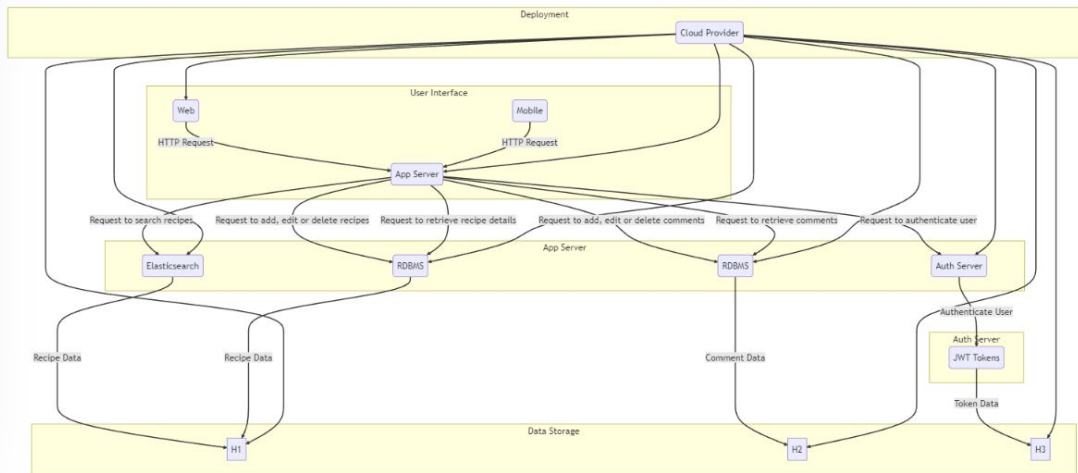
> Actions

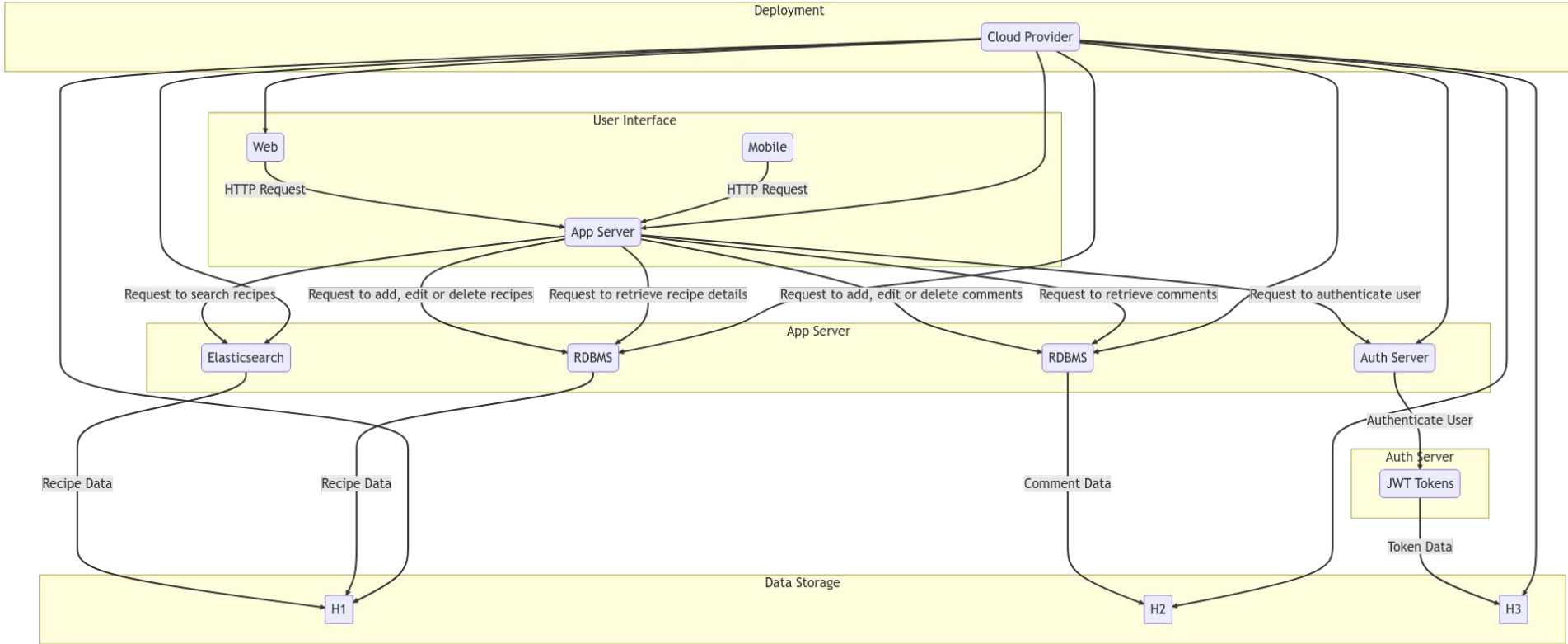
Diagram

Pan & Zoom



FULL SCREEN





Потерял контекст!



А что такое H1, H2, H3?



H1, H2, H3 - это заголовки первого, второго и третьего уровня соответственно. Это общепринятая нумерация для разных уровней заголовков в текстовом документе, включая документацию, веб-страницы и



А что такое H1, H2, H3 на диаграмме у тебя на диаграмме архитектуры?



Извините за путаницу, H1, H2 и H3 на диаграмме архитектуры не имеют никакого отношения к заголовкам. Это названия блоков, используемые для идентификации разных компонентов системы. Я использовал их в качестве меток для дальнейшего удобства описания функций и пользовательских историй.



Так а по смыслу это что? Они у тебя в блоке Data Storage



Да, верно, H1, H2 и H3 на диаграмме архитектуры относятся к блоку Data Storage. H1 - это блок базы данных, который содержит рецепты и информацию о пользователях. H2 - это блок хранения медиафайлов, в котором хранятся изображения, используемые в рецептах. H3 - это блок для хранения файлов локализации, в которых содержатся переводы на разные языки интерфейса приложения.



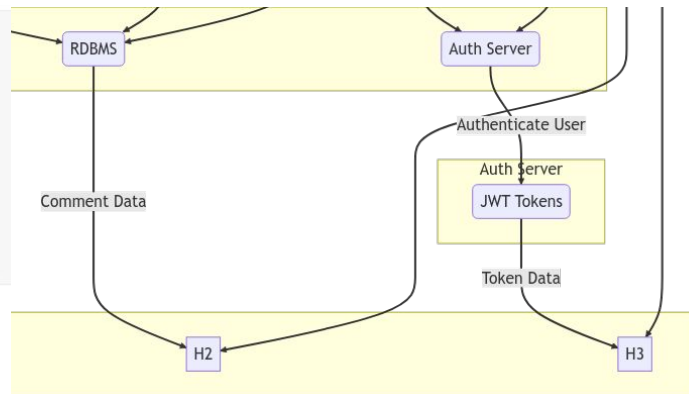


Так а по смыслу это что? Они у тебя в блоке Data Storage



Да, верно, H1, H2 и H3 на диаграмме архитектуры относятся к блоку Data Storage. H1 - это блок базы данных, который содержит рецепты и информацию о пользователях. H2 - это блок хранения медиафайлов, в котором хранятся изображения, используемые в рецептах. H3 - это блок для хранения файлов локализации, в которых содержатся переводы на разные языки интерфейса приложения.

Regenerate response

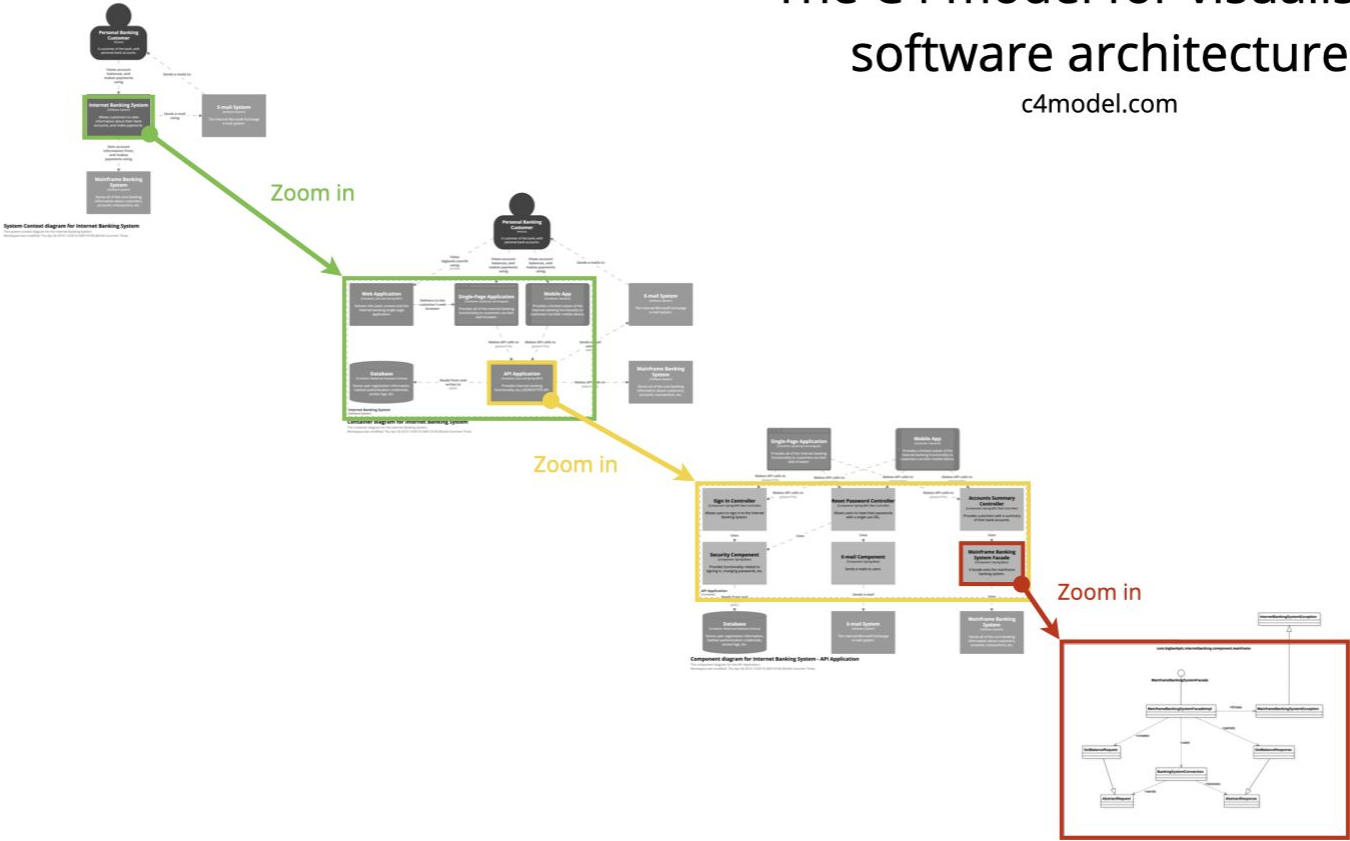


ChatGPT Mar 14 Version. Free Research Preview. Our goal is to make AI systems more natural and safe to interact with. Your feedback will help us improve.

Совсем всё перепутал с испугу

The C4 model for visualising software architecture

c4model.com



Level 1
Context

Level 2
Containers

Level 3
Components

Level 4
Code

Контекстная диаграмма: запрос по образцу



Давай попробуем ещё раз изобразить архитектуру системы рецептов. Вот пример контекстной диаграммы в синтаксисе mermaid:

```
C4Context
```

```
title System Context diagram for Internet Banking System
```

```
Enterprise_Boundary(b0, "BankBoundary0") {
```

```
    Person(customerA, "Banking Customer A", "A customer of the bank, with personal bank
```

Используй этот синтаксис и опиши в нём контекстную диаграмму для системы рецептов



Конечно, давай создадим контекстную диаграмму для системы рецептов в синтаксисе mermaid:



```
sql
```

Copy code

```
graph LR
```

```
    User[Пользователь]
```

Контекстная диаграмма: запрос по образцу



Отлично, давай теперь вспомним всю архитектуру, которую ты предлагал, и опишем её в этой нотации. Напоминаю архитектуру:

Фронтенд-клиенты: мобильное приложение и веб-приложение. Они взаимодействуют с сервером через API.

Бэкенд-сервер, который содержит бизнес-логику, взаимодействует с базой данных и обеспечивает API для фронтенд-клиентов.

База данных, которая хранит все данные, связанные с приложением, включая пользователей, рецепты, комментарии, изображения и т.д.

Хранилище файлов, которое используется для хранения изображений, связанных с рецептами.

Система авторизации и аутентификации, которая обеспечивает безопасность приложения и управление доступом к различным функциям приложения.



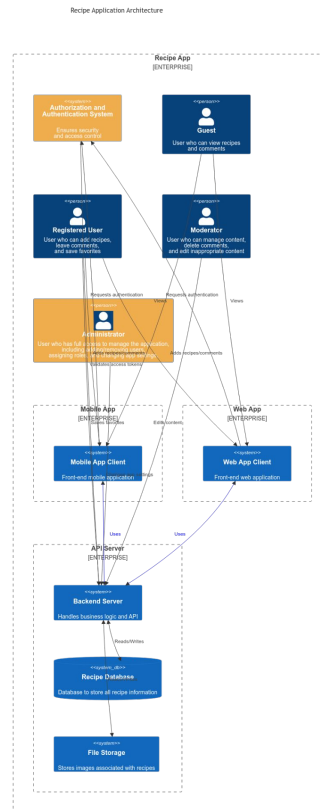
Конечно, давай я опишу всю архитектуру в нотации Mermaid:



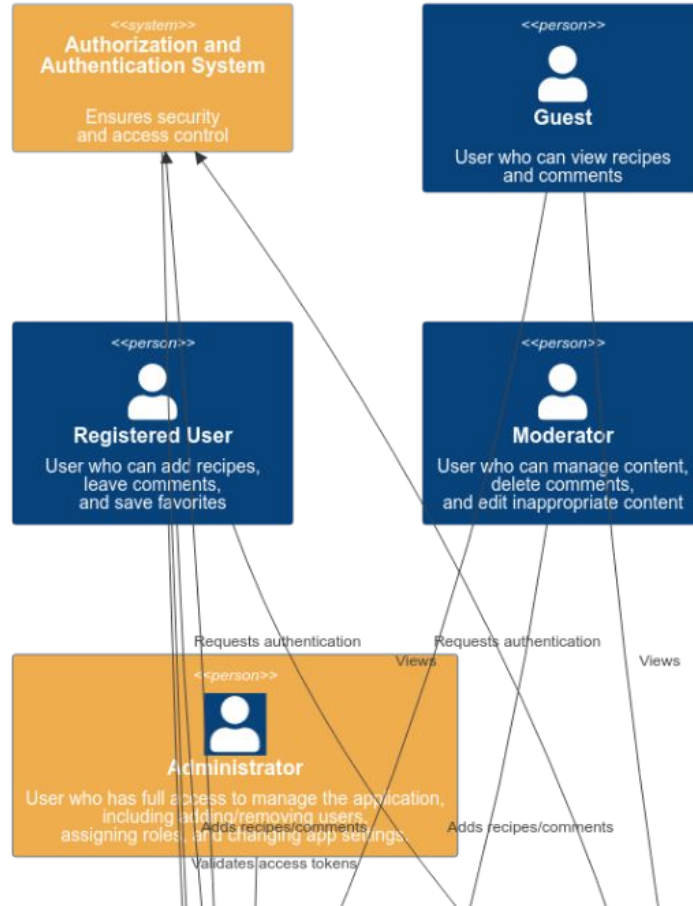
```
mermaid
C4Context
title Recipe Application Architecture
Enterprise_Boundary(b0, "Recipe App") {
  System_Boundary(b1, "Mobile App") {
    System(ClientMobile, "Mobile App Client", "Front-end mobile application")
  }
}
```

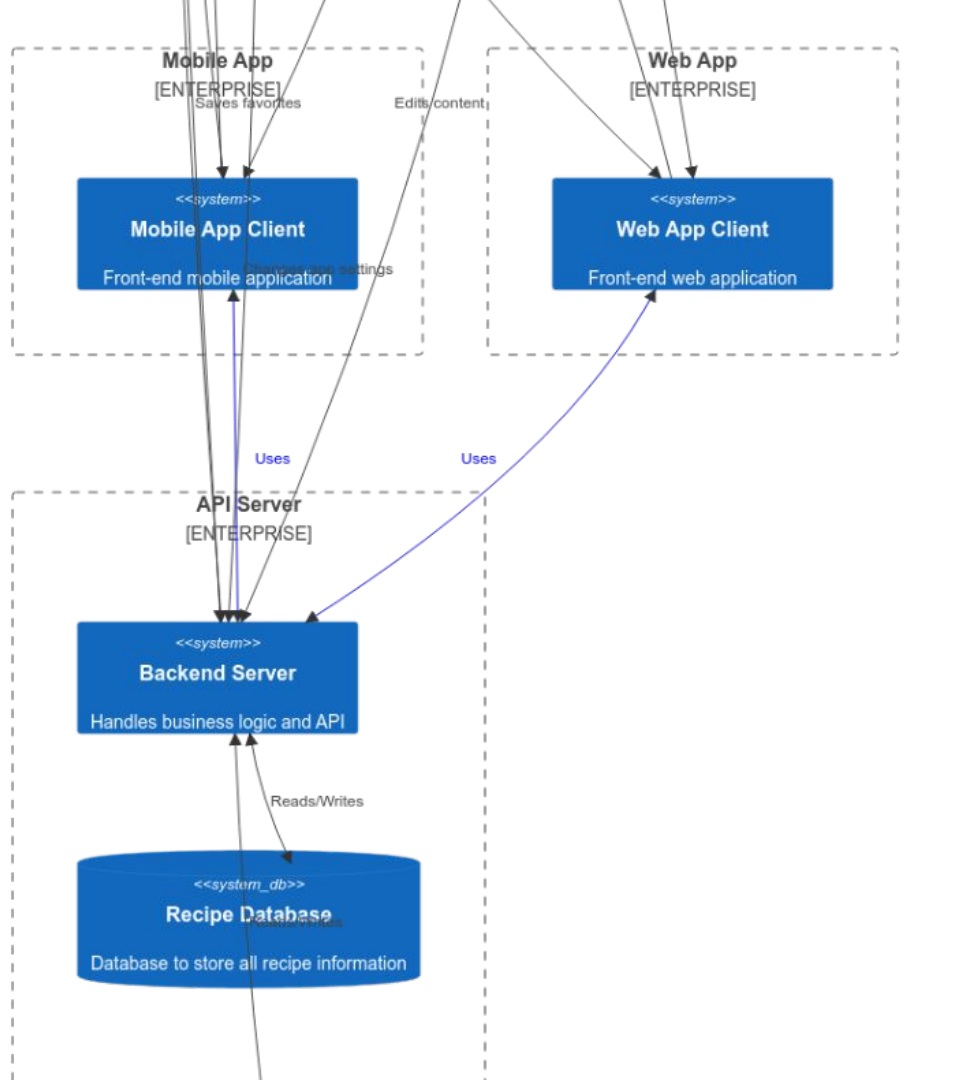
Regenerate response

Результат →

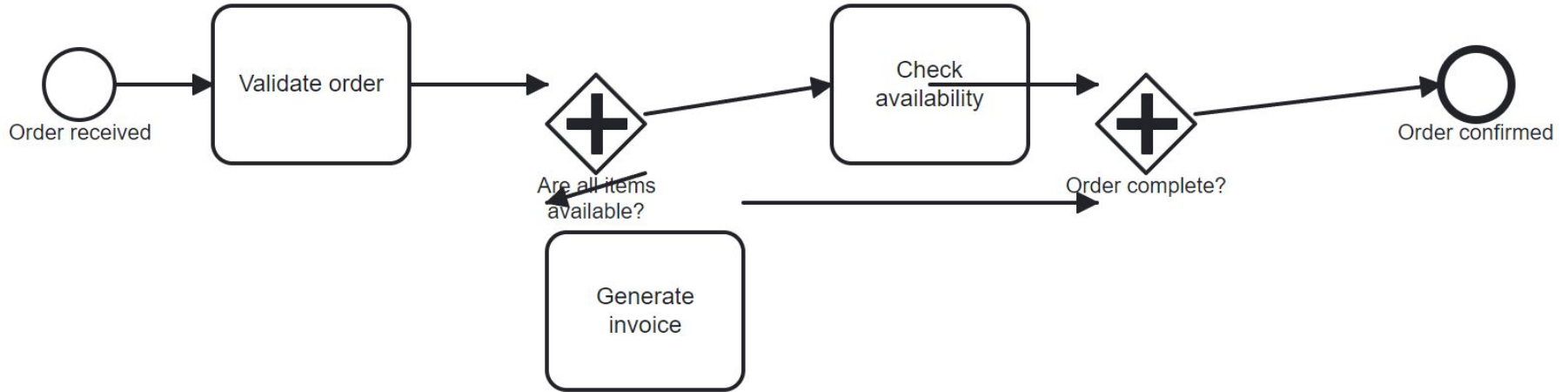


Recipe App
[ENTERPRISE]





Со стрелками на диаграмме беда, но бывает и хуже

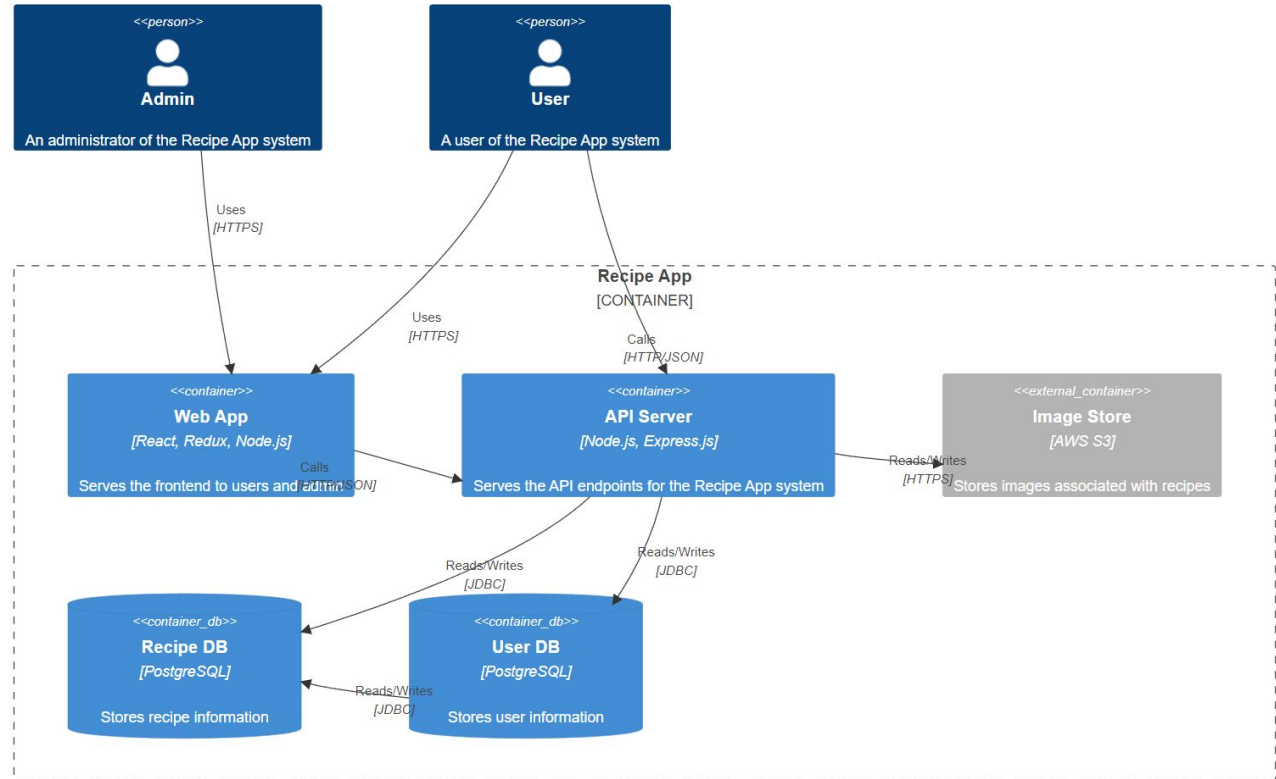


Контейнеры

Container diagram for Recipe App

Как сделано – цепочка запросов:

1. **восстановил контекст:**
OK, we start with following functions: <описание функций приложения>.
2. **спросил про архитектуру:**
Suggest high level architecture for this system. Give answer in text.
3. **показал образец: C4**
4. **попросил архитектуру по образцу**



Объяснение хода рассуждений



Why do you suggest to use node.js / express.js for API Server? Explain step by step your way of thinking



Sure, here are the steps I would consider when suggesting Node.js/Express.js for the Recipe App API server:

1. **Understanding the requirements:** As the first step, I would review the requirements of the Recipe App API server, such as the desired **functionality, performance, scalability, security, and maintainability**. Based on this, I would evaluate which programming languages and frameworks would be suitable for the project.
2. **Considering the project constraints:** In addition to the requirements, I would also take into account any project constraints such as **time, budget, team expertise, and existing technologies**. These factors can influence the choice of technologies and frameworks for the project.
3. **Assessing the strengths and weaknesses of Node.js/Express.js:** Node.js is a JavaScript runtime built on Chrome's V8 JavaScript engine, and it has become increasingly popular for building server-side applications. Express.js is a popular Node.js web framework that provides a lightweight and flexible architecture for building web applications. Some of the advantages of Node.js/Express.js include: [...]
4. Considering the requirements and constraints in light of the strengths and weaknesses of Node.js/Express.js: Based on the above analysis, I would consider the requirements of the Recipe App API server, such as the need for **high performance and scalability**. Since Node.js/Express.js are known for their performance and scalability, they would be a good fit for the project.
5. **Reviewing the team expertise:** If the **development team has experience with Node.js/Express.js**, it would be easier for them to develop and maintain the Recipe App API server. However, if the team has no prior experience with these technologies, there may be a learning curve involved, which could impact the project timeline and budget.

Объяснение хода рассуждений



3. **Assessing the strengths and weaknesses of Node.js/Express.js:** Node.js is a JavaScript runtime built on Chrome's V8 JavaScript engine, and it has become increasingly popular for building server-side applications. Express.js is a popular Node.js web framework that provides a lightweight and flexible architecture for building web applications. Some of the advantages of Node.js/Express.js include:
 - **High performance:** Node.js is known for its **speed and scalability**, and it can handle large volumes of data and requests efficiently.
 - **Large developer community:** Node.js/Express.js has a **large and active developer community**, which means there are many resources, libraries, and tools available for developers to use.
 - **JavaScript-based:** Node.js/Express.js are both JavaScript-based technologies, which makes it **easier for developers to transition from front-end to back-end** development.

Some of the potential drawbacks of Node.js/Express.js include:

- **Asynchronous programming model:** Node.js uses an **asynchronous, non-blocking I/O model**, which can make it more complex for developers to write code compared to traditional synchronous programming.
- **Scalability issues:** While Node.js is known for its scalability, it can be more **difficult to scale horizontally across multiple servers** compared to other languages such as Java or Go.

Спецификация API



Describe API endpoints on this server



Sure, here are some example API endpoints that could be implemented on the Node.js/Express.js server for the recipe app:

1. **POST /register** - Allows users to create a new account by providing their name, email, and password.
2. **POST /login** - Allows users to log in to their account by providing their email and password.
3. **GET /recipes** - Returns a list of all recipes in the database.
4. **POST /recipes** - Allows users to create a new recipe by providing its name, ingredients, directions, and an image.
5. **GET /recipes/:id** - Returns the details of a specific recipe identified by its ID.
6. **PUT /recipes/:id** - Allows users to update an existing recipe by providing its ID and the new information.
7. **DELETE /recipes/:id** - Allows users to delete an existing recipe identified by its ID.
8. **POST /recipes/:id/comments** - Allows users to add a comment to a recipe by providing its ID and the comment text.
9. **GET /recipes/:id/comments** - Returns a list of all comments for a specific recipe identified by its ID.
10. **POST /recipes/:id/favorite** - Allows users to add a recipe to their favorites list by providing its ID.
11. **DELETE /recipes/:id/favorite** - Allows users to remove a recipe from their favorites list by providing its ID.

Спецификация API: OpenAPI 3.0

yaml

Copy code


```
openapi: 3.0.0
info:
  title: Recipe App API
  version: 1.0.0
  description: API for managing recipes
servers:
  - url: http://localhost:3000/api/v1
paths:
  /recipes:
    get:
      summary: Get all recipes
      operationId: getAllRecipes
      responses:
        '200':
          description: OK
          content:
            application/json:
              schema:
                type: array
                items:
                  $ref: '#/components/schemas/Recipe'
```


Спецификация API: OpenAPI 3.0

Example: Create a new recipe

Request:

json

 Copy code

```
POST /api/recipes HTTP/1.1
Content-Type: application/json
Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxIiwiaWF0Ijox

{
  "title": "Spaghetti Carbonara",
  "description": "Delicious pasta with bacon, eggs, and cheese",
  "ingredients": [
    "400g spaghetti",
    "200g bacon",
    "4 eggs",
    "100g parmesan cheese"
  ],
  "instructions": [
    "Cook spaghetti according to package directions.",
    "Fry bacon in a pan until crispy.",
    "Beat eggs in a bowl and mix in grated parmesan cheese.",
    "Drain spaghetti and add it to the bacon in the pan.",
    "Remove pan from heat and add the egg mixture, stirring quickly until eggs are",
    "Serve hot and enjoy!"
  ]
}
```

Диаграмма последовательности

Add New Recipe Sequence Diagram

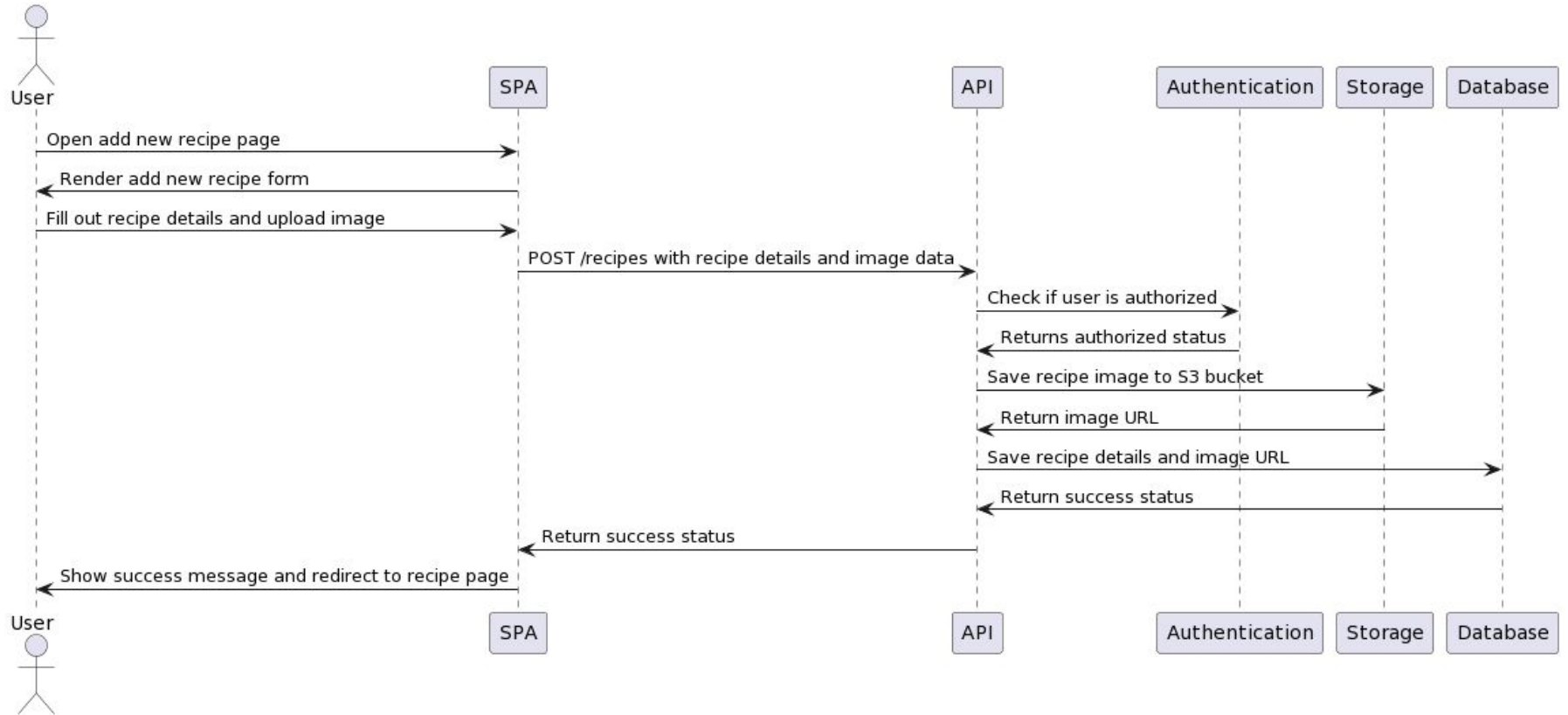


Диаграмма классов (что бывает, если не указать формат вывода)



Provide a class diagram with attributes



Sure, here is a class diagram with attributes for the receipt app system:

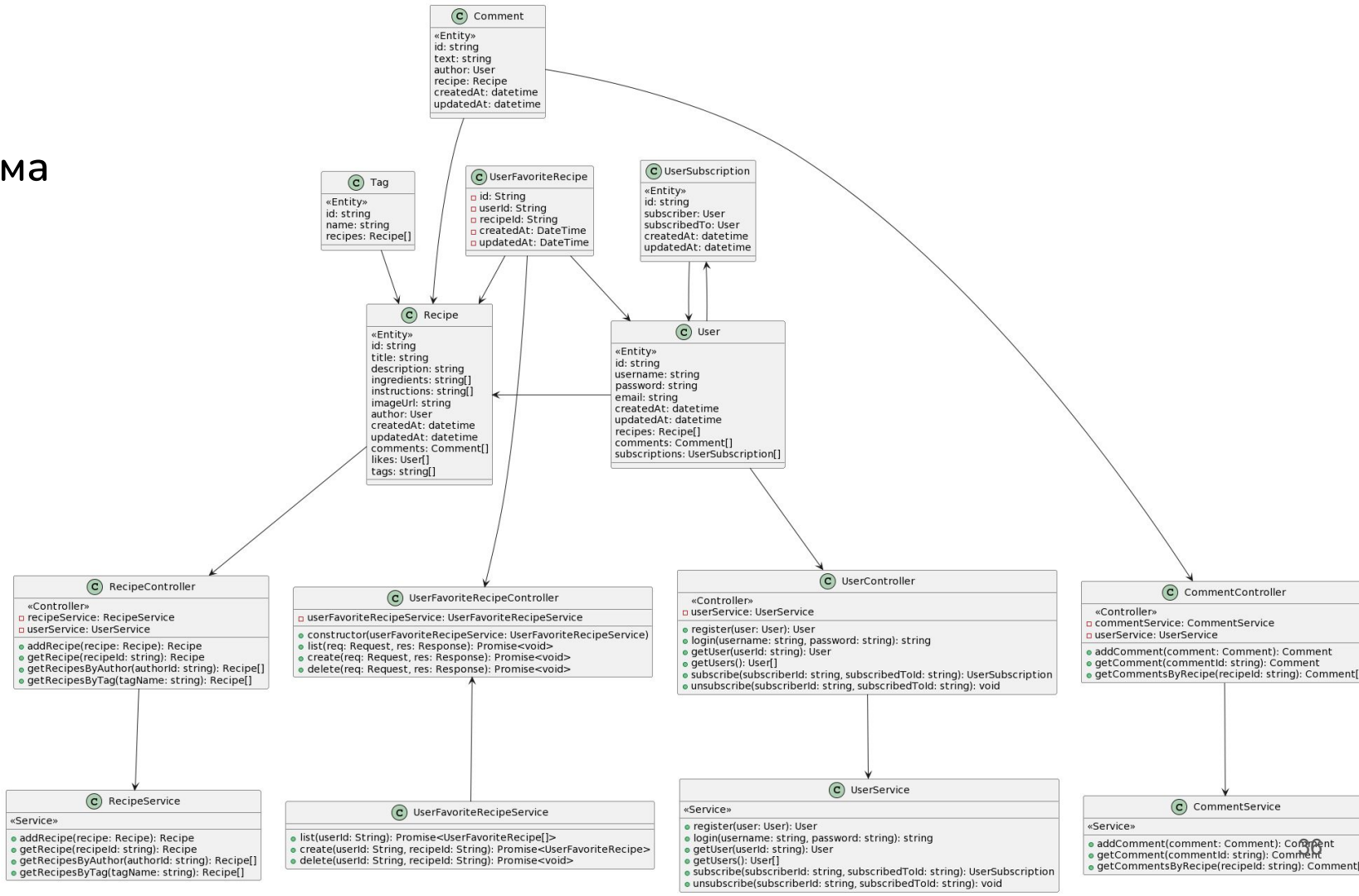


```
php Copy code
+-----+
|           User           |
+-----+
| -id: int                 |
| -name: string            |
| -email: string           |
| -password: string        |
+-----+
| +register(name: string,  |
|   password: string):    |
|   bool                  |
| +authenticate(email:    |
|   string, password:    |
|   string): bool         |
| +update_profile(name:  |
|   string, email:       |
|   string, password:    |
|   string): bool         |
| +delete_profile():      |
|   bool                  |
+-----+

+-----+
|           Recipe         |
+-----+
| -id: int                 |
| -name: string            |
| -description: string     |
```

Код:

Диаграмма классов



Код:

Диаграмма классов

Не всё так просто!



go on



I apologize, but it seems like there's an issue with the rendering of the class diagram. Could you please provide me with the text representation of the diagram so that I can double-check it?



OK, let's do it step by step. Start with classes, that representing domain objects and users.



Sure, here's the PlantUML code for the domain objects and user classes:



yaml

Copy code

```
@startuml
class User {
  <<Entity>>
  id: String
  name: String
  email: String
  password: String
  created: LocalDateTime
  updated: LocalDateTime
}
```

Regenerate response

Код: node.js



Give me code of RecipeController for node.js



Sure! Here is an example of a RecipeController in Node.js using Express:



scss

Copy code

```
const express = require('express');
const router = express.Router();
const recipeService = require('../services/recipeService');

// GET /recipes
router.get('/recipes', async (req, res) => {
  try {
    const recipes = await recipeService.getRecipes();
    res.status(200).json(recipes);
  } catch (error) {
    console.error(error);
    res.status(500).json({ message: 'Internal server error' });
  }
});

// GET /recipes/:id
router.get('/recipes/:id', async (req, res) => {
  const id = req.params.id;
  try {
    const recipe = await recipeService.getRecipeById(id);
    if (!recipe) {
```

Код: Unit-тесты



Can you write an example of unit test for recipes app backend



Sure, here's an example of a unit test for the `RecipeService` class in the recipes app backend:

```
scss Copy code

// Import the necessary modules for testing
const { expect } = require('chai');
const sinon = require('sinon');

// Import the RecipeService class
const RecipeService = require('./RecipeService');

// Define the test suite for the RecipeService class
describe('RecipeService', () => {
  // Define a test case for the getRecipes method
  describe('getRecipes', () => {
    it('should return an array of recipes', async () => {
      // Create a mock database client
      const dbClient = {
        query: sinon.stub().resolves({ rows: [{ id: 1, name: 'Spaghetti Carbonara' }] });
      };

      // Create a new RecipeService instance with the mock database client
      const recipeService = new RecipeService(dbClient);
      // Call the getRecipes method and store the result
```

```
    });
  });

  // Define a test case for the getRecipeById method
  describe('getRecipeById', () => {
    it('should return a recipe object with the specified ID', async () => {
      // Create a mock database client
      const dbClient = {
        query: sinon.stub().resolves({ rows: [{ id: 1, name: 'Spaghetti Carbonara' }] });
      };

      // Create a new RecipeService instance with the mock database client
      const recipeService = new RecipeService(dbClient);
      // Call the getRecipeById method with a test ID and store the result
      const result = await recipeService.getRecipeById(1);
      // Assert that the result is the expected recipe object
      expect(result).to.deep.equal({ id: 1, name: 'Spaghetti Carbonara' });
      // Assert that the database client's query method was called with the expected query
      expect(dbClient.query.calledWith('SELECT * FROM recipes WHERE id = $1', [1])).to.be.true;
    });
  });
});
```

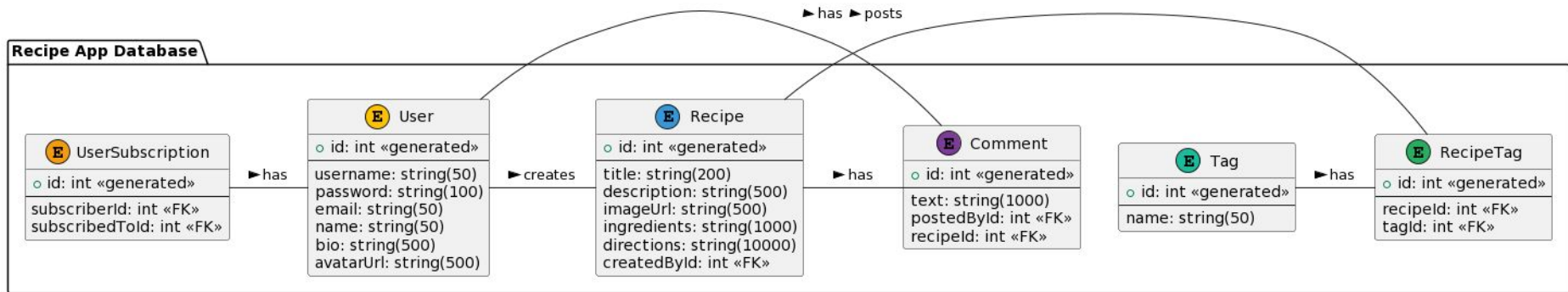
Код: структура БД и SQL



You have mentioned DB table 'recipes'. Describe all the tables of the db



Sure, based on our previous discussions, here's an example of the database schema for our Recipe app:



Write an SQL query which select all recipes that has been commented of favorite in the last week



The SQL query to select all recipes that have been commented or favorited in the last week can be written as:

```
sql Copy code
SELECT DISTINCT r.id, r.title, r.description
FROM recipes r
LEFT JOIN comments c ON r.id = c.recipe_id
LEFT JOIN favorites f ON r.id = f.recipe_id
WHERE (c.created_at >= NOW() - INTERVAL 1 WEEK OR f.created_at >= NOW() - INTERVAL 1 WEEK)
```


Макеты интерфейса и верстка

```
html Copy code

<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Recipe Details</title>

    <!-- Material UI CSS -->
    <link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Roboto:300,400,500,700,900">
    <link rel="stylesheet" href="https://fonts.googleapis.com/icon?family=Material+Icons">
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/materialize-css@1.0.0/css/materialize.min.css">

    <!-- Custom CSS -->
    <style>
      /* Add your custom styles here */
    </style>
  </head>

  <body>
    <div class="container">
      <h1>Recipe Details</h1>
      <div class="row">
        <div class="col s12">
          <div class="card">
            <div class="card-image">
              
            </div>
            <span class="card-title">Recipe Name</span>
          </div>
          <div class="card-content">
            <p>Recipe description goes here.</p>
            <table>
              <thead>
                <tr>
                  <th>Ingredients:</th>
                  <th>Instructions:</th>
                </tr>
              </thead>
              <tbody>
                <tr>
                  <td>Ingredient 1</td>
                  <td>1. Step 1</td>
                </tr>
                <tr>
                  <td>Ingredient 2</td>
                  <td>2. Step 2</td>
                </tr>
                <tr>
                  <td>Ingredient 3</td>
                  <td>3. Step 3</td>
                </tr>
              </tbody>
            </table>
            <div class="card-action">
              <h3>Comments:</h3>
              <table>
                <thead>
                  <tr>
                    <th>User Name</th>
                    <th>Comment text goes here.</th>
                </tr>
              </thead>
              <tbody>
                <tr>
                  <td><input type="text" value="User Name" />
                </td>
                <td><input type="text" value="Comment text goes here." />
                </td>
              </tbody>
            </div>
          </div>
        </div>
      </div>
    </div>
  </body>
</html>
```



🔄 Regenerate response

Управление проектом: сбор команды



So, put the whole team members in a table, and add approximate number of people we need at each role



Sure, here's an example table outlining the team members we might need for this project:



Role	Responsibilities	Number of people
Project Manager	Oversees the entire project and ensures that it is delivered on time, within budget, and meets the requirements.	1
Backend Developer	Develops the API server and database for the application.	2-3
Frontend Developer	Develops the web and mobile applications for users.	2-3
UI/UX Designer	Designs the user interface and user experience for the web and mobile applications.	1-2
Quality Assurance	Tests the application to ensure that it meets the requirements and is free of bugs.	1-2
DevOps Engineer	Sets up and maintains the infrastructure for the application, including servers and deployment pipelines.	1-2

Keep in mind that the number of people needed for each role may vary depending on the size and complexity of the project, as well as the timeline and budget.



Add technology stack

Управление проектом: сбор команды с нужными компетенциями



Sure, here's a proposed technology stack based on our previous discussions:



Role	Responsibilities	Recommended Skills	Number of People
Project Manager	Oversees the project and ensures it meets	Project management	1
	the client's requirements		
UX/UI Designer	Designs the user interface and user experience	UX/UI design, Adobe Creative Suite, Figma, Sketch	1-2
Front-end Developer	Implements the user interface and	HTML, CSS, JavaScript, React,	2-3
	user experience designed by the UX/UI	Angular, Vue.js, responsive	
	designer	design, cross-browser compatibility	
Back-end Developer	Implements the server-side logic and API	Node.js, Express.js, MongoDB,	2-3
	endpoints	SQL, API design, authentication	
DevOps Engineer	Sets up the development and deployment	Linux, AWS, Docker,	1-2
	infrastructure	Regenerate response tes, CI/CD	

Управление проектом: календарный план



Suppose we are starting at 2023 April 10, generate a Gantt diagram code for mermaid



Recipe App Development Schedule



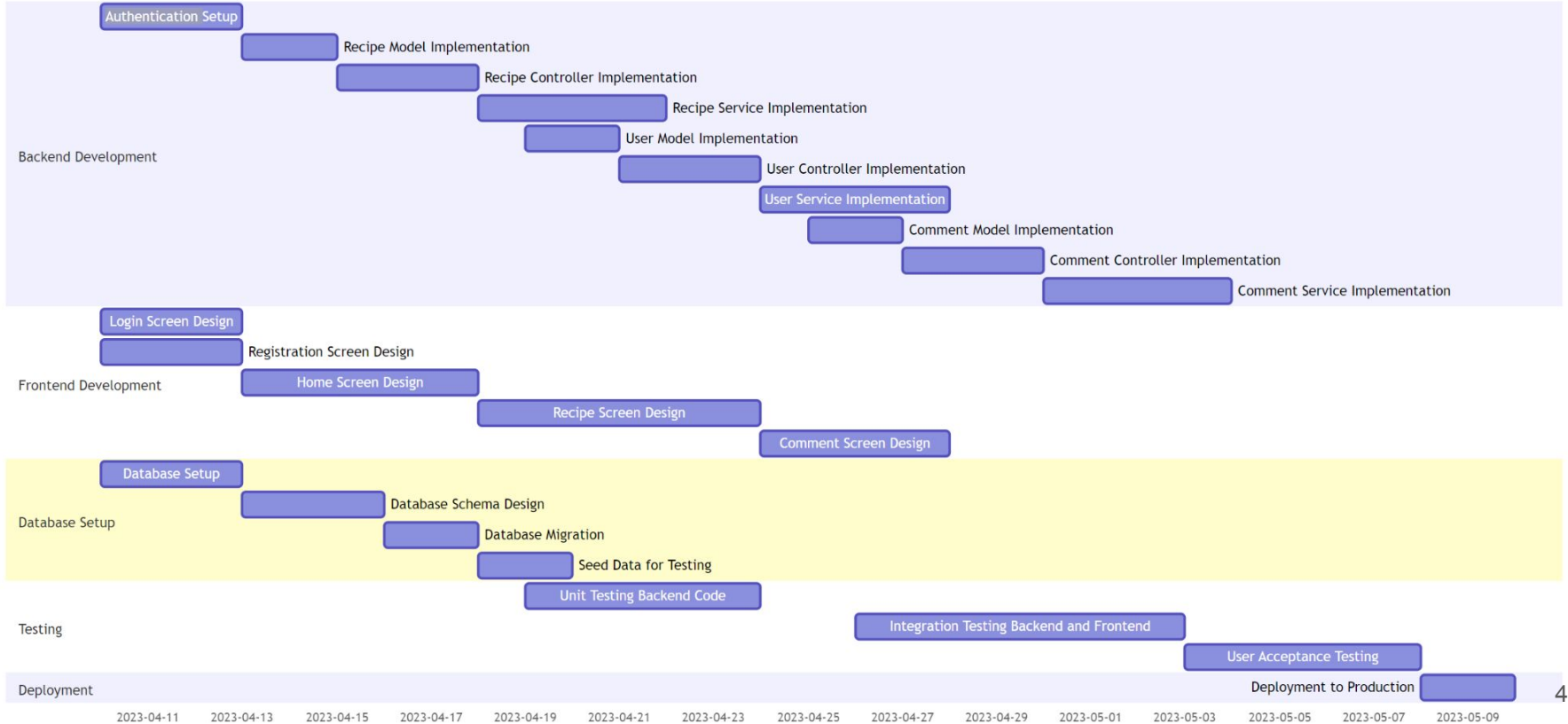
Note that the dates and duration of the tasks are just estimations and may vary depending on various factors such as team size, experience, and unexpected issues.



Can we put some works in parallel?



Yes, we can put some works in parallel to optimize the timeline. However, it's important to ensure that dependencies are accounted for and that parallel work doesn't negatively impact the quality of the project.



ChatGPT – старательный джун.

Умеет неплохо кодить

Знает библиотеки

Понимает контекст проекта: бизнес, требования, архитектуру, моделирование данных, UI, управление проектом.

Скруплезный, вспоминает вещи, про которые люди часто забывают. Помогает задавать правильные вопросы.

Отвлекается и забывает часть разговора, теряет контекст.

То одно сделает хорошо, то другое, а совместить не может.

ChatGPT: как применять

- Для проработки требований в начале проекта, в том числе – чтобы задать правильные вопросы.
- Как генератор идей – ещё одного “актера” в команде на мозговых штурмах.
- Для “раскрытия объема”: по короткой вводной придумать всё остальное.
- Для переоформления (“оформи это API по спецификации OpenAPI”)
- Для перевода естественного языка в программный код
- Для создания заготовок кода, требований и т.п.



Спасибо за внимание! Можешь задать мне какие-нибудь вопросы по докладу?



Вас развлекал **Юрий Куприянов**.

Читайте про эксперименты с ChatGPT, а ещё про системный анализ и проектирование в моём канале “Системный сдвиг”: t.me/systemswing