29.09.23

Задача

Рассмотреть высокоуровневые решения (BAF, BaaS) для **реализации административной панели** и найти то, которое больше всех подойдёт сервису:

Большая часть задач может решаться почти любой CMS (CRUD для проектов и тегов, ручное редактирование команд). Нужно выбрать такое решение, которое позволит упростить следующие задачи:

1. **Автоматическое создание команд.**   
   Страница с кнопкой "Сгенерировать команды", где выбирается список студентов из прошедших анкету, которых надо разбить по командам. После этого создаётся черновик, который можно редактировать вручную. Черновик должен сохраняться в админ-панели. Должна быть возможность утверждать команды частями или сразу все.
2. **Просмотр и обновление метрик.**   
   Страница со списком показаний для используемых проектом метрик. Для каждой метрики делаются снапшоты, которые используются фронтендом для рендера графиков изменения метрики со временем. Должна быть кнопка обновления конкретной метрики вне очереди.

При этом преимуществом будет возможность расширения фреймворка при помощи React и/или Node.js для сокращения сложности стека технологий.

Итог

Решил использовать **strapi**. На самом деле почти все решения **отличаются очень слабо** и либо задачи эти фреймворки решают **одинаково хорошо**, либо без дополнительных надстроек задачу **не может решить ни один фреймворк**.

На strapi остановился из-за **большого сообщества**, **удобной реализации плагинов** и **понятной документации**. В остальных моментах, важных для сервиса, strapi был не хуже конкурентов.

Сложные задачи фреймворк **сам по себе не решает**, поэтому буду реализовывать через плагины.

Другие рассмотренные решения

Nocobase

+ Использование React и Node.js

+ Лёгкая расширяемость.

- Ранняя стадия разработки, частые изменения

Фреймворк не покрывает более сложные задачи, но позволяет внедрить их решение плагинами. Отказался из-за ранней стадии разработки, которая может негативно повлиять на поддерживаемость и скорость разработки проекта до выхода стабильной версии фреймворка.

Directus

+ Использование Node.js, Vue.js (не React, но достаточно близко)

+ Лёгкая расширяемость

+ Можно использовать свою бд, Directus будет для неё надстройкой

+ Высокая кастомизируемость в сравнении с другими инструментами (со strapi)

- Возможные проблемы с лицензией

Выбор по большому счёту стоял между ним и strapi. Не сказал бы, что какое-то из этих двоих решений явно лучше в контексте данного сервиса. Directus бесплатен для [“orgs with less than $5,000,000 in total finances”](https://directus.io/pricing/self-hosted), из-за этого решил не рисковать и остановиться на strapi.

django

+ Экосистема (ORM, готовая административная панель и др.)

+ Большое сообщество

- Гибкость (субъективно)

- Время на обучение

Отказался из-за необходимости полностью изменять стек технологий для бекенда и потратить время на изучение экосистемы django и python в целом. Возможно было бы использование django как headless CMS, которое не требовало бы такого тяжёлого перехода, но конкретно в этой сфере есть альтернативы, более подходящие в плане моих знаний используемых технологий.

Tooljet

Более сложный в изучении и менее надёжный из-за принципа работы. По сути - конструктор сайтов, который в том числе и предоставляет админ-панель, но не ограничивается ей. Вначале казалось, что это будет преимуществом, потому что можно будет реализовать просмотр метрик средствами фреймворка, но после чтения документации стало ясно, что редактор недостаточно гибкий для этого.

Refine

Не обладает достаточной расширяемостью и при этом не предоставляет способов решить все задачи проекта

Hashura GraphQL

Существенно отличается от остальных рассмотренных решений. Не предоставляет GUI, только способ составления запросов. Может быть полезен для других частей сервиса, но прямо сейчас среди BAF конкретно для админ-панели есть альтернативы с встроенным функционалом

19.09.23

Потенциальные проблемы

1. **Изучение технологии это вложение времени**. Все высокоуровневые технологии, с которыми мне приходилось работать на серьёзном уровне (Wordpress, Joomla), по моему мнению, не подходят проекту по причинам, которые я опишу ниже. Значит, для использования BAF-like технологий потребуется **потратить время на выбор технологии** и **потратить время на изучение выбранной технологии**.

2. **Ограниченность выбранной технологией**. При выборе высокоуровневого фреймворка часто **тяжелее использовать не предусмотренные** в технологию (и плагины для этой технологии) **решения**. Проект сейчас на очень раннем этапе и до конца не ясно, какие возможности и в каком виде он будет предоставлять (особенно в скрытой от обычных пользователей части). В связи с этим у меня есть опасения, что может возникнуть принципиально важный юзкейс, не встроенный в фреймворк. Тогда либо придётся **потратить бОльшие усилия на внедрение** этой возможности, чем без высокоуровневого фреймворка, или, в худшем случае, **придётся отказаться от данного фреймворка** в пользу другого, возвращаясь к проблеме 1.

3. **Необходимость в предоставляемых возможностях**. Представим, что выбранный фреймворк легко расширяемый и проблема 2 не актуальна. Как сильно в этом случае высокоуровневый фреймворк ускорит разработку?

Из каких сущностей на данный момент может состоять админ-панель?

- **Проекты**. Они в основном включают в себя **простые поля**, с которыми хорошо справляются высокоуровневые фреймворки. Но при этом особенно важная часть проекта это то, что у него есть своё **состояние (статус)**. В различные моменты времени страница проекта в административной панели должна выглядеть по-разному и должна быть возможность переводить проект из одного состояния в другое.

- **Студенты**. Здесь проблем нет, никаких принципиально сложных действий конкретно с этой сущностью не производится.

- **Команды**. **Ручное редактирование** уже существующих команд в BAF будет реализовать несложно, но у меня есть сомнения по поводу **автоматического создания и распределения на команды**. Можно использовать REST API для программного создания команды и включения в неё студентов, при этом оставив команду "черновиком" в админ-панели до момента ручного редактирования/подтверждения, но будет ли это удобно в использовании?

- **Анкеты**. Здесь проблем нет, можно придумать реализации, которые выбранный фреймворк будет поддерживать.

- **Метрики**. По сути сразу 3 сущности: **Metric**, **MetricInProject** и **MetricInProjectSnapshot**.

- **Metric** - самая абстрактная, описывающая метрику в целом. Сложность в использовании для высокоуровневых фреймворков зависит от реализации, но скорее всего **проблем не возникнет**.

- **MetricInProject** - используемая в конкретном проекте метрика с конкретными параметрами. В высокоуровневых фреймворках можно для каждого проекта создать отдельную страницу под редактирование всех используемых в проекте метрик, но тогда может несколько усложниться бизнес-логика созданием лишних сущностей на уровне фреймворка. В целом, однозначно **реализуемо** при помощи высокоуровневых фреймворков.

- **Snapshot**. Конкретные показания метрики в некоторый момент времени. В чистом виде сущность не используется в фреймворках, но участвует в странице просмотра показаний метрик проекта. На данном этапе неясно, какие ограничения будут на добавляемые метрики (и нужны ли эти ограничения вообще). Из-за этого отображать эту страницу средствами высокоуровневого фреймворка может быть затруднительно. Можно эту страницу вынести и за фреймворк и использовать её для просмотра показаний метрик без возможности совершать какие-то действия с ними, хоть это и **ограничит функционал**.

В целом, на данный момент кажется, что **приличная часть основных функций не будет поддерживаться** высокоуровневыми фреймворками и их плагинами "из коробки", а те, которые поддерживаются, **не слишком сложны в реализации и без BAF**. Скорее всего придётся самому вручную расширять возможности выбранного BAF. Как много смысла в этом случае от фреймворка, учитывая проблемы 1 и 2?

Всё написанное выше можно отнести скорее к вопросу **скорости разработки**. Высокоуровневые фреймворки очевидно выигрывают в плане **надёжности**, при условии, что необходимые возможности реализованы фреймворком. **Поддерживаемость** будет обеспечиваться использованием только распространённых технологий и принятием архитектурных решений, напрямую не связанных со стеком. Вопрос **тестируемости** зависит от конкретного фреймворка, но мне кажется, что выбранный на данный момент стек в достаточной степени её обеспечивает.

Планы на неделю

Таким образом, в ближайшую неделю я планирую заняться изучением конкретных фреймворков из предложенной подборки в контексте пункта 3 перечисленных потенциальных сложностей. Если я смогу найти решение, достаточно покрывающее юзкейсы (необязательно сразу все, но хотя бы некоторые из сложных в реализации) или смогу преобразовать некоторые запланированные функции сервиса под возможности фреймворка, то тогда начну более подробное изучение конкретного найденного фреймворка.