Внедрение современных подходов к написанию кода, основанных на надежных архитектурах, играет ключевую роль в обеспечении возможности проведения успешного расширения приложений в любое время. Особенно если говорить о данном *web*-решении, которое может с легкостью перерасти в большую и сложную информационную систему. Эти подходы не только способствуют повышению эффективности разработки, но и обеспечивают стабильность и гибкость системы для последующего развития. Реализация таких методов необходима для современных проектов, стремящихся к долгосрочной успешной эксплуатации и удовлетворению потребностей пользователей в быстро меняющейся среде информационных технологий.

Литература

1. Новости / Нац. правовой интернет-портал Респ. Беларусь. – 2024. – Режим доступа: https://pravo.by. – Дата доступа: 02.04.2024.
2. Документация / FeatureSliceDesign. – 2024. – Режим доступа: https://feature-sliced.design/docs. – Дата доступа: 02.04.2024.

WEB-АГРЕГАТОР ПО ПРОДВИЖЕНИЮ УСЛУГ СПЕЦИАЛИСТОВ ОБСЛУЖИВАНИЯ МУЗЫКАЛЬНЫХ ИНСТРУМЕНТОВ

М. А. Макеев

Учреждение образованимя "Гомельский государственный технический университет имени П. О.Сухого», Республика Беларусь

Научный руководитель Н. В. Самовендюк

Описано web-приложение, предоставляющее: интерактивное взаимодействие между клиентом и необходимым специалистом обслуживания музыкальных инструментов, информацию о специалистах обслуживания музыкальных инструментов и их услугах, возможность продвижения услуг специалистов музыкальных инструментов.

Ключевые слова: web-агрегатор, музыкальные инструменты, специалисты обслуживания, многослойная архитектура.

В современном цифровом мире все больше и больше людей обращаются к интернету для поиска различных услуг и специалистов. Музыкальная индустрия не является исключением. Многие музыканты и владельцы музыкальных инструментов ищут специалистов, которые могут обеспечить обслуживание, ремонт или настройку их инструментов.

Однако, в сети интернет существует огромное количество информации и ресурсов, что может затруднять процесс поиска и выбора квалифицированных специалистов. В такой ситуации, создание *web*-агрегатора, специализирующегося на продвижении услуг специалистов обслуживания музыкальных инструментов, становится актуальной задачей.

Основная задача разрабатываемого *web*-приложения – это помочь людям, которые столкнулись с проблемами поиска специалистов обслуживания музыкальных инструментов. Для различных музыкантов нашей страны остро стоит проблема с поиском услуг обслуживания музыкальных инструментов. Соответственно, данное приложение должно облегчить поиск специалистов, а также предоставить данные о ценах и репутации соответствующих мастеров обслуживания для различных регионов.

Главное преимущество такого агрегатора в том, что он самодостаточен: пользователи, как клиенты так и мастера, сами следят за актуальностью и качеством предоставляемой информации. За счет гибкой системы оценивания и отзывов, а также переписки между клиентом и специалистом, между пользователями идет интерактивное взаимодействие с непосредственным предоставлением актуальной информации. Несмотря на это, осуществляется модерация приложения для соблюдения порядка.

Приложение предоставляет следующие функции:

– регистрация и создание профилей;

–размещение объявление с фото и описанием;

– поиск и фильтрация;

– рейтинг и отзывы;

– бронирование услуг;

– возможность вести переписку;

– администрирование;

Функционал приложения разбит на роли:

– гость (незарегистрированный пользователь);

– специалист обслуживания;

– пользователь(клиент);

– администратор.

Структура разработанного приложения основана на клиент-серверной многослойной архитектуре, которая имеет ряд преимуществ перед другими типами архитектур построения программ.

Многоуровневая клиент-серверная архитектура – это подход к разработке приложений, в котором функциональность приложения разделена на отдельные уровни, выполняющие определенные задачи. Каждый уровень имеет свою специализацию и взаимодействует с другими уровнями по определенным правилам и протоколам.

В такой архитектуре обычно выделяют следующие уровни:

– клиентский уровень (представление);

– уровень логики приложения (бизнес-логика);

– уровень доступа к данным.

Каждый уровень взаимодействует с соседними уровнями по определенным протоколам и интерфейсам. Например, клиентский уровень отправляет запросы к серверу через API, а серверный уровень обрабатывает эти запросы, выполняет необходимую бизнес-логику и возвращает данные обратно клиенту.

Рассмотрим клиентский уровень приложения. Он отвечает за взаимодействие между пользователем и системой, предоставляя пользователю возможность ввода данных, отображения информации и взаимодействия с функциональностью приложения. Его основная задача состоит в представлении данных и функциональности системы в удобной и интуитивно понятной форме для пользователей. Он обеспечивает визуальное представление данных, элементы управления, механизмы взаимодействия и обратной связи с пользователем.

В данном приложение на стороне клиента в качестве уровня представления был выбран инновационный фреймворк Blazor. Выбор Blazor в качестве технологии отображения данных обусловлен его преимуществами, такими как удобство разработки, повышенная производительность и возможность использования C# на клиентской стороне. Он позволяет создавать современные пользовательские интерфейсы, которые могут быть легко интегрированы с серверной логикой приложения, что делает его незаменимым в разработке приложения с использованием технологий ASP.NET.

Уровни логики приложения и доступа к данным располагаются на серверной части приложения.

Технологией используемой в данном web-приложении на стороне сервера является ASP.NET CoreWebAPI. Выбор ASP.NET Core Web API в качестве сервера обработки данных обусловлен его надежностью, высокой производительностью, кроссплатформенностью и интеграцией с другими инструментами .NET. Он предоставляет разработчикам удобные средства для создания масштабируемых и современных web-сервисов, которые могут быть использованы в широком спектре приложений и сценариев.

Слой бизнес-логики является центральной частью архитектуры программного обеспечения. Он обрабатывает бизнес-правила и логику приложения, принимает решения и управляет бизнес-процессами. Слой бизнес-логики служит посредником между пользовательским интерфейсом и слоем доступа к данным, обеспечивая обработку данных и применение правил бизнеса.

В разработке слоя бизнес-логики были применены такие технологии как маппинг данных с помощью библиотеки AutoMapper, паттерн проектирования внедрения зависимостей (DependencyInjection)который позволяет управлять зависимостями объектов и обеспечивает инверсию управления (InversionofControl, IoC), а также в качестве реализации механизмов авторизации и аутентификации была применена технология ASP.NET Identity.

Слой доступа к данным представляет собой компонент или слой в приложении, который отвечает за взаимодействие с базой данных или другими источниками данных. Он предоставляет абстракцию для выполнения операций чтения, записи, обновления и удаления данных, скрывая детали конкретной реализации базы данных от остальных компонентов приложения.

В качестве ORM (Object-Relational Mapping) системыбылвыбран Entity Framework. EntityFramework является мощным инструментом, который обеспечивает удобное взаимодействие с базами данных, позволяя разработчикам работать с данными в виде объектов и сущностей, а не непосредственно с SQL-запросами. Выбор EntityFramework в данном приложении был обусловлен его преимуществами, такими как удобство использования, интеграция с ASP.NET Core, поддержка миграций базы данных и работа с LINQ. Он предоставляет эффективные инструменты для работы с данными и помогает ускорить разработку и улучшить общую производительность приложения.

В данном приложении была выбрана Microsoft SQL Server (MS SQL)в качестве базы данных. MS SQL является одной из наиболее популярных и надежных реляционных баз данных, разработанных и поддерживаемых Microsoft. Главным преимуществом MS SQL является его интеграция с другими инструментами и технологиями Microsoft, такими как .NET Framework и Entity Framework. Это обеспечивает гладкую работу между базой данных и приложением, а также упрощает разработку и поддержку.

На рис. 1 можно увидеть получившуюся структуру приложения.

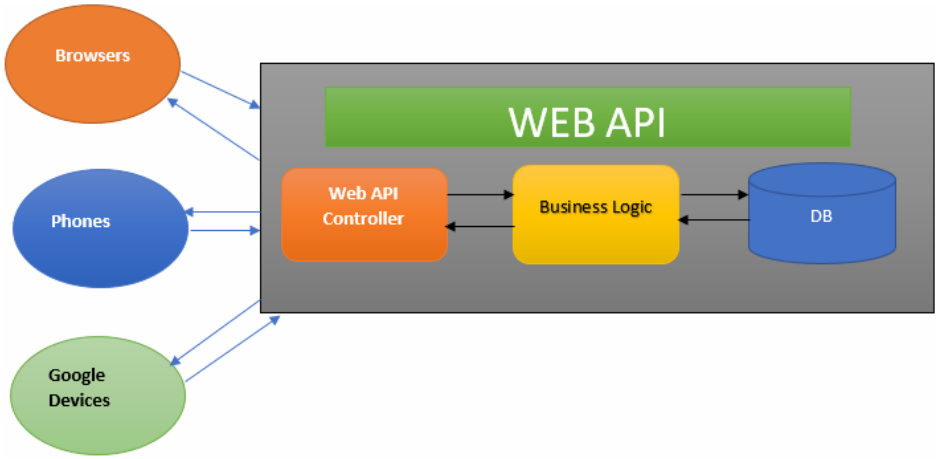

Рис. 1. Архитектура приложения

Ключевыми сущностями в базе данных являются: Services, Appointments, Conversations, Messages, Ratings. Ниже представлен подробный разбор каждой из этих сущностей.

Сущность Services позволяет хранить информацию о различных услугах обслуживания музыкальных инструментов, и каждая запись содержит идентификатор услуги, идентификатор мастера, идентификатор инструмента, название услуги и ее цену. Эти данные могут быть использованы для отображения доступных услуг, расчета стоимости обслуживания или выполнения операций связанных с управлением услугами в системе обслуживания музыкальных инструментов.

Сущность Appointments позволяет хранить информацию о назначенных приемах в системе обслуживания музыкальных инструментов. Каждая запись содержит идентификатор приема, идентификатор клиента, идентификатор мастера, идентификатор услуги и дату/время приема. Эти данные используются для планирования и отслеживания приемов, связанных с обслуживанием музыкальных инструментов, а также для выполнения операций связанных с управлением записями на прием в системе.

Сущности Conversations и Messages позволяют хранить информацию о диалогах и сообщениях, связанных с общением между клиентами и мастерами в системе обслуживания музыкальных инструментов. Каждая запись в сущности Conversations содержит уникальный идентификатор диалога, идентификатор клиента и идентификатор мастера. Каждая запись в сущностиMessages содержит уникальный идентификатор сообщения, идентификатор связанного диалога, идентификатор отправителя, текст сообщения и временную метку. Эти данные используются для отслеживания и сохранения истории коммуникации, а также для выполнения операций связанных с управлением диалогами и сообщениями в системе.

Сущность Ratings позволяет хранить информацию о рейтингах и отзывах, оставленных клиентами о мастерах. Каждая запись содержит уникальный идентификатор рейтинга, идентификатор клиента, идентификатор мастера, рейтинг и текстовый отзыв. Эти данные используются для отображения рейтингов мастеров, анализа качества обслуживания и принятия решений по улучшению сервиса в системе обслуживания музыкальных инструментов

Схематическое представление базы данных предметной области изображено на рис. 2.

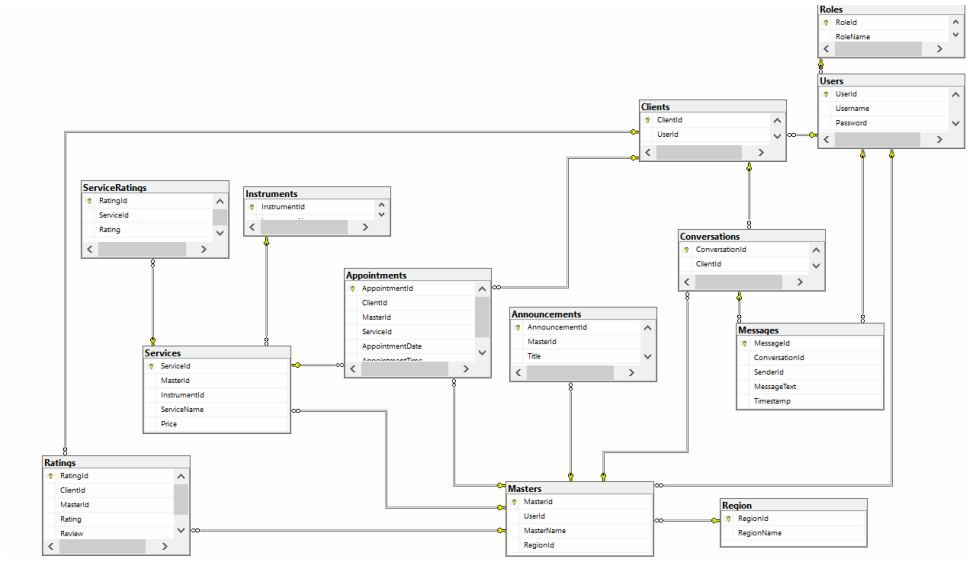

Рис. 2. Схема реляционной базы данных

Литература

1. Модели баз данных, системы управления базами данных – Электронные данные. – Режим доступа: https://www.internet-technologies.ru/articles/modeli-baz-dannyh-sistemy. – Дата доступа: 01.04.2024.
2. Введение в REST API. – RESTful веб-сервисы – Режим доступа: https://habr.com/ru/post/483202/. – Дата доступа: 03.04.2024.
3. ASP.NET Core Blazor. – Режим доступа: https://learn.microsoft.com/ru-ru/aspnet/core/-blazor/?view=aspnetcore-8.0. – Дата доступа: 01.04.2024.

ЦИФРОВОЙ ПОРТАЛ В ТЕСТИРОВАНИИ ПРИ ОБУЧЕНИИ ИНОСТРАННОМУ ЯЗЫКУ

Г. Г. Акмырадов, М. Джумадурдыев, Б. А. Ылясов, М. К. Улугов

Государственный энергетический институт Туркменистана, г. Мары

Показана роль тестирования при обучении иностранному языку на основе цифровизации образования. Рассмотрены аспекты использования цифрового портала в обеспечении эффективности и надежности оценки качества результата обучения студентов по данной дисциплине, а также в создании благоприятных условий для перехода на самостоятельное обучение.

Ключевые слова: технология, преподавание, иностранный язык, тестирование, портал цифрового образования.

Использование информационных технологий стало неизбежным в жизни каждого человека. Это также внесло большой вклад в область преподавания. Инновационные и творческие методы и приемы обучения были внедрены с использованием новых технологий в аудитории. Традиция преподавания иностранного языка радикально изменилась с появлением новых технологий, поскольку традиционного метода обучения недостаточно для эффективного преподавания иностранного языка и поддержания интереса в аудитории. Технологии предоставляют множество возможностей: сделать обучение интересным, а также сделать обучение более продуктив-