МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»

Кафедра теории функций и стохастического анализа

РАЗРАБОТКА WEB-САЙТА И СИСТЕМЫ ОБСЛУЖИВАНИЯ НА ОСНОВЕ ТЕЛЕГРАММ-БОТА ДЛЯ ДОМА КУЛЬТУРЫ

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студентки 4 курса 451 группы
направления 38.03.05 — Бизнес-информатика

механико-математического факультета
Булгаковой Анастасии Сергеевны

Научный руководитель
ст. препод. _____ Н. В. Сергеева

Заведующий кафедрой
д. ф.-м. н., доцент _____ С. П. Сидоров

Саратов 2025

ВВЕДЕНИЕ

Актуальность темы исследования. В наше время наличие веб-сайта является неотъемлемой частью любого бизнеса или организации. Эффективный веб-сайт может не только повысить узнаваемость бренда, но и привлечь клиентов и увеличить продажи. Веб-сайт служит эффективным средством представления информации, взаимодействия с пользователями и продвижения продуктов или услуг. База данных, в свою очередь, обеспечивает надежное и структурированное хранение данных, необходимых для функционирования веб-приложений и информационных систем.

Таким образом, актуальность дипломной работы обусловлена высоким спросом на специалистов в области веб-разработки и баз данных, а также возможностью дальнейшего профессионального роста и развития в сфере информационных технологий.

Актуальность определила выбор темы данной работы: «Разработка Web-сайта и системы обслуживания на основе телеграмм-бота для дома культуры».

Цель работы: развитие HTML-сайта на основе предыдущих разработок для интеграции с базой данных, а так же разработка телеграмм-бота. В ходе работы будут изучены основные принципы и технологии Web и Backend разработки, построения клиент-серверного взаимодействия.

Для достижения поставленной цели в работе необходимо решить следующие задачи:

— Провести анализ существующих веб-технологий и подходов к разработке клиент-серверных приложений;
— Разработать структуру и пользовательский интерфейс (Frontend) веб-сайта;
— Спроектировать и реализовать серверную часть (Backend) веб-сайта;
— Создать базу данных;
— Интегрировать Frontend и Backend веб-сайта;
— Разработать Telegram-бота;
— Провести тестирование разработанного веб-сайта и Telegram-бота.

Практическая значимость работы заключается в реализации проекта, который поможет наращиванию пользовательской аудитории и поспособствует удобному пользованию.

Основное содержание работы

Данная работа состоит из: введения, одной теоретической и трех практических глав, заключения, списка использованных источников, приложения.

Введение содержит основные положения: актуальность темы иссле дования (цель, объект, предмет, задачи исследования); практическую значи мость исследования.

В первом разделе происходит размышление о технологиях которые будут применяться для дальнейшей разработки, приводится сравнение между аналогами для выявления лучших среди них.

Выбор технологий, используемых в Web-разработке. HTML(это язык гипертекстовой разметки) и CSS (это язык стилей, который используется для описания внешнего вида веб-страниц) являются основополагающими технологиями, используемыми в веб-разработке.

Вместе они создают основу для создания интерактивного, привлекательного и удобного для пользователя веб-контента.

В данной дипломной работе был выбран наиболее удобный и доступный метод разработки сайта при помощи IDE Rider от компании JetBrains. Данная IDE является лидером для проектирования приложений на языке C#, а так же имеет большие возможности для верстки и отладки Web приложений.

Технологии для Backend разработки. Для реализации Backend части сайта для дома культуры выбран ASP.NET (фреймворк для создания веб-приложений на языке программирования C # из-за большого списка плюсов и сторонних библиотек которые могут быть интегрированы на серверную часть.

Для хранения пользовательских данных в Web-приложениях была выбрана реляционная база данных PostgreSQL.

Для реализации визуальной составляющей Web сайта необходимо использовать концепцию представленную ASP.NET Core Razor Pager.

Razor Pages — это концепция, основанная на шаблонах, которая помогает организовать веб-страницы. Каждая страница представляет собой модель страницы (Page Model), которая включает как C sharp код, так и представление в виде HTML-разметки. Это делает Razor Pages более компактными и удобными. Каждая страница сайта представлена двумя файлами.

Для обеспечения безопасности необходимо реализовать систему авторизации на Web-сайте. Для этого серверу необходимо удостовериться в подлинности пользователя или передать какие-то данные в зашифрованном виде между различными частями системы. Для этого будут использоваться JWT токены. JWT (JSON Web Token) — это открытый стандарт (RFC 7519), который определяет компактный и самостоятельный способ передачи информации между двумя сторонами в виде JSON-объекта.

Для повышения отзывчивости веб-сайта и эффективности сервисов в микросервисной архитектуре, особенно при длительных операциях, таких как запись в базу данных, необходимо использовать брокер сообщений. В данном проекте был выбран RabbitMQ (брокер сообщений, который позволяет приложениям обмениваться сообщениями через очередь).

После создания серверного API необходимо убедиться в правильности его работы. Для тестирования API будет применен Swagger (это популярный набор инструментов для разработки, документирования и тестирования RESTful API).

После разработки необходимо опубликовать проект на удаленный сервер. Для упрощения этого процесса следует использовать контейнеризацию, создание Dockerfile, в котором будет описан процесс сборки всего приложения и сборка его в единый контейнер.

Во втором разделе приводится Frontend разработка будущего сайта.

Структура и навигация. Перед переходом к верстке необходимо иметь представление о нужных разделах:

— Главная страница - содержит перечень всех нижеописанных разделов;
— Коллективы - этот раздел представляет собой галерею профилей артистов, групп и ансамблей, хореографов и постановщиков, чьи работы становятся основой культурной жизни города;

— Расписание занятий - данный раздел предоставляет удобный и интуитивно понятный интерфейс, позволяющий пользователям легко искать занятия по датам, наименованиям групп или по номерам кабинетов;
— Покупка билетов - раздел представляет собой форму для приобретения билетов на будущие мероприятия. Покупка осуществляется путем предоставления пользователем данных о банковской карте;
— Контакты - перейдя на эту вкладку пользователь может связаться с контактными лицами по телефону, почте, мессенджерам или перейти в телеграмм-бот, который так же поможет решить насущные вопросы;
— Авторизация - перейдя на эту вкладку пользователь может зарегистрироваться или войти в уже созданный аккаунт. Это необходимо для покупки билетов.

На основе таблиц стилей и HTML разметки создаем макет сайта, описанного ранее. Используя технологию ASP.NET Razor Pages, можем оптимизировать разработку отдельных страниц сайта, предотвращая дублирование кода.

Дизайн и вёрстка. Дизайн - является визитной карточкой сайта, и его грамотное планирование, выбор цветовых гамм и шрифтов играет ключевую роль в впечатлении, которое сайт производит на посетителей.

Проанализировав большое количество современных Web-сайтов и получив обратную связь от ведущих разработчиков, было принято решение использовать минималистический стиль оформления сайта, так как он явялется наиболее удобным для восприятия информации. Сайт оснащен достаточным количеством значков, которые улучшают визуальный аспект, делая его более понятным и привлекательным для пользователей.

В третьем разделе описывается серверная разработка как для самого сайта, так и для телеграмм-бота. Рассматриваем стандарты безопасности, продумываем архитектуру будущего проекта.

Моделирование Базы Данных. Модель «Сущность-связь» (ER-модель) представляет собой графическое представление структуры базы данных, где сущности (объекты) и связи между ними отображаются в виде диаграммы. ER-модели используются для проектирования схем баз данных, обеспечивая понятное и удобное визуальное представление структуры данных.

Созданная база данных предназначена для хранения, накопления и предоставления данных о новостях и различных спектаклях, о коллективах и расписании занятий, о зарегистрированных пользователях и купленных билетах.

Разработка ASP.NET приложения. Исходя из выбранных ранее технологий, создадим Backend приложение, которое будет обрабатывать все поступающие запросы с сайта на сервер. Для реализации был выбран микросервисный стиль разработки.

Необходимо реализовать разбиение на сервисы таким образом, чтобы каждый из них имел уникальную зону ответственности, при этом деление на слишком большое количество потребует более сложной инфраструктуры и управления. В данном случае были разработаны следующие виды сервисов:

— News-Service: будет обрабатывать запросы, которые поступают на создание, удаление и отображение новостей на сайте;
— Event-Service: будет использоваться, как сервис, который позволяет создавать транзакций на покупку билетов и будет хранить информацию о текущих концертах, на которые можно купить билеты;
— Schedule-Service: Используется для работы с пользователем. Через него будет происходить регистрация новых пользователей, отображение расписания пользователей.

Помимо сервисов, отвечающих за функции сайта, нужны и сторонние сервисы. Необходимо создать дополнительные сервисы, которые будут способствовать работе вышеописанных сервисов. К таким сервисам можно отнести:

— PostgreSQL-Service: в данном случае база данных всего сайта, как и все остальные сервисы, находится в Docker и представляет отдельный сервис, что сильно упрощает процедуру переноса готового проекта на удаленный сервер;
— Email-Service: необходим для отделения логики, отправки кодов подтверждения и любой другой информации на почту пользователя;
— RabbitMQ-Service: служит прослойкой между Email-Service и другими сервисами для получения быстрого отклика для пользователя, так как

некоторые операции могут выполняться непозволительно долго, что может повлиять на опыт работы с сайтом.

Реализация News-Service. Для данного сервиса и всех последующих сервисов база данных строится на основе C# классов.

Используя атрибуты, добавленные ORM библиотекой, можно наделить пере- менные из языка C # свойствами столбцов из базы данных. Рассмотрим основные атрибуты:

— Key: Данное поле выступает в роле PK в таблице;
— Required: Данное поле обязательно к заполнению;
— MaxLength: Указывает на максимальную длину строки, которая может быть сохранена в таблице;
— ForeignKey: Указывает, что данный столбец является ссылкой на первичный ключ другой таблицы.

Создадим класс News, который будет содержать время создания, заголовок, текст новости.

В задумке к каждой новости могут прикрепляться фотографии, причем в неограниченном количестве. Для этого необходимо создать дополнительную таблицу, в которой будем хранить фотографии.

Для того что бы реализовать загрузку фотографии, которая поступает, как объект IFromFile, необходимо создать функцию, которая переводит файл в поток байт и сохраняет атрибуты файла. Опишем функцию, которая принимает объект IFromFile и возвращает массив байт.

После того, как реализована возможность хранения файлов и новостей, необходимо добавить методы, которые будут отправлять существующие новости на сайт. Для достижения данной цели используем контроллеры. Необходимо создать 2 точки API: на первой клиент будет получать объект типа News, а для загрузки каждой фотографии по отдельности будет использоваться 2 точка API. Создадим класс NewsController, в который будут приходить запросы сайта на создание, добавление новостей.

Для того что бы NewsController мог исполнять свои обязанности, необходимо в файле конфигурации подключить к приложению контроллеры. Для этого на моменте конфигурации приложения, зарегистируем контроллеры в

списке сервисов, а после сборки приложения, добавим контроллеры в конвейер обработки запросов.

Для тестирования используется Swagger. После разработки API, которое позволяет работать с новостями, его можно увидеть и протестировать через SwaggerUI. Чтобы подключить Swagger в ASP.NET приложение необходимо интегрировать библиотеку Swashbuckle и OpenApi и на этапе сборки приложения, добавить их в список сервисов.

Авторизация и аутентификации пользователей. Для покупки билетов на любое из мероприятий пользователь должен пройти процесс регистрации. Для создания этого механизма нужно:

— Реализовать процедуру аутентификации. Аутентификация — это процесс проверки личности пользователя. В этом процессе система проверяет, кто именно пытается получить доступ. Это обычно осуществляется с помощью учетных данных. В данном случае будет использоваться связка из пользовательских данных вида пароль и почта;

— Реализовать механизм авторизации. Авторизация — это процесс определения, какие ресурсы и действия доступны для аутентифицированного пользователя. То есть, даже если аутентификация пройдена успешна и была доказана своя личность, система решает, что именно разрешено делать в рамках текущих прав и ролей. Например, роль администратора и обычного пользователя является совершенно разной, хотя оба прошли процедуру аутентификации.

Для начала необходимо реализовать и понять какие роли будут существовать. Для их декларации использует UserType типа Enum. Данное перечисление будет хранить виды ролей, которые будут существовать на сайте.

После чего, используя ORM, опишем сущность User, в которую поместим все необходимые поля для работы системы. Необходимы следующие поля:

— 1. HashPassword — используется для хранения пароля в зашифрованном виде по стандартам SHA256. Будем использовать для сравнения с паролем, который приходит во время входа на сайт;

— 2. IsEmailConfirmed — используется, чтобы понимать подтверждена ли почта у пользователя или нет;

— 3. CreatedAt — хранит время создания объекта в базе данных;
— 4. UpdatedAt — хранит время последнего обновления в базе данных;
— 5. Salt — используется, как кусок пароля, который генерируется на стороне сервера и увеличивает пароль клиента на некоторое количество элементов, что повышает надежность всей системы и безопасность для пользователей;
— 6. PublicId — тот id, который используется пользователем для отправки запросов на сервер. Используется стандарт GUID.

Как говорилось в начале, будет использоваться механизм JWT для подтверждения пользователей. Что бы подключить механизмы авторизации и аутентификации, необходимо добавить их в файл конфигурации.

Для этого необходимо на стороне сервера настроить обработку. Для этого необходимо считать уникальный ключ, на основе которого будут генерировать токены доступа для API.

На стороне клиента необходимо уметь правильно хранить данные токены доступа и при любом запросе на сервер встраивать их в запрос. Для этого создаем Event, который будет отвечать за управление Access токена.

Обработка исключений. Один из важнейших принципов при разработке клиент серверного взаимодействия — это стандартизированная форма общения между клиентом и сервером. В случаях, когда клиент посылает данные, сервер должен быть готов обработать любого рода запросы, в том числе и не соответствующие стандартам.

Для обработки таких исключительных ситуаций необходимо реализовать систему, которая не будет зависеть от какого-то сервиса, а может быть добавлена в любой нужный сервис.

Для добавления необходимо реализовать собственный MiddleWare, который можно встроить в конвейер обработки запроса.

Для этого создадим абстрактный класс ApiErrorResponse, который будет отправляться клиенту в случае, если на сервере была ошибка.

Отправка сообщений на Email. Для того, чтобы создать возможность отправки сообщений на почту, необходимо настроить Smtp профиль на сервере. Для этого создадим структуру, в которую поместим все необходимые

данные. Будем использовать почтовый сервис от компании yandex, так как он предоставляет бесплатный доступ для работы. Создадим класс SmtpSettings, в котором укажем адрес почты, с которой будут оправляться письма пользователем, а так же пароль отданной почты, порт и SSL сертификат.

Создадим класс EmailService, который будет лежать в основе отправки писем на почту. В данном случае необходимо реализовать метод Send, который принимает следующие аргументы:

— EmailMessageType type: аргумент, на основе которого будет строиться вид письма, отправляемого клиенту;
— EmailLanguage language: язык, на котором будет написано сообщение пользователю;
— BaseEmailArgs args: необходимые аргументы. Например: имя, почта на которую будем отправлять письмо.

Оркестрация сервисов. Для сборки всех компонентов в единое целое необходимо создать DockerCompose файл, в котором будем описывать как те или иные сервисы будут взаимодействовать друг с другом. Для этого создадим виртуальную локальную сеть, к которой подключим все сервисы. Это позволит производить общение между ними без выхода в глобальный интернет. Так же для PostgreSQL и RabbitMQ сделаем хранилище, которое позволит переиспользовать данные, сохраненные в базу данных после перезапуска контейнера.

Для настройки PostgreSQL, в качестве контейнера, используем официальный образ PostgreSQL. Подключаем его к существующей виртуальной сети, а так же указываем путь до хранилища данных. Используем порт 5432, на который другие сервисы будут отправлять запросы.

Для настойки основных сервисов необходимо понимать следующий факт. Необходимо обязательно гарантировать, что некоторые сервисы, такие как PostgreSQL-Service или RabbitMQ-Service, будут запущены раньше, чем Schedule-Service, так как Schedule-Service на момент запуска должен установить с ними соединение, иначе будет ошибка подключения.

В четвертом разделе акцент переходит на телеграмм-бота. На основе уже разработанного API создаем удобный интерфейс для работы с пользова-

телем.

Разработка телеграмм-бота.

При разработке телеграмм-бота будем опираться на созданный серверный API, описанный в прошлой главе. При разработке будем использовать библиотеку iogramm для языка Python, которая будет лежать в основе телеграмм-бота. Для удобства пользователей откажемся от ввода сообщений, предоставив пользователю удобный ввод благодаря кнопкам.

Для работы с API необходимо создать сущности, с которыми будем работать после получения данных от сервера. Одной из таких сущностей является ClientLesson. Данная сущность является аналогом того, что было описано на языке C #. Для работы необходимо создать ее вновь на языке Python.

Для общения телеграмм-бота с уже разработанным API реализуем класс, который будет отправлять и принимать данные от сервера. Для этого используем библиотеку requests, которая позволяет устанавливать http соединение.

Основные результаты

— Проведен анализ существующих веб-технологий и подходов к разработке клиент-серверных приложений;
— Разработана структура и пользовательский интерфейс (Frontend) веб-сайта;
— Спроектирована и реализована серверная часть (Backend) веб-сайта;
— Создана база данных;
— Разработан Telegram-бота;
— Проведено тестирование разработанного веб-сайта и Telegram-бота.

Программный код приводится в **Приложении А.**