

МИНЕСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ
УНИВЕРСИТЕТ ИМ. В.Г. Шухова»

Расчётно-графическое задание
Разработка web-приложения

Выполнил: ст. гр. ВТ-191
Фахретдинов Владислав Станиславович

Проверил: ст.преп.
Картамышев Сергей Владимирович

Белгород 2022

Содержание

1	Введение	3
2	Проектирование	3
3	Создание frontend стороны на Vue	4
4	Создание backend стороны на Django	5
5	Полученные результаты	6

1 Введение

На текущий момент, разработка web-приложений является одной из самых распространённых областей в мире, так как охватывает наибольшее количество устройств, а использование этих приложений не требует ничего, кроме браузера. Мы имеем множество платформ, начиная от социальных сетей, стриминг площадок, интернет магазинов и дистрибьюторов, информационных и обучающих ресурсов, которые представляют из себя web-приложения и которыми мы пользуемся каждый день для своего удобства и получения образования и новых навыков, удовольствия, веселья и так далее.

В качестве цели, в данной работе был выбран сайт-визитка - «Лаборатория анализа данных Владислава Фахретдинова», который будет предоставлять информацию о гипотетической "компании".

2 Проектирование

Для того, чтобы начинать разработку, нам необходимо представлять себе структуру сайта, его особенности и функционал. Так как наш сайт будет представлять из себя визитку компании, мы хотим иметь несколько "страничек которые будут предоставлять основную информацию.

1. Главная страница, на которой будут описаны примерные предоставляемые услуги.
2. Демонстрация, на которой можно будет воспользоваться одной из современных разработок.
В качестве демонстрации, выберем суммаризацию текста, одну из распространённых задач.
3. Страница отзывов, на которой будет несколько отзывов от наших клиентов.
4. Страница контактов, на которой будут представлены сотрудники лаборатории, с описанием должностей и возможностей.

Помимо этого, наш сайт будет иметь титульную верхнюю часть страницы, с помощью которой можно будет переключаться между описанными выше страничками.

Для реализации всего этого, будем использовать JavaScript фреймворк Vue, который является одним из самых популярных фреймворков на текущий момент. В его плюсы входит отличная документация со множеством примеров, адаптивность, превосходная интеграция, масштабирование.

Описав примерный интерфейс, теперь нам необходимо понять и описать, что и как наш сайт будет получать от сервера. В нашем случае, для каждой из страничек 1, 3 и 4, описанных выше, потребуется получение лишь одного списка примерно

похожих элементов. Для 2 страницы, нам необходимо будет передавать текст, набранный пользователем, на сервер, и получать в ответ суммаризованный текст для дальнейшего его отображения. Запросы к серверу будем делать с помощью библиотеки `axios`.

В качестве языка программирования и фреймворка, для написания серверной части, были выбраны `python` и `Django` из-за их простоты и возможности гибкой реализации и плотной интеграции с другими модулями. Хранение данных для отображения будем проводить в `SQLite`, так как объём информации небольшой, не требуется высокая производительность, а сами данные простые и между ними нет сложных связей (услуги, отзывы, контакты).

Наконец, можем перейти к разработке.

3 Создание frontend стороны на Vue

Web-приложение на Vue состоит из двух основных логических блоков - «View» или страниц, и «Component» или же компонент.

Страницы из себя представляют какую-то часть реальных страниц на сайте, которые можно переиспользовать вместо других, оставляя при этом некоторые фиксированными. В нашем случае, для каждой из страниц которые мы описывали во введении, будет представлено соответствующее View - `ServicesView`, `DemoView`, `ReviewsView` и `ContactsView`.

В свою очередь, компоненты представляют ещё меньший кусок страницы, обычно это элементарные единицы, которые зачастую могут быть представлены на страницу в виде нескольких экземпляров, в которых изменяется только информация. Мы будем создавать такие компоненты, как `Service`, `Review`, `Demo`, `Contact` и `Link`, а также `Header` и `Footer`, которые будут оставаться на каждой из страниц.

Код страниц, или же View, представляет из себя несколько секций. Первая из секций, это `<template>`, которая задаёт разметку страницы на адаптированном под нужды Vue HTML, в ней можно описывать создание компонент, а также использовать классические блоки, которые мы использовали ранее. Также одной из важных секций является `<style>`, в которой мы описываем наши стили, по аналогии с `css`, которые будут применяться к нашей разметке. Есть также `<script>` секция, в которой мы описываем наше View, его методы, способы получения информации, используемые компоненты, создание данных и обращение к серверу - всю основную часть, которая работает невидимо для обычного пользователя, и позволяет сайту быть таким, каким мы его задумываем.

Код компонент состоит из тех же блоков, но логически представляется оно по другому. Секция `<template>` теперь описывает один блок с переданной ему информацией, а `<style>` настраивает как раз взаимодействие не между элементами, как во View, а внутри одного элемента.

4 Создание backend стороны на Django

Наш сервер будет отвечать за реализацию API, по которому frontend часть сможет получать информацию. Будем реализовывать четыре метода. Три из них будут GET, то есть отдавать нам информацию на наш запрос, это services, reviews, contacts. И один из них будет POST, он уже будет отдавать нашу информацию в модифицированном виде - это demo.

Для сохранения данных, в Django существуют модели, которые позволяют воспользоваться ORM подходом. Опишем наши модели данных:

```
class Service(models.Model):

    title = models.CharField(max_length=30)
    image = models.ImageField()
    text = models.TextField()

    def __str__(self):
        return self.title


class Review(models.Model):

    text = models.TextField()

    def __str__(self):
        return self.text[:20]


class Contact(models.Model):

    image = models.ImageField()
    fullname = models.CharField(max_length=30)
    title = models.CharField(max_length=30)
    description = models.TextField()

    def __str__(self):
        return self.fullname
```

```
class Link(models.Model):
    logo = models.ImageField()
    url = models.CharField(max_length=30)
    contact = models.ForeignKey(Contact, on_delete=models.CASCADE)

    def __str__(self):
        return self.
```

После того, как мы задали модели для наших данных, осталось реализовать методы, которые будут отвечать на запросы и отдавать информацию. Используя модели, по большей части эти методы представляют из себя получение всех объектов и формирование json'a а в том формате, в котором мы принимаем его на frontend стороне.

5 Полученные результаты

После длительного процесса вёртки и создания интерфейса, формирования модели данных, согласования самим с собой api для получения этих данных, реализации отправки их с сервера - можем сказать, что сайт готов. Посмотреть на UI можно по ссылке: <https://emperornao.github.io/web/>.

Также, весь код доступен в репозитории <https://github.com/EmperorNao/web>.



Рис. 1: Главная страница

Суммаризация текста

Суммаризация — автоматическое создание краткого содержания, например заголовка, резюме, аннотации, на основе исходного текста.

Введите исходный текст

Попробовать

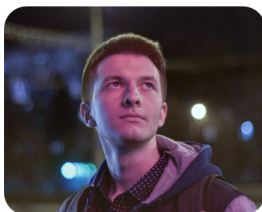
Рис. 2: Демонстрация

Анатолий Волчко, директор по коммуникациям, "Абсолют": «Наша компания заказывала в лаборатории сбор и обработку данных. Своих рук на всё не хватает. Радует гибкий подход: о полях и формате выгрузки всегда можно договориться.»

Евгения Васильева, руководитель "Inter Vision": «Мы много взаимодействовали с лабораторией, в основном в области компьютерного зрения: написание инфраструктуры для сегментации в реальном времени, дообучение YOLO моделей. Работа была выполнена в срок, порадовало качество и налаженность взаимодействия.»

Даниил Танжиров, руководитель "Push To Talk": «Заказывали интерактивного чат-бота для замены персонала поддержки. Понравилось, что работы были сделаны в срок, у нас не возникло проблем с интеграцией и решение сразу заработало с хорошим качеством.»



Рис. 3: Отзывы

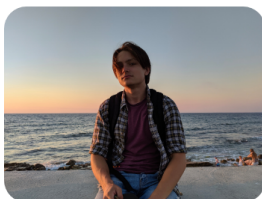


Фахретдинов В.С

Главный разработчик, 20 у.о

Создатель лаборатории и по совместительству главный разработчик.
Middle ML Software Engineer developer. 3 года опыта разработки: в
прошлом аналитик данных в "Белгородский информационный фонд",
на текущий момент разработчик машинного обучения в "Яндекс".

 <https://vk.com/emperorao>
 <https://t.me/emperorao>
 <https://github.com/EmperorNao>



Товстоганов А.К

Бертухай младший, 21 у.о

Вечный студент, младший разработчик.

 https://vk.com/lexus_flexus
 <https://t.me/lexusflexus>
 <https://github.com/Lexus1227>

Рис. 4: Контакты