В процессе игры игрок собирает монеты в виде мошек, количество которых влияет на конечный исход игры.

В результате была разработана архитектура и прототип игрового приложения «*Echo*-10» в жанре «Конечный раннер» с использованием авторской графики в среде *Unity*.

WEB-ПРИЛОЖЕНИЕ ДЛЯ АВТОМАТИЗАЦИИ РАБОТЫ ЕГЕРСКОЙ СЛУЖБЫ ОХОТНИЧЬЕГО ХОЗЯЙСТВА

Н. В. Тишков

Учреждение образования «Гомельский государственный технический университет имени П. О. Сухого», Республика Беларусь

Научный руководитель Т. Л. Романькова

Представлены результаты разработки функциональных требований, выбора технологий реализации и обоснования архитектуры проектируемого web-приложения для проведения индивидуальных тренировок в спортивном центре.

Ключевые слова: *web*-приложение, спортивный центр, микросервисная архитектура.

Охота является одной из старейших деятельностей человека, уходящей корнями в глубокую историю. И сегодня, несмотря на развитие сельского хозяйства и промышленности, охота остаётся важной отраслью, которая не только обеспечивает регулирование популяций диких животных, но и имеет культурное и экономическое значение.

На 1 января 2023 г. численность охотников в Беларуси составила 94,6 тыс. человек, сообщили БЕЛТА в пресс-службе Министерства лесного хозяйства [1].

В центре этой отрасли стоит роль егеря. Егерь – это профессионал, специализирующийся на управлении дикими животными и контроле за охотничьими угодьями. Главная профессиональная задача егеря – выполнение биотехнических, охотхозяйственных и учетных работ на территории охотничьих угодий, охрана животных, проведение охоты, отлов и отстрел животных.

Это древняя профессия, но несмотря на это, остается неизменным тот факт, что егерям приходится выполнять большое количество монотонных действий, таких как заполнение многочисленных документов.

Данная область в наше время уже давно подвергается всевозможной автоматизации. В итоге, можно наблюдать значительный прирост эффективности работы предприятий, что увеличивает их экономическую ценность. Сложно найти аналоги в качестве *web*-приложений в Республики Беларусь, что делает разработку такого решения более востребованной.

Web-приложение сможет решать проблемы в области автоматизации не только для егеря, но и для охотоведа, а также для обычных охотников.

Для охотоведа, как одного из главнейших лиц охотхозяйства, приложение предлагает целый ряд функций, сфокусированных на эффективном управлении охотничьими угодьями и ресурсами. Оно предоставляет следующие возможности:

– составление охотничьих путевок;

– выдача разрешения на добычу охотничьего животного;

– получение отчетов и статистики;

– составление плана подкормки диких животных.

Для егеря приложение становится незаменимым инструментом в их повседневной деятельности. Ему доступны следующие функции:

– составление охотничьих путевок;
– заполнение разрешения на добычу охотничьего животного;
– составление отчетов о рейдах;
– составление отчета о проведении охоты;
– составление квартальных отчетов о подкормке диких животных.

Для обычных охотников приложение становится удобным инструментом для участия в охоте. Они могут:

– покупать охотничьи путевки;
– оплачивать охотничьи пошлины;
– общаться с егерем посредством встроенного чата;
– арендовать амуницию.

Кроме того, возможность верификации охотничьего билета обеспечивает безопасность и легальность охотничьей деятельности.

Для решения поставленной задачи по разработке *web*-приложения были выбраны две среды разработки: «*VisualStudio*» и «*WebStorm*». Такой выбор сред разработки обусловлен рядом факторов, прежде всего, оптимальным сочетанием поддержки языков программирования, которые будут использоваться в проекте.

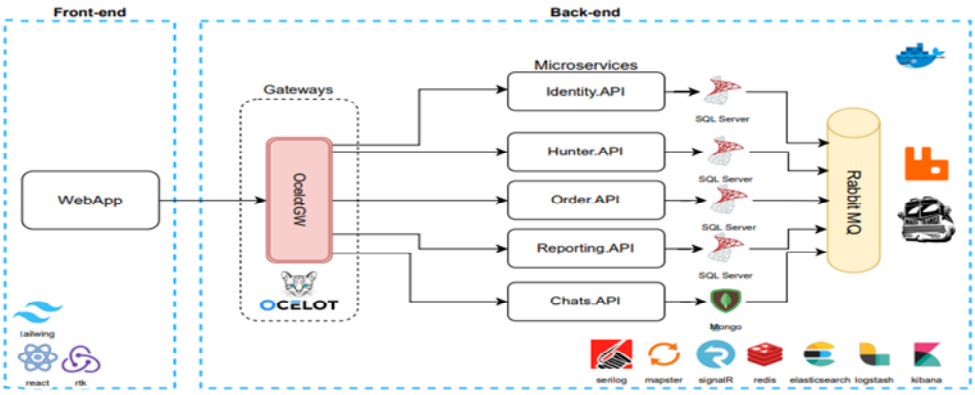

Рис. 1. Общая схема веб-приложения

Как показано на рис.1, приложение будет состоять из множества компонентов

Создание серверной части осуществляется в соответствии с принципами микросервисной архитектуры. Этот подход ориентирован на использование небольших, слабо связанных и легко изменяемых модулей – микросервисов.

Разбиение на микросервисы осуществляется с учетом синергии *Domain-Driven Design (DDD)* и бизнес-возможностей. Подход *DDD* предполагает, что каждый микросервис соответствует отдельному контексту предметной области, а контекстные границы определяются на основе бизнес-задач и ограничений. Разбиение по бизнес-возможностям простое и прямолинейное, где каждый микросервис отвечает за конкретную функциональность.

Каждый микросервис может быть реализован с использованием различных архитектурных подходов, таких как *Onionarchitecture* и *Command and Query Responsibility Segregation (CQRS)*. Данный выбор архитектур обусловлен возможностью расширения нашего *web*-приложения в будущем путем добавления нового функционала.

При использовании микросервисной архитектуры необходимо учитывать сепарацию баз данных для каждого микросервиса. Планируется использование двух типов баз данных: *MSSQL* Server и *MongoDB*, но также была учтена возможность легкой замены баз данных, так как многие базы данных имеют разные политики использования, что может быть неудобно с учетом разных факторов.

Для общения между микросервисами и репликации данных планируется использовать *messagebroker*. Один из подходящих вариантов – *RabbitMQ*, который обеспечивает буферизацию, маршрутизацию и управление очередями сообщений.

Современные подходы к написанию *front-end* приложений тоже предусматривают разработку архитектуры. К сожалению, данное направление только начало развиваться в рамках клиентских приложений относительно недавно, в отличие от *back-end*. На данный момент самой успешной архитектурой является *Feature-Sliced Design (FSD)*[2]. Это архитектурная методология для проектирования *frontend*-приложений. Проще говоря, это свод правил и соглашений по организации кода. Главная цель методологии — сделать проект понятным и структурированным, особенно в условиях регулярного изменения требований бизнеса. Проект на *FSD*, как показано на рис. 2, состоит из слоев (*layers*), каждый слой состоит из слайсов (*slices*) и каждый слайс состоит из сегментов (*segments*).

Слои стандартизированы во всех проектах и расположены вертикально. Модули на одном слое могут взаимодействовать лишь с модулями, находящимися на слоях строго ниже. На данный момент слоев семь (снизу вверх):

Shared – переиспользуемый код, не имеющий отношения к специфике приложения/бизнеса;

Entities – бизнес-сущности;

Features – взаимодействия с пользователем, действия, которые несут бизнес-ценность для пользователя;

Widgets – композиционный слой для соединения сущностей и фич всамостоятельные блоки;

Pages – композиционный слой для сборки полноценных страниц изсущностей, фич и виджетов;

Processes – сложные сценарии, покрывающие несколько страниц;

App – настройки, стили и провайдеры для всего приложения.

Затем есть слайсы, разделяющие код по предметной области. Они группируют логически связанные модули, что облегчает навигацию по кодовой базе. Слайсы не могут использовать другие слайсы на том же слое, что обеспечивает высокий уровень связности при низком уровне зацепления.

В свою очередь, каждый слайс состоит из сегментов. Это маленькие модули, главная задача которых – разделить код внутри слайса по техническому назначению.

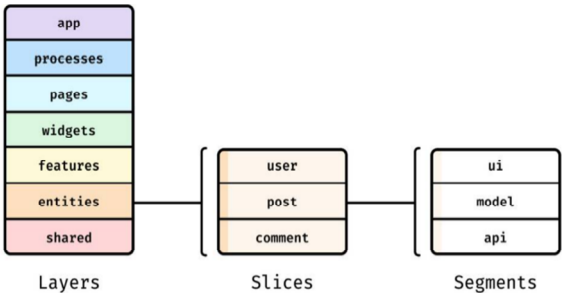

Рис. 2. Компоненты архитектуры *FSD*

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.