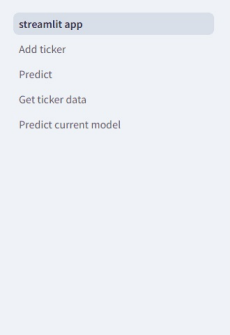
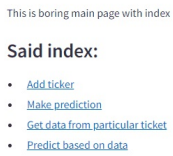
С технической точки зрения, проект на текущем этапе состоит из двух основных частей:

1. **Бекэнд (FastAPI)**
   Основные фичи:
   - Подробная документация используемых в API схем
   - Реализованы:
     - Логирование в файл (`RotatingFileHandler`) и консоль (`StreamHandler`)
     - Возможность получения списка доступных для анализа тикеров [`/api/tickers`]
     - Добавление новых тикеров (пока что на уровне списка доступных для анализа, добавление самих данных находится в разработке), а также удаление имеющихся
     - Возможность просмотра исторических данных тикеров [`/api/tickers/{ticker}/history`]
     - Выбор типа модели [`/api/model/select`]
     - Создание моделей [`/api/model/select`] и построение прогнозов [`/api/predict`] - пока что доступна одна модель - `auto_arima_60`
   - В разработке:
     - Упомянутое добавление данных для новых тикеров
     - Добавление функционала использования других моделей
     - Серверная составляющая, хранение данных в БД (на текущем этапе используется готовый csv файл)
   - В планах:
     - Добавление возможности изменения гиперпараметров различных моделей (на текущем этапе предоставление пользователю возможностей по их редактированию ограничены, т.к модель, которая реализована, принимает 7 различных параметров и это совсем не user-friendly)
       
2. **Фронтэнд (Streamlit)**
   - Реализовано четыре вкладки:
     - Add_ticker -- создание нового тикера со сгенерированными данными
     - Predict -- Предсказание на будущее для существующего тикера
     - Get_ticker_data -- получение данных и eda по существующеему тикеру (диаграммы на данной вкладке отрисовываются при помощи plotly)
     - Predict_current_model -- предсказание на основе предоставленных данных
   - Дальнейшее развитие фронта будет идти по мере развития бэкенда (по мере появления новых функций будет делаться графический интерфейс к ним).





Обе части обернуты в `Docker` контейнеры. Их совместная работа реализована посредствами `docker-compose`.
Со стороны пользователя приложения запросы к серверу поступают по открытому порту 8000. Само приложение развернуто на порте 8501.
К портам есть доступ и у хоста, поскольку они пробрасываются на `localhost`. Это сделано для возможности проверки отдельных запросов с помощью `swagger`.

Таким образом для запуска требуется в корневом каталоге запустить команду:

```
docker-compose up
```