**Документация к проекту DMMT**

**Машинное обучение**

Машинное обучение в проекте реализовано при помощи catboost. Модель обучена на датасете, который был взят с сайта Kaggle, CatBoostRegressor’ом. Она высчитывает предполагаемую стоимость квартиры исходя из введенных пользователем параметров.

**Хранение картинок**

Хранение картинок на сайте реализовано при помощи minIO. Файл docker-compose.yml создает это хранилище, затем в файле s3.py (импортируется переменная settings из файла settings.py, которая передается туда) инициализируется переменная s3, которая может генерировать ссылки на изображение в minIO (В bucket’е “images” хранятся все изображения). Обращаясь к этим ссылкам и выводятся картинки.

**Роуты**

Back-end часть реализована на Flask.

***/collection***

В этом роуте пользователь может посмотреть существующие квартиры по параметрам (фильтрам), которые он введет. Это работает через метод request.form.get(). Пользователь может не заполнять все поля. Это сделано через присвоение нужных значений в случае незаполнения (например, незаполненная макс. площадь становится равной максимальной площади в датасете). Дальше формируется отфильтрованный датасет. Также создается список индексов картинок, которые должны присвоиться.

***/cost\_calculation***

В этом роуте пользователь вводит параметры, и по этим параметрам высчитывается предполагаемая стоимость такой квартиры. С помощью request.form.get() передаются параметры, из них создается pd.Series(), который подается в модель. Модель же подгружает загруженные веса и формирует на основе параметров стоимость.

***/about***

Страница about с авторами проекта и ссылкой на тг-бота для связи с разработчиками (например, в случае багов).

***/index***

Это главная страница сайта. На ней реализована форма заполнения заявки для квартиры

***/registration***

Это страница с регистрацией пользователя на сайте. Здесь пользователь вводит свои данные в форму, после сервер обрабатывает их и добавляется в таблицу users в БД. Также в этом роуте осуществлена проверка данных, которые вводятся: нельзя оставить поля пустыми, проверяется email, которые вводится, чтобы не было совпадений в таблице БД, а также для безопасности вводимый пароль хэшируется и также добавляется в БД.

***/login***

Это страница входа на сайт. Здесь реализовано то же самое, что и на странице /registration, но при этом данные не добавляются в БД, а взаимодействуют с ней, проверяя почту на совпадения в БД и правильность ввода пароля.

***/logout***

Функция выхода пользователя из аккаунта. Данная страница открывается только в случае, если человек зарегистрирован, это реализовано при помощи декоратора @login\_required

***/per\_acc***

Страница личного аккаунта. Данная страница открывается только в случае, если человек зарегистрирован, это реализовано при помощи декоратора @login\_required

***/favourites***

Страница избранных квартир. На данной странице можно увидеть квартиры, которые пользователь добавил в избранное. Здесь берутся данные (user\_id) из таблицы в БД и все ID квартир, которые были добавлены ранее. После уже шаблонами Jinja2 данные обрабатываются и выводятся на страницу. Данная страница открывается только в случае, если человек зарегистрирован, это реализовано при помощи декоратора @login\_required

***/redirect\_to\_signin***

Обработка ошибки, если человек не зарегистрирован.

**База данных**

В проекте БД реализована при помощи питоновского модуля flask\_sqlalchemy. Было создано 4 таблицы в БД, которые активно используются в проекте.

***Table: users***

В данной таблице создано 4 колонки. Суть данной таблицы в том, чтобы хранить данные о зарегистрированных пользователя.

1) id – int, primary\_key=True

2) name – str(100), nullable=False, unique=True – имя пользователя

3) email - str(100), nullable=False – почта пользователя

4) password - str(100), nullable=False – пароль пользователя

***Table: bot\_req***

В данной таблице 3 колонки. Суть данной таблицы в том, чтобы сохранять информацию о пользователе, которые захотел оставить заявку на квартиру в роуте /index

1) id – int, primary\_key=True

2) name – str(100), nullable=False, unique=True – имя пользователя

3) tg - str(100), nullable=False – телеграм-ник пользователя

***Table: bag\_bot***

В данной таблице создано 4 колонки. Суть данной таблицы в том, чтобы хранить информацию, которую пользователь пишет в бота по поводу багов.

1) id – int, primary\_key=True

2) name – str(100), nullable=False, unique=True – имя пользователя

3) tg - str(100), nullable=False, unique=True – телеграм-ник пользователя

4) bag - str(1000), nullable=False – сообщение о баге до 1000 символов

***Table: favorites***

В данной таблице создано 3 колонки. Суть данной таблицы в том, чтобы хранить информацию о том, какие квартиры пользователь добавил в избранное.

1) id – int, primary\_key=True

2) user\_id– int, nullable=True, unique=False – id пользователя, который добавил квартиру в избранное

3) product\_id - int, nullable=True, unique= False – id квартиры, которая была добавлена в избранное