«Национальный исследовательский университет

«Высшая школа экономики»

Лицей

Индивидуальная выпускная работа

ИТ-ПРОЕКТ: WEB-приложение — футбольный трекер

Unified By Football

https://github.com/goltsmannn/unified_by_football

Выполнил: Гольцман Марк Михайлович

Группа: 11ЭЗ

Консультант:

Палеев Даниил Алексеевич

Москва – 2023

1) Опыт в разработке, предметная область продукта

Я занимался промышленной разработкой на языке Python 1 год в Московской Школе Программистов, кратко проходил основные библиотеки, но в качестве web-фреймворка на занятиях использовал FastAPI. У меня есть опыт создания сайтов на HTML+CSS+JS без серверной логики (JS использовался для анимаций, прелоадеров, и т.д.), написания PET-проекта на Flask+MySQL, который представлял из себя исключительно страницы для запросов и отображения данных из БД. Я хорошо ориентируюсь в чистом C++, занимался олимпиадным программированием и машинным обучением на Python. Опыта с Django, DRF, React до этого не было.

Главная тема проекта — футбол. Я играю в него с детства, занимался профессионально, сейчас на любительском уровне. Слежу за мировой футбольной ареной, болею за московскую команду. Проблема, которую решает проект, коснулась меня, что привело к его созданию.

2) Проблемное поле

Проблема отсутствия информации о футбольных полях стоит достаточно остро. Во-первых, не существует ни одного агрегатора информации, который предоставляет информацию о некоммерческих площадках — на специализированных сайтах пользователь может найти только платные манежи/поля. Google, Yandex (и другие создатели городских карт) не выделяют как локации на карте дворовые поля, зачастую помечая их “спортплощадка”, или не отмечают их совсем. К большинству существующих площадок не прикреплены фотографии и не поддерживается актуальная информация об их состоянии. Ориентироваться приходится по 3D карте, что достаточно долго и возможно только в случае уже имеющейся информации, где нужно искать поле. В общем, информации либо нет, либо ее недостаточно, либо выборка доступных площадок крайне ограничена.

Я не один раз становился заложником этой проблемы, когда искал в центре Москвы площадку для игры с командой (поля, к которым мы приходили были закрыты на ремонт/переполнены трижды). Более того, я играл на маленьких хоккейных площадках 10 лет, и лишь недавно обнаружил в глубине районных дворов футбольное полноразмерное поле, на котором мог бы играть все эти годы; на картах его найти проблематично даже зная,

где оно находится. Это окончательно натолкнуло меня на решение проблемы с помощью данного проекта.

3) Образ продукта

Во-первых, в проекте реализована функция предложения полей, т.е. пользователи смогут вносить свой вклад в проект, прикладывая знакомые им площадки, а администрации останется посетить отмеченное место и проверить, есть ли там поле на самом деле и верифицировать/отклонить метку. Помимо названия и локации, информация о поле включает в себя фотографию, часы работы и подробное описание, в которое будет включена информация о покрытии и т.д. Таким образом, решится проблема отсутствия информации в проблемной области.

Отзывы к площадке позволяют поддерживать более актуальную информацию о ней, к прим., если будут встречаться упоминания о том, что поле закрыто, оно может быть временно скрыто из списка администрацией. Пользователи смогут указывать, занимается ли данное поле жильцам для своих неспортивных целей (зачастую на площадках гуляют с детьми, что может доставлять лишние проблемы). Пользователи могут жаловаться на недостоверные отзывы и систематически выкладывающие ложную информацию будут заблокированы на проекте. Все это невозможно в больших проектах, таких как Google, ведь модераторы и пользователи не могут удивлять столь большое внимание мелочам, следовательно качество информации значительно падает.

Пользователи могут указать подробную информацию о себе в специализированных настройках профиля, актуальную для подбора игроков в команду (рост, вес, и т.д.). Функция активности на поле позволит оценить его занятость, а заполненные профили понять, возможно ли там будет составить конкуренцию другим игрокам в одиночку или с командой.

Более того, на сайте реализована внутренняя почта, которая позволяет писать сообщения пользователям, на которые оформлена подписка. Игроки могут списываться, обмениваться соцсетями или договариваться об игре оставаясь анонимными. Игроки могут добавлять друг друга в черные списки, чтобы не получать сообщения. Кроме того, почта используется автоматическими сервисами проекта для оповещения, когда кто-то играет на поле, которое добавлено пользователем в избранное. В будущем будет реализована функция оповещения активности друзей.

Для более быстрой ориентации на сайте сделан список избранных полей.

4) Сделанный продукт

Был реализован сценарий регистрации. Пользователь должен выбрать уникальный никнейм и ввести валидную почту, подтвердить пароль, после чего ему будет отправлена ссылка для активации на почту.

Добавление в друзья. Пользователи могут подписываться друг на друга, чтобы потом иметь возможность писать сообщения, или блокировать, чтобы не получать сообщения. Поиск пользователей происходит в специальном окне по введенному никнейму.

Почта. Пользователь может переключаться между полученными и отправленными сообщениями, кликать на блок сообщения, чтобы прочитать его (в списке сообщений видны только никнеймы и времена отправки). Пользователь может написать сообщение любому аккаунту из своих подписок, а также удалить полученное сообщение.

Карта от сервиса Яндекс. Пользователь может перемещаться по карте, искать, и переходить на страницу деталей о поле, которое его интересует. Страница площадки показывает активность на ней, подтвержденную администратором информацию и отзывы других юзеров.

Добавление площадки. Пользователь открывает специальную страницу с пустым полем, отмечает местоположение площадки и вводит необходимые дополнительные данные в всплывающем окне. После этого площадка отправляется на подтверждение администратору, и после валидации она будет отображена на главной странице.

Активность. Пользователь может отметить активность на поле, после чего эта функция ему будет недоступна до завершения указанного промежутка времени. Активность пользователя отображается у него в профиле, на поле, на котором она была отмечена, а также может быть удалена или скрыта.

Пользователь может удалить аккаунт, поделиться ссылкой на профиль с активностью, смотреть статистику своих посещений, менять настройки приватности активности, получить свой идентификатор чтобы быть указанным как реферал при регистрации.

5) Backend

Бэкенд реализован на языке Python при помощи фреймворка Django и надстройки djangorestframework. С помощью Django реализуется конфигурация URL, панель администратора, настройка сервера, прокси, конфигурация паролей для отправки писем, и т.д. Почтовые запросы написаны на чистом Django без использования сторонних библиотек. SMTP сервер mail.ru используется для отправки верификационных писем. Также используется встроенная ORM для запросов в базу данных и отображения моделей при доступе через API. API (конечные точки, сериализаторы) описаны с помощью DRF. Представления используют Viewset классы с роутерами, APIView классы и отдельные декорированные функции в зависимости от сложности поставленной перед ними задачи.

Для авторизации используются JWT токены, которые сохраняются в общий контекст приложения на frontend-е (а также хранятся в локальном хранилище), а генерируются при помощи библиотеки simplejwt на сервере.

База данных в данный момент - sqlite, в будущем будет заменена на поддерживающую онлайн-доступ (MySQL/Postgres). База данных содержит таблицы, связанные с авторизацией (token для JWT аутентификации, user с данными пользователя, и т.д.), служебные (migrations, admin_log), связанные с реализацией пользователя (subscriptions для подписок, blacklist для черного списка, messages для почты) и с реализацией карты (placemark для хранения меток, activity для хранения активности пользователей, report для хранения жалоб, review для хранения отзывов и reviewpictures для хранения фотографий к отзывам, favorites для списка избранных меток). Т.к. встроенные в django авторизация и аутентификация не используются, служебные таблицы этих функций пусты.

6) Средства разработки

Backend: Django, т.к. изначально фреймворк содержит в себе все необходимые модули и нет необходимости выбирать между альтернативами. DjangoRestFramework - ввиду необходимости большого количества запросов к API, множества endpoint точек, сложных объектов, которые необходимо было бы конвертировать из queryset-формата в JSON вручную.

Frontend: React, т.к. приложение на каждой странице отправляет большое количество запросов к разным таблицам, ответы на которые необходимо динамически подгружать. Постоянная перезагрузка страницы негативно бы влияла на UX. Более того, React

использовался для реализации концепции Single Page Application, которая позволяет уменьшить время, необходимое для перехода между страницами ввиду изначальной подгрузки скриптов. React предлагает более удобные решения для маршрутизации и наследования шаблонов, чем Jinja2, используемый в чистом Django. React значительно облегчает создание динамических элементов с полученной из базы данных информацией (синтаксис JSX+функции языка JavaScript).

Значительным плюсом React также является полная независимость серверной и клиентской логики, что позволяет легче отлавливать ошибки и быстрее разворачивать приложение.

Однако решающим фактором в выборе данного фреймворка является желание реализовать впоследствии дружелюбный адаптивный дизайн, что, учитывая количество динамических форм, элементов, модальных окон и всплывающих ошибок, было бы весьма сложно без реактивного фреймворка.

Авторизация: JWT. Была выбрана ввиду популярности данного подхода, что позволяет применять полученные знания в будущих проектах. Также JWT сравнительно быстро настраивается на стороне сервера и может применяться для защиты API при помощи класса JWTAuthentication. На клиентской стороне JWT-токен обрабатывается контекстом авторизации, и потом может использоваться неограниченное количество раз при обращении к этому контексту.

Yandex-Maps API. Существует детальная документация на русском языке и большое количество пользователей, разрабатывающих конкретные доработки и делящихся опытом на GitHub Issues и Habr. Кроме того, YMaps API имеет простую и понятную React-обертку. В случае масштабирования проекта, Yandex предлагает хорошие тарифные планы для оплаты API-геокодера и запросов к карте. Проблем с оплатой сервиса возникать не будет (что вполне вероятно с Google). Наконец, yandex-карты более знакомы пользователю в России, поэтому их привычный дизайн будет не будет вызывать отторжения.

В качестве IDE сначала использовал PyCharm, т.к. не планировал Frontend на React, однако даже написание HTML-шаблонов в нем весьма затруднительно, поэтому решил попробовать Visual Studio Code. Сделал выбор в пользу VSC ввиду огромного количество плагинов и удобного интерфейса для множественных терминалов и смены ядер, на которых они работают.

Пробовал проводить отладку с Django PDB, однако для моего проекта это было менее оптимально, чем просто пользоваться средствами веб-разработчика в браузере и читать исходный код библиотек.

7) Этапы работы

Изначальная структура этапов проекта была сдвинута на два месяца ввиду большой учебной и внеучебной нагрузки в конце 10 класса. В целом, несмотря на большое количество форс-мажорных ситуаций, работа была распланирована близко к реальности и выполнена без спешки и ущерба качеству.

1) Изначальное проектирование БД было сделано при создании первой версии проекта на чистом Django, и в процессе разработки постепенно добавлялись новые таблицы. Однако когда в качестве токена для авторизации был выбран JWT (вместо встроенной django.auth), пришлось сделать кастомную модель юзера, что вынудило обнулить все миграции проекта, удалить базу данных и начать заполнять все с нуля (т.к. модель юзера должна задаваться первой миграцией).
2) В начале проекта у меня не было четкого понимания, какую карту я буду использовать. Когда был написан скелет на чистом Django и встал вопрос выбора провайдера карты, я пытался разобраться с google-картой, однако подключение и перспективы оплаты Яндекса оказались намного лучше. Я зарегистрировался как разработчик и приступил к настройке карты. Карту, балуны и метки я настроил без особых проблем. В будущем в связи с выбором React я буду вынужден переписать карту с помощью контейнеров. Также выбор React внес в проект самую большую проблему: в React Ymaps API не существовало встроенных балунов. Ее решение будет описано в следующем разделе.
3) Бэкенд должен был быть закончен к сентябрю, но я усиленно занимался летом и должен был закончить его минимум на неделю раньше. В середине августа было принято решение писать фронтенд на React, что повлекло за собой полное удаление бэкенда, т.к. он не поддерживал концепцию SPA, создавал статические шаблоны и полностью зависел от встроенной модели аутентификации Django. С основами DRF я разобрался за несколько недель и понемногу доделал весь необходимый backend. В результате серверная часть проекта была переписана полностью и закончена в октябре.
4) В данный момент проект закончен и в ближайшие дни будет задеплоен.

8) Рефлексия

Существенной проблемой для меня стала необходимость переписывать полностью бэкенд, так как я разобрался в чистом Django, потратил очень большое количество времени чтобы понять Class-Based-Views и их встроенные методы, что значительно ускорило разработку, а потом удалил все сделанное за три недели. DRF было изучить не так сложно, ведь он следует логике Django во многом, однако разница между View для просмотра/отправки информации и API Endpoint-ами все равно очень значительная.

Однако самым большим препятствием для меня стал React, т.к. я никогда не использовал реактивные фреймворки для разработки. Я многое менял в процессе разработки проекта из-за того, что нередко выбирались неоптимальные решения. К примеру, первый месяц разработки фронтенда я очень многое делал с помощью классовых компонент, что считается устаревшим подходом, все это пришлось впоследствии переписывать при помощи функциональных компонент и хуков. Разработка на React в принципе шла очень тяжело первое время, т.к. я никогда не встречался с жизненным циклом компонент, такими понятиями как state, props, и т.д. Большие трудности возникали с сочетанием асинхронных запросов и синхронных методов жизненного цикла. Я решил их, используя свои знания в области замыканий: создавал асинхронные функции, отправлял запросы в них и “замыкал” ответ в состояние компонента. Метод жизненного цикла завершал работу, а потом приходил ответ от сервера, записывался в состояние и вызывал повторный рендеринг измененного компонента. Кроме того, я реализовывал концепцию Single-Page-Application в своем React приложении, что означало необходимость создания внутреннего роутера. Роутер в его актуальной версии был добавлен не так давно, поэтому на некоторые вопросы было достаточно затруднительно найти ответы в интернете. Еще одной важной проблемой стала уязвимость маршрутов для неавторизованных пользователей и различный функционал для владельца аккаунта, стороннего наблюдателя или аутентифицированного наблюдателя. В JSX-коде компонент очень много тернарных операторов, которые регулируют наполнение страницы исходя из статуса пользователя. Некоторые страницы рендерят различные в корне компоненты в зависимости от контекста авторизации.

Проблема, изначально казавшаяся небольшой, но в результате потребовавшая огромное количество времени — отсутствие балунов в React обертке для Яндекс карт. Я потратил на решение этого вопроса неделю. Изначально я создавал отдельный компонент, рендерил его в строку и передавал в балун метки, однако внутри этого компонента переставал работать React, в частности, маршрутизация. Контекст роутера терялся внутри статического элемента и переставали работать ссылки. Я пробовал использовать классические anchor-теги, однако они не работали вместе с React-Router. Решением проблемы стало создание портала — проброса динамического элемента как ребенка в балун с помощью хука.

Также были небольшие проблемы, связанные с блокировками запросов с сервера на сервер по разным портам из-за политики CORS. Это решилось внимательной конфигурацией прокси двух серверов и установкой и настройкой пакета django-cors-headers.

В первую очередь я буду улучшать дизайн после защиты проекта. Добавлять более сложные комбинации блоков, больше анимаций, сделаю свой логотип. Т.к. React был изначально выбран чтобы быстро сделать адаптивный дизайн, я воспользуюсь этой возможностью. Я перейду на облачную базу данных, и после тестирования безопасности продукта буду предлагать услуги рекламы владельцам манежей. На сайте будут отмечены только те манежи, которые сотрудничают с проектом, их метки будут выделяться специальным логотипом. Также я недавно участвовал в организации турнире ЛФА и понял, как тяжело набрать участников, разослать им нужную информацию, так что на моем сайте в ближайшее время появится вкладка турниров со всей необходимой информацией, ссылками для регистрации и возможностью собрать команду прямо в веб-приложении. И, разумеется, будут добавляться новые возможности кастомизации профиля, более детальные настройки.

В процессе разработки проекта я приобрел огромное количество навыков, т.к. изначально все технологии были мне незнакомы. Во-первых, я очень много использовал JavaScript и мне пришлось вспоминать и разбираться в тонкостях языка. Во-вторых, я с нуля выучил такие фреймворки, как Django, DRF, React, что само по себе является базой для full-stack разработки. Разобрался, как работает авторизация, токены, получил опыт с JWT. Узнал много нового про протоколы (ssh, http, dns), в деталях разбирал HTTP запросы в процессе разработки. Получил опыт работы с Windows Linux Subsystem, PostMan, Putty, SQLite Browser, bash, инструментами разработчика Opera GX. Вел разработку с двух устройств одновременно и постоянно пользовался GitHub. Начал вести MLO, делал карту проекта в XMind. Открыл для себя контейнеризацию. Погрузился в тонкости языка Python: замыкания, алгоритмы множественного наследования, магические методы и т.д. Спроектировал большую БД и

пытался оптимизировать запросы при помощи более сложных SQL операций (порядок применения действий, облегчение запросов через выборку полей, разные виды Joins). Наконец, я очень много общался с разными разработчиками, узнал очень много нового об IT как отрасли в целом, наметил себе план развития (roadmap) на будущее. В целом, я планирую развивать этот проект как свой основной, а также брать заказы на разработку серверной логики и реализации функционала React в проектах других людей.

Почти все риски осуществились в процессе разработки проекта. Возникали проблемы с совмещением разных подходов, что привело к переписыванию бэкенда и карты. Были трудности с нюансами фреймворков, что заставило искать профессиональную помощь среди коллег на рабочем месте, в интернете. Присутствовали спорные реализации, и они разрабатывались в отдельных Git ветках. Лишь вопрос SSL-сертификата пока не стоит, т.к. все сервисы сайта успешно функционируют без шифрования и приложение еще не находится на хост-сервере.

9) Заключение

Я очень многому научился в процессе создания проекта и рад что выбрал именно IT-проект в качестве ИВР. Я получил огромный опыт, познакомился с многими замечательными людьми, улучшил коммуникацию с коллегами по работе, научился применять большое количество программ в процессе разработки. Также мне стало понятнее, чем я хочу заниматься в программировании, а какие области я бы оставил и не возвращался бы к ним. Проект в любом случае пойдет мне в портфолио, однако я всерьез планирую закончить все дополнительные работы по нему уже после защиты и вывести его в общее пользование. Мое давнее участие и большое количество связей в футбольном комьюнити поможет мне доработать нюансы, получить должную огласку и впоследствии монетизировать продукт.