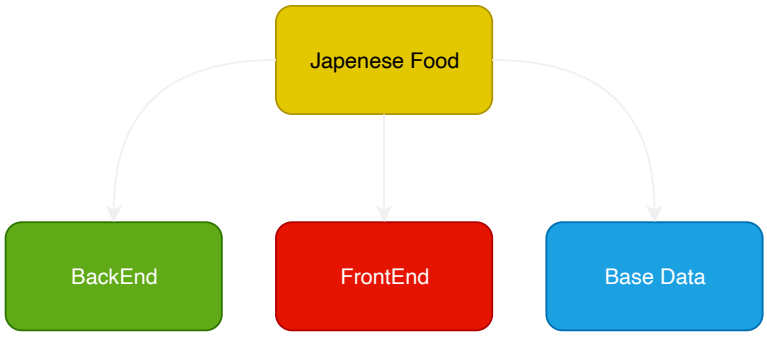
Контрольная точка 1

Структура проекта

Проект будет делиться на 3 большие части это **Backend**, **Frontend** и **База данных**.



Рассмотрев все возможные и интересные вариации исполнения подобных проектов, мною были выбранные определенные технологии для реализации всех проессов.

Backend

За однову такого большого проекта был взят Python по причине простого написания кода, и безпроблемного дальнейшего обслуживания, а так же маштабирования. На базе его очен просто пишутся асинхронные функции, а так же обрабатываеются потоки. Так же большим плюсом Python является то, что у него большое коммьюнити и огромное кол-во библиотек. Основные перечислю:

- FastAPI
- sqlalchemy
- ormar
- requests
- pydantic

FrontEnd

Так как наше приложение подрузомевает под собой многопользовательный режим, с большим трафиков, а так же совершенно разные пользовательские устройства, мою был выбран вариант статики, а именно собирать весь FrontEnd на JavaScript. На самом деле тут все не обназначно, был вариант использования фреймворка Flask который смог бы заменить нам отдельный веб сервер и FastAPI, но если мы говорим о стабильности и большой производительности, то было решенр разделить на корню BackEnd и FrontEnd.

Наш FrontEnd будет строиться так же на популярных фреймфорках. За основу был взят vueJS 3 все так же из-за его простоты и не плохих возмностях, за сборку проекта будет отвечать Vite с модулем Cypress для автоматического шифрования нашего кода и повышения безопастности.
Сам дизаин макет будет написан на BootStrap, с использованием небольшого кол-ва мелких open-source фреймвороков которые могут пригодиться во время напсиания кода.

Base Data

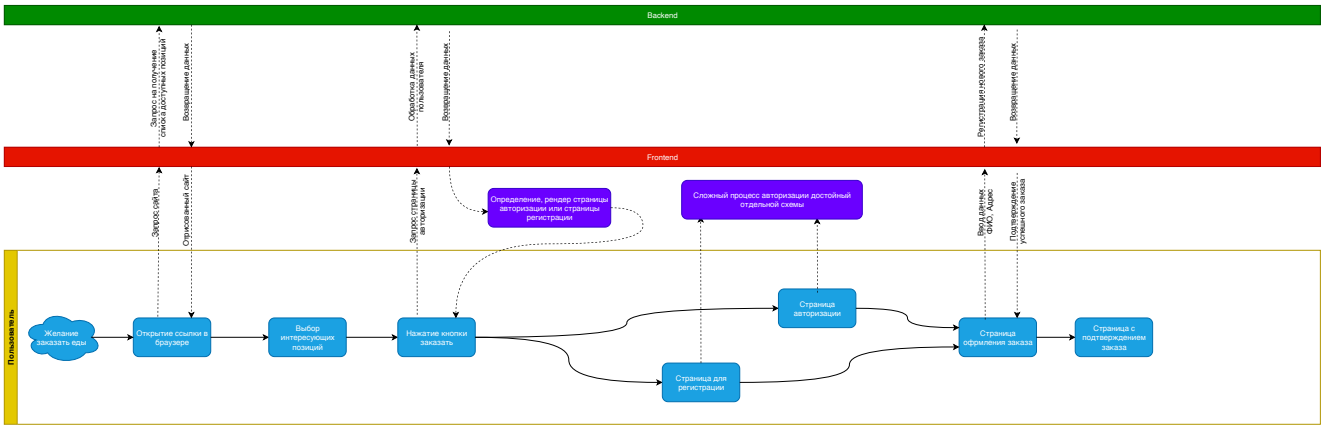
В роли базы данных было решено взять Postgres, за его гибкость в настройке, множественных встроенных функций, а так же набирающую популярность последние годы.

Когда все переменных всали на свои места и разработка началась, стали понятны некоторые проблемы, а именно расхождения в версиях многих комонетов, а так же неудобство в тестировании всех частей приложения. Эти пробелы было решено решить упаковкой всего проекта в Docker, посредством Docker-Compose. Это сильно упращает разработку разнородного кода, и тестирования работоспособности в целом, а так же в плюсы можно отнести улудшеную моштабируемость и ускоренный процес разворачивания нового инстанса.

Бизнес сценарии

После того как я обрисовала всю концепцию своего проекта, можно уже представлять схему взаимодействия пользователя и всех частей приложения. Рассмотрим один не самый простой процесс, но как мне кажется самый популярный в будующем и один из самый маштабных в проекте.
Немного пояснений, для более простого понимания на схеме не указана база данных по причине того что общение с ней происходит посредством библиотеки SQLAlchemy, с преминением технологии ORM.

Процесс создания нового заказа



Таким образом на схеме мы видим все плюсы нашего подхода к проектированию такого типа проектов, каждая наша часть по своему изолирована, доступа к базе данных вне контура, отсутствует от слова совсем, пользователь видит всего статичный сайт, который будет рабоать(Частично) при проблемах на нашем Backend'e, а так же Backend который может обеспечивать одновременную работу нескольких инстасов Frontend, а так же возможно мобильного приложения, терминала или чего-то другого в будующем.