УЧРЕЖДЕНИЕ «МЕЖДУНАРОДНЫЙ УНИВЕРСИТЕТ КЫРГЫЗСКОЙ РЕСПУБЛИКИ»

ФАКУЛЬТЕТ ВЫСШЕЙ ШКОЛЫ БАКАЛАВРИАТА

КАФЕДРА ПРОГРАММНОЙ ИНЖЕНЕРИИ

УДК \_\_\_\_\_\_\_\_\_\_\_\_

**БЕСПАЛОВ ДАНИЛ АЛЕКСАНДРОВИЧ**

**РАЗРАБОТКА ВЕБ-СЕРВИСА ПО ПОДБОРУ СОТРУДНИКОВ**

Выпускная квалификационная работа студента направления подготовки 710400 - Программная инженерия

(академическая степень «бакалавр»)

профиль: Разработка программно-информационных систем

|  |  |
| --- | --- |
| Допущен к защите на ГАК:  Заведующий кафедрой: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (ученая степень, ученое звание) | Научный руководитель:  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (ученая степень, ученое звание) |

(подпись) (фамилия и инициалы) (подпись) (фамилия и инициалы)

Рецензент: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(ученая степень, ученое звание)

(подпись) (фамилия и инициалы)

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись) (фамилия и инициалы автора работы) «\_\_\_ » \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 202\_ г.

БИШКЕК 2024

СОДЕРЖАНИЕ

[Аннотация 3](#_Toc168313391)

[Введение 4](#_Toc168313392)

[Глава 1. АНАЛИЗ ТРЕБОВАНИЯ. 8](#_Toc168313393)

[Описание автоматизируемых бизнес-процессов 9](#_Toc168313394)

[Роль и значимость веб-сервисов в современном рекрутинге 11](#_Toc168313395)

[Анализ существующих веб-сервисов для подбора сотрудников 13](#_Toc168313396)

[Определение требований к создаваемому веб-сервису 14](#_Toc168313397)

[Пользователи и посетители 15](#_Toc168313398)

[Глава 2. ПРОЕКТНЫЕ РАБОТЫ 16](#_Toc168313399)

[Формирование структуры и функционала веб-сервиса 16](#_Toc168313400)

[Определение состава требований к разрабатываемой системе 18](#_Toc168313401)

[Описание автоматизируемых бизнес-процессов 22](#_Toc168313402)

[Определение поведения системы Skill Sale 27](#_Toc168313403)

[Выбор архитектуры системы Skill Sale 29](#_Toc168313404)

[Обзор технологий для разработки веб-приложения Skill Sale 32](#_Toc168313405)

[Конкретный выбор технологий для реализации 37](#_Toc168313406)

[Глава 3. РАЗРАБОТКА ВЕБ-СЕРВИСА 40](#_Toc168313407)

[Реализация серверных компонентов 42](#_Toc168313408)

[Создание моделей для базы данных 46](#_Toc168313409)

[Разработка front-end компоненты системы. 55](#_Toc168313410)

[Выявленные требований 56](#_Toc168313411)

[Проектирование интерфейса 57](#_Toc168313412)

[Развертывание системы Skill Sale 64](#_Toc168313413)

[Экспериментальный раздел 65](#_Toc168313414)

[Заключение 67](#_Toc168313415)

[Библиографический список 69](#_Toc168313416)

# Аннотация

Выпускная квалификационная работа содержит описание исследования «Разработка программной системы автоматизации подбора сотрудников в IT-компаниях». Программное решение позволяет автоматизировать процесс подбора сотрудников, выполнить сбор данных й в области подбора сотрудников, генерировать отчеты о деятельности, пользователей в системе и др. Система имеет трехзвенную архитектуру с разделением приложения на 3 уровня: клиент, сервер приложений и сервер баз данных. Front-end часть системы разработана в среде разработки Visual Studio с помощью технологии ASP.NET Core MVC. Для реализации back-end компоненты системы использовалась технология .Net Core. В качестве системы управления версиями базы данных использовалась система миграций Entity Framework Core. Для хранения учетных записей пользователей, информации о вакансиях и резюме использовалась база данных Microsoft SQL Server. Для поддержки кода, возможности работать над проектом нескольким разработчикам и возможности откатиться на предыдущую версию приложения использовалась система контроля версий Github.

Цель работы: Целью работы является создание эффективного программного инструмента, в виде программной системы, способной оптимизировать, автоматизировать и упростить основные процессы работы поиска подходящих кандидатов на должности. Улучшить оперативность принятий решений при выборе и дать возможность кандидатам заявить о себе как о наиболее конкурентноспособном сотруднике.

Работа состоит из введения, главы анализа предметной области, главы проектирования веб-сервиса, главы разработки, заключения, библиографического списка и приложений.

Глава анализа описывает рассуждения автора о предметной области, конкретно – о подборе сотрудников. Так же содержит сравнительный анализ существующих программных продуктов в области темы выпускной работы, выявление, составление требований к системе, а также описание основных бизнес-процессов, которые система будет автоматизировать.

Глава проектирования заключает выбор технологий, применяемых для разработки системы, проектированию концептуальной модели системы, описанию поведения системы, проектированию классов системы, построению и тестированию моделей данных веб-сервиса.

Глава разработки содержит информацию о выборе и применении архитектуры системы и конечно о реализации front-end и back-end частей приложения, как основной составляющей. Здесь также описывается процесс тестирования системы и развертывания на серверах компании somee для возможности подключения к веб-сервису из глобальной сети internet.

# Введение

Для написания выпускной квалификационной работы, необходимым является определение главных задач, которые будут достигнуты в процессе. Так как многие задачи вытекают из предыдущих, начнем с малого, определим нашу очевидно главную цель. Главная задача данной работы, о чем свидетельствует название, заключается в разработке программного продукта, способного работать независимо. Из чего следует, то, что нам нужно разобраться в том, что такое вообще мы будем разрабатывать. Всем известный поисковик трактует задачу как: «Подбор персонала — специализированная деятельность HR предназначенная для поиска и подбора кандидатов на вакантные позиции в организации или компании. Эта деятельность может осуществляться как HR-менеджерами или специалистами по рекрутменту в штате организации, так и непосредственно линейными руководителями, например, в малом бизнесе.». Стало быть, подбор персонала — это профессия определенного круга людей. На современном рынке труда существует инфраструктура рекрутмента, которая включает специализированные агентства по поиску и подбору кандидатов - рекрутинговые агентства, предоставляющие платные услуги работодателю, и агентства по трудоустройству, предоставляющие платные услуги для кандидатов. В инфраструктуру также входят профессиональные социальные сети и специализированные интернет-сайты по поиску работы.

Любая организация имеет перспективу роста, любые изменения ее деятельности открывают возможности сотрудникам, наиболее развитым в различных сферах деятельности, принять участие в ее развитии. Что становится одной из причин возникновения вакантных мест, коих очень много: текучка кадров, некомпетентность сотрудников и другие. Целью рекрутмента является своевременное обеспечение организации необходимым количеством и качеством персонала, следовательно спрос на такие агентства постоянно имеет рост.

В настоящее время любая организация заинтересована в поиске и удержании высококвалифицированных кадров. Именно поэтому процесс найма и отбора персонала на вакантные должности является одним из важнейших вопросов и первостепенной задачей в системе управления персоналом. Привлечение в организацию потенциальных кандидатов является одной из жизненно важных функций системы управления человеческими ресурсами организации, поскольку оказывает самое непосредственное влияние на будущее компании.

Доски объявлений и карьерные сайты являются центрами для квалифицированных кандидатов. Эти цифровые рынки сотрудников позволяют рекрутерам находить подходящих кандидатов, создавая специальные объявления о вакансиях, в которых излагаются минимальные требования для каждой должности и сопоставляются компании с подходящими кандидатами.

Некоторые карьерные веб-сайты используют технологию искусственного интеллекта (ИИ), чтобы сделать процесс найма более эффективным. Эти инструменты искусственного интеллекта сканируют, анализируют и сопоставляют резюме квалифицированных кандидатов с объявлениями о вакансиях, чтобы обеспечить точность и возможность подать заявку только тем кандидатам, которые предлагают необходимые навыки и образование для работы.

Иногда описание работы может показаться потенциальным кандидатам устаревшим или неинформативным. Для рекрутеров важно просмотреть описания вакансий и убедиться, что описание отражает необходимые навыки и требования к образованию для должности. Например, если у вас есть описание работы, которое гласит: «Квалифицированный кандидат должен управлять оборудованием и работать для выполнения ежедневных норм», потенциальный кандидат может воспринять эту информацию как слишком расплывчатую.

Вместо этого вы можете попробовать сформулировать описание следующим образом: «Ищу квалифицированного оператора станка для обслуживания оборудования, выполнения производственных квот, проведения проверок безопасности, управления оборудованием вместе с другими членами команды и консультирования по вопросам безопасной эксплуатации оборудования». В этом описании работы излагаются некоторые ключевые обязанности, которые выполняет кандидат, вместо общего описания работы. Предоставляя конкретную информацию, вы можете помочь кандидатам почувствовать себя более уверенными в своей квалификации и лучше понять, что они могут делать на новой должности.

Несмотря на ключевую роль мотивации в системе управления персоналом, многие руководители и специалисты по работе с кадрами часто недооценивают её значение. Это, в свою очередь, приводит к снижению эффективности деятельности предприятия и уменьшению прибыли. Однако даже при наличии мотивационной системы в организации может наблюдаться низкий уровень мотивации сотрудников. Часто это обусловлено тем, что при разработке системы мотивации не учитываются специфические особенности деятельности организации и основные мотивы её сотрудников. Недостаточное понимание мотивационных механизмов и неспособность эффективно их применить внутри рабочего коллектива могут сделать мотивационную систему неэффективной и существенно снизить трудовую активность работников. Во многих случаях мотивация сотрудников сводится лишь к материальному вознаграждению, то есть выплате заработной платы. При этом работодатели считают, что этого достаточно для активного выполнения работниками своих обязанностей и полного удовлетворения от работы. Это мнение широко распространено среди большинства работодателей, но оно ошибочно, так как одно лишь денежное вознаграждение не гарантирует постоянную заинтересованность работника в своей работе. Даже при стабильной заработной плате уровень мотивации сотрудника может оставаться низким, что может быть обусловлено множеством причин.

В настоящее время взаимодействие между кандидатом и потенциальным работодателем осуществляется следующим образом: кандидат представляет свое резюме, в котором описаны его личные характеристики, навыки и квалификация, в то время как работодатель публикует вакансию на требуемую в компании должность. Для размещения вакансий и резюме существуют специализированные онлайн-платформы, предназначенные для поиска работы. В рамках этих платформ соискатели изучают предлагаемые должности, а работодатели исследуют кандидатов. Этот этап является наиболее трудоемким и требует значительного времени из-за необходимости анализа большого числа кандидатов и соотнесения их способностей с требованиями конкретной должности. Дополнительной проблемой является множество интернет-порталов, занимающихся поиском сотрудников, что повышает вероятность упущения подходящего кандидата или дублирования записей. Отбор проводится на основе субъективного мнения работодателя о характеристиках кандидата и его опыта, что приводит к принятию решения на интуитивном уровне.

Таким образом, актуальность данной темы обусловлена отсутствием универсального программного решения, которое позволило бы ускорить процесс поиска и подбора сотрудников среди множества информационных ресурсов на основе сопоставления данных. Это привело бы к сокращению времени, затрачиваемого на поиск кандидатов, и повышению эффективности подбора персонала. Современные инструменты, такие как доски объявлений, карьерные сайты и ИИ (Искусственный интеллект), имеют решающее значение для процесса найма, потому что компании теперь проводят большую часть операций по найму онлайн. Многие люди имеют онлайн-представительство, и старый метод заполнения бумажных заявлений становится неактуальным. Использование современных инструментов позволяет компании связываться с нужными кандидатами на часто используемых платформах, помогает уменьшить географические ограничения и оптимизировать процесс в целом.

Объектом исследования является процесс подбора персонала с проверкой характеристик кандидата на соответствие заявленным требованиям, предмет исследования – средства автоматизации данного процесса.

Цель выпускной квалификационной работы: разработка программной системы, позволяющей выполнять поиск и подбор персонала путем мониторинга онлайн-порталов и сопоставления характеристик кандидата и требований, заявленных в вакансии. Разрабатываемой системе присваивается название Skill Sale.

* Для достижения поставленной цели были определены следующие задачи:
* Рассмотрение существующих программных решений, выполняющих автоматизированный поиск сотрудников, выявление достоинств и недостатков рассматриваемых решений.
* Выявление и формирование требований к системе, автоматизирующей подбор персонала в IT-компаниях.
* Описание бизнес-процессов, которые будет автоматизировать разрабатываемая система.
* Проектирование системы, выбор технологий для реализации системы.
* Реализация системы Skill Sale.
* Тестирование и оценка качества разработанной системы.
* Написание документации для разработанной системы.

Степень разработанности темы автоматизации подбора персонала путем мониторинга достаточно высока, что подтверждается существованием большого числа HRM‑систем (Human Resources Management) для управления трудовыми ресурсами, которые зачастую не обладают возможностью анализа вакансий, а также систем мониторинга онлайн-порталов, которые не предусматривают автоматизацию процессов управления персоналом. Следовательно, необходимость создания универсального, гибкого, быстрого и качественного программного продукта существует до сих пор.

Методы исследования:

* Анализ собранных материалов по теме исследования.
* Методы объектно-ориентированного проектирования и программирования.
* Методы анализа и сбора данных.
* Методы проектирования, обучения и тестирования нейронных сетей.
* Методы гибкой разработки web-проектов.
* Методы компонентного и модульного тестирования.

Практическая значимость: реализуемое приложение может значительно упростить процесс поиска сотрудников и работы. Программное решение должно будет быть развернуто на локальных серверах, в случае успешной эксплуатации систему можно будет распространять в глобальную сеть.

В первой главе описывается процесс анализа предметной области подбора персонала, а также выявляются основные требования к разрабатываемой системе и описываются процесс формализации данных требований и автоматизируемые бизнес-процессы.

# Глава 1. АНАЛИЗ ТРЕБОВАНИЯ.

Процесс анализа предметной области в разработке информационных систем предполагает выделение основных и вспомогательных бизнес-процессов, которые призваны обеспечить производство продукта/услуги. Но, наряду с этим, выделение и рассмотрение бизнес-процессов предоставляет возможность определиться с бизнес-элементами и структурами данных, которые должны участвовать в обработке данных. Такие возможности требуют от разработчика информационной системы в моделировании базы данных отталкиваться не только от документов, используемых в деятельности предметной области, но и окружения каждого бизнес-процесса и функций, включающего определение бизнес-элементов, объектов данных, исполнителей обработки, владельцев процессов и функций, предшествующих и последующих функций, инициирующих и результирующих событий, прочие элементы. Глубина рассмотрения бизнес-процессов и функций дает максимально полную информацию о процессах, происходящих в предметной области, и позволяет лучше понимать задачи, которые необходимо реализовать при разработке базы данных, к которым относятся моделирование структуры базы данных, определение правил ссылочной целостности, формирование процедур обработки и представления данных, но запросам пользователей.

Один из современных подходов к разработке информационных систем основывается на выделении документарных элементов в бизнес-процессе, где атрибутами являются сущности (объекты) предметной области, впоследствии представляемые сущностями логической модели базы данных. Этот подход может быть применим только на уровне контекстного рассмотрения бизнес-процессов и проведения операций декомпозиции бизнес-процессов, по, при переходе па уровень моделирования потоков данных смысл использования документарного подхода теряется, поскольку сама суть моделей потоков данных предполагает работу с объектами данных, некоторые из которых могут представляться в предметной области виртуальными представлениями и не существовать в документах, а также по причине отсутствия необходимости, кроме отдельных частных случаев, хранения информации о документах, которые сами по себе не являются объектами предметной области.

Цель анализа предметной области процесса подбора сотрудников заключается в выявлении потребностей пользователей разрабатываемой информационной системы. Основные задачи анализа:

* изучить основные понятия, связанные с подбором персонала, разобраться в основных терминах данной предметной области, рассмотреть современные методики рекрутмента;
* провести анализ существующих приложений данной предметной области, выявить их положительные и отрицательные стороны;
* выявить основные требования к разрабатываемой программной системе, формализовать их с помощью диаграммы прецедентов;
* описать автоматизируемые бизнес-процессы в области подбора персонала с помощью UML диаграмм активностей.

## Описание автоматизируемых бизнес-процессов

Прежде всего, для автоматизации бизнес-процесса необходимо его изучение. В данном разделе для повышения наглядности и упрощения анализа бизнес-процесса будет использовано моделирование бизнес-процесса. Моделирование бизнес-процессов (БП) – это построение и изучение бизнес-процессов какой-либо организационной единицы (организации или ее отдела). Является самостоятельной дисциплиной, с помощью которой можно описать предметную область с целью ее анализа и изменения, поиска моментов, которые можно оптимизировать. Кроме того, моделирование БП сегодня часто входит в состав процесса разработки программного обеспечения (ПО). Так, моделирование БП используется для определения автоматизируемых процессов и операций, а также для определения требований к ИС. Моделирование БП практически реализуется в специализированных графических языках, называемых нотациями. Как правило, нотации состоят из блок-схем и правил представления их компонентов для описания исследуемого объекта. Выбранный бизнес-процесс найма на работу нового сотрудника состоит из следующих этапов. Кандидаты откликаются на вакансию через сайт «hh.ru», сайт организации или через почту, отправляя свои резюме. Также кандидат может прийти в компанию лично. Далее происходит первичный отсев кандидатов. Если кандидат подходит, то начинается переписка. Дальше выдается тестовое задание, и происходит его проверка специалистами и в зависимости от результатов его выполнения назначается или нет техническое собеседование, по результатам которого принимается решение о найме сотрудника.

Размещение вакансии: Процесс начинается с создания и размещения вакансии на веб-платформе или специализированном веб-сервисе по подбору сотрудников. Вакансия содержит информацию о требуемой должности, квалификационных требованиях, обязанностях и других ключевых аспектах работы.

Поиск кандидатов: Веб-сервис автоматически анализирует базу данных резюме и профилей пользователей, используя различные критерии, такие как навыки, опыт работы, образование и географическое расположение. Это позволяет выявить наиболее подходящих кандидатов для данной вакансии.

Собеседование и выбор кандидатов: после предварительной оценки кандидатов проводятся собеседования с наиболее перспективными кандидатами. Веб-сервис может также помочь в организации и планировании собеседований, обмене обратной связью и принятии решения о найме.

Отслеживание процесса найма: Веб-сервис предоставляет инструменты для отслеживания и управления всем процессом найма, включая обновление статуса кандидатов, взаимодействие с ними и хранение всех необходимых данных и документов.

Разбирая современные тенденции в подборе сотрудников, можно выделить несколько ключевых аспектов. Первый из них — это рост значимости цифровых технологий. С развитием интернета и цифровизации бизнес-процессов, компании все больше обращаются к онлайн-платформам для поиска персонала. Это позволяет им достигать более широкой аудитории соискателей и ускорять процесс подбора. Далее, мы видим рост востребованности алгоритмов искусственного интеллекта и аналитических инструментов. Компании используют алгоритмы машинного обучения для автоматического анализа резюме и выявления наиболее подходящих кандидатов. Это помогает сэкономить время рекрутеров и повысить точность отбора.

Также наблюдается усиление фокуса на кандидатском опыте. Стремление к созданию позитивного и эффективного опыта для кандидатов становится важным элементом подбора. Компании уделяют больше внимания процессу общения с соискателями, отклику на их запросы и предоставлению обратной связи.

В последнее время мы также видим растущую значимость разнообразия и инклюзивности в подборе персонала. Компании стремятся создать более разнообразные и инклюзивные команды, что не только способствует улучшению рабочей атмосферы, но и повышает креативность и инновационность.

Наконец, стоит упомянуть об увеличении гибкости и мобильности в трудовых отношениях. Рост числа фрилансеров и специалистов на удаленной работе создает потребность в адаптации методов подбора и управления персоналом для работы с такими формами занятости.

Эти тенденции представляют собой важные вызовы и возможности для разработки веб-сервиса по подбору персонала. Веб-платформа, учитывающая эти аспекты, будет успешно конкурировать на рынке и отвечать на современные потребности бизнеса в подборе квалифицированных сотрудников.

## Роль и значимость веб-сервисов в современном рекрутинге

ИТ рекрутинг претерпел значительное развитие в последние годы. Благодаря постоянному развитию технологий, рекрутеры имеют доступ к более широкому спектру методов поиска и оценки кандидатов. Они могут использовать онлайн-платформы, профессиональные сети и специализированные ИТ рекрутинговые ресурсы для поиска подходящих специалистов. Технологии, такие как искусственный интеллект и машинное обучение, также нашли применение в ИТ рекрутинге, упрощая и автоматизируя процессы подбора персонала.

Влияние ИТ рекрутинга на рост отрасли информационных технологий не может быть недооценено. Способность компаний находить, нанимать и задерживать талантливых ИТ специалистов является ключевым фактором в достижении успеха в современном цифровом мире. Эффективный ИТ рекрутинг позволяет организациям привлекать высококвалифицированных специалистов, которые в свою очередь способствуют инновациям, развитию продуктов и конкурентоспособности компании. В современном рекрутинге веб-сервисы играют ключевую роль, выступая важным инструментом для успешного подбора персонала.

Их роль и значимость в этой области обусловлены несколькими факторами:

* Широкий доступ к кандидатам
* Автоматизация процесса подбора
* Точность и надежность отбора
* Улучшение кандидатского опыта
* Аналитика и отчетность
* Исследование потребностей рынка в сфере подбора персонала

Веб-сервисы обеспечивают широкий доступ к базам данных резюме и профилей кандидатов. Это позволяет компаниям быстро и эффективно находить кандидатов, соответствующих их требованиям.

Веб-сервисы позволяют автоматизировать множество этапов процесса подбора, начиная от размещения вакансий и фильтрации резюме, и заканчивая оценкой кандидатов и проведением собеседований. Это значительно сокращает время, затрачиваемое на подбор персонала, и улучшает его эффективность.

Благодаря использованию алгоритмов машинного обучения и искусственного интеллекта, веб-сервисы способны проводить более точный и надежный отбор кандидатов. Они автоматически анализируют резюме, выявляют наиболее подходящих кандидатов и предлагают рекомендации по их выбору.

Веб-сервисы помогают улучшить опыт соискателей, предоставляя им удобные инструменты для поиска вакансий, заполнения анкет и отправки резюме. Это способствует улучшению репутации компании и привлечению квалифицированных кандидатов.

Многие веб-сервисы предоставляют инструменты для анализа данных о подборе персонала и составлении отчетов о его эффективности. Это позволяет компаниям оценить результативность своих рекрутинговых стратегий и вносить необходимые коррективы.

В целом, веб-сервисы играют важную роль в современном рекрутинге, обеспечивая компаниям доступ к широкой базе потенциальных кандидатов, автоматизацию процесса подбора, повышение его эффективности и улучшение опыта как для работодателей, так и для соискателей. Они являются неотъемлемой частью современных рекрутинговых стратегий и способствуют успешному привлечению и найму квалифицированных специалистов.

Исследование потребностей рынка в сфере подбора персонала является важным этапом для успешной разработки веб-сервиса. Анализируя потребности рынка, можно выявить основные требования и ожидания компаний и соискателей, что позволит создать продукт, наиболее точно отвечающий их потребностям.

На текущем рынке труда можно наблюдать ряд изменений и тенденций, влияющих на процесс подбора персонала. Одной из них является увеличение конкуренции среди компаний за талантливых специалистов. В условиях острой конкуренции работодатели стараются предлагать более привлекательные условия труда, что включает как финансовые бонусы, так и различные льготы, и возможности для профессионального и личностного роста.

Еще одной значимой потребностью рынка является необходимость быстрого и эффективного подбора персонала. С развитием технологий и увеличением объемов информации процесс подбора персонала становится все более сложным и объемным. Компании ищут инструменты, которые помогли бы им оптимизировать этот процесс и сократить время на его осуществление.

Важной потребностью считается также поддержание качественного и разнообразного состава персонала. Компании стремятся создать балансированные и инклюзивные команды, которые могли бы эффективно решать разнообразные задачи и стимулировать инновационные решения. Поэтому важно, чтобы веб-сервис предоставлял инструменты для привлечения кандидатов с различным опытом, культурным и профессиональным фоном.

Кроме того, на рынке существует растущая потребность в гибких формах трудоустройства. Специалисты все чаще предпочитают гибкий график работы, возможность удаленной работы или работу на проектной основе. Поэтому веб-сервис должен учитывать и такие аспекты, предлагая возможности для поиска таких вакансий и поддержки гибких форм трудоустройства.

Исследование этих потребностей рынка позволит выявить основные требования и ожидания как со стороны компаний, так и со стороны соискателей. Это поможет создать веб-сервис, который будет максимально соответствовать потребностям и ожиданиям всех заинтересованных сторон.

## Анализ существующих веб-сервисов для подбора сотрудников

При анализе существующих веб-сервисов для подбора сотрудников можно выделить ряд интересных особенностей и характеристик.

LinkedIn, например, является одним из наиболее популярных и широко используемых сервисов для поиска сотрудников. Его преимущества включают обширную базу профилей профессионалов со всего мира, продвинутый инструментарий для поиска и фильтрации кандидатов, а также возможность проведения рекрутинговых мероприятий и создания бренда работодателя. Однако, недостатком LinkedIn может быть высокая конкуренция и сложность выделения своей компании среди множества других работодателей.

Glassdoor предоставляет не только возможность поиска вакансий, но и обзоры о компаниях, зарплатные ожидания и информацию о культуре организации. Это позволяет кандидатам получить более полное представление о потенциальном работодателе. Однако, на Glassdoor могут присутствовать отзывы, которые не всегда отражают реальное положение дел в компании, что может искажать восприятие.

AngelList специализируется на поиске работы в стартапах и предоставляет возможность не только найти вакансии, но и привлечь инвестиции для своего проекта. Это может быть полезно для специалистов, заинтересованных в работе в инновационной среде и в поиске инвестиций для своих собственных проектов. Однако, ограничение по целевой аудитории может быть недостатком для более широкого круга работодателей и соискателей.

Платформа Indeed также предоставляет широкий выбор вакансий и простой в использовании интерфейс. Однако, качество вакансий может быть неоднородным, и некоторые из них могут быть устаревшими или неактуальными. Кроме того, ограниченный функционал по сравнению с другими сервисами может быть недостатком для некоторых пользователей.

Общим для многих существующих веб-сервисов может быть также высокая стоимость использования для работодателей, особенно для малых и средних компаний, что может стать препятствием для широкого использования сервисов.

Таким образом, каждый из существующих веб-сервисов для подбора сотрудников имеет свои сильные и слабые стороны, и выбор определенного сервиса зависит от конкретных потребностей и целей как работодателей, так и соискателей.

## Определение требований к создаваемому веб-сервису

При определении требований к создаваемому веб-сервису для подбора сотрудников необходимо учитывать ряд ключевых аспектов, которые позволят создать продукт, наиболее точно отвечающий потребностям рынка и пользователей.

Во-первых, важно определить функциональные требования к веб-сервису. Это включает в себя определение основных возможностей, которые должен предоставлять сервис, таких как возможность размещения вакансий, поиск и фильтрация кандидатов, проведение собеседований, а также аналитика и отчетность по результатам подбора.

Далее, необходимо учитывать технические требования к веб-сервису. Это включает выбор технологического стека для разработки, учет требований к безопасности и защите данных, а также обеспечение масштабируемости и производительности сервиса.

Также важно учитывать пользовательские требования к веб-сервису. Это включает в себя создание удобного и интуитивно понятного интерфейса, адаптированного под различные устройства и типы пользователей, а также предоставление дополнительных возможностей для улучшения кандидатского опыта и удовлетворения потребностей работодателей.

Кроме того, важно учитывать требования к поддержке и обслуживанию веб-сервиса. Это включает в себя обеспечение надежности и доступности сервиса, оперативное реагирование на обратную связь и запросы пользователей, а также регулярное обновление и совершенствование функционала и интерфейса.

Определение требований к создаваемому веб-сервису для подбора сотрудников требует внимательного анализа потребностей рынка и пользователей. Важно учитывать функциональные, технические, пользовательские и требования к поддержке и обслуживанию.

Функциональные требования определяют основные возможности сервиса, такие как размещение вакансий, поиск и фильтрация кандидатов, проведение собеседований и аналитика результатов подбора. Технические требования включают выбор технологического стека, обеспечение безопасности и масштабируемости. Пользовательские требования касаются создания удобного интерфейса, адаптированного под различные устройства и типы пользователей. Требования к поддержке и обслуживанию включают надежность, доступность и оперативное обновление функционала.

## Пользователи и посетители

Можно выделить два типа пользователей: обычные посетители и пользователи системы. Обычные посетители, запрашивают малое количество страниц и остаются на них в течение короткого промежутка времени. Такие посетители меньше всего заинтересованы в информационном наполнении сайта. Но их можно заинтересовать изображениями, дизайном, различными элементами управления и персонализированным подходом. Пользователи запрашивают большее число страниц и остаются на сайте в течение большего количества времени, чем посетители. Обладают меньшим интересом к изображениям и дизайну, их более интересует содержимое сайта. Они 13 могут завести себе учётную запись на сайте и использовать преимущества персональной панели. Для успешной коммерциализации проекта важен любой пользователь. Важно уметь заинтересовать каждого. Наиболее эффективный способ этого достигнуть – создать привлекательный интерфейс пользователя. Когда пользователя привлекает интерфейс, он тратит больше времени на просмотр сайта, тем самым растут шансы его заинтересовать.

# Глава 2. ПРОЕКТНЫЕ РАБОТЫ

## Формирование структуры и функционала веб-сервиса

При формировании структуры и функционала веб-сервиса для подбора сотрудников важно учесть целевую аудиторию и основные цели использования платформы. Например, если это сервис для небольших и средних предприятий, функционал должен быть интуитивно понятным и легко настраиваемым. Если же это решение для крупных корпораций, важно обеспечить масштабируемость и высокую степень персонализации. Целевая аудитория веб-сервиса для подбора сотрудников Skill Sale включает в себя как работодателей, так и соискателей. Определение терминов можно прочитать ниже.

Работодатели — это компании и организации, которые ищут новых сотрудников для своих вакансий. В основном, это HR-менеджеры, рекрутеры и бизнес-владельцы, ответственные за найм персонала. Они ищут эффективные и удобные инструменты для размещения вакансий, фильтрации кандидатов, проведения собеседований и управления рекрутинговыми процессами.

Соискатели — это люди, которые ищут новые возможности для работы и профессионального роста. Это могут быть специалисты различных профессиональных областей, студенты, выпускники учебных заведений и другие люди, ищущие работу. Они ищут удобные инструменты для поиска вакансий, создания и редактирования своего профиля, а также взаимодействия с работодателями и другими соискателями.

Основная цель любой платформы для подбора сотрудников заключается в том, чтобы помочь работодателям найти лучших кандидатов для своих вакансий. На платформе должны быть предусмотрены инструменты для размещения вакансий, проведения поиска и фильтрации кандидатов, а также возможности для проведения собеседований и оценки квалификации кандидатов. Основной аргумент для этой цели - повышение эффективности и результативности рекрутинговых процессов, что в конечном итоге приводит к увеличению производительности и успеху компании.

Вторая важная цель платформы - помочь компаниям оптимизировать рекрутинговые процессы, уменьшив время и затраты, необходимые для найма сотрудников. Путем использования современных технологий и инновационных методов подбора персонала платформа может существенно сократить время, затрачиваемое на поиск и оценку кандидатов, что в свою очередь позволит компаниям быстрее реагировать на изменения в рыночных условиях и улучшать свою конкурентоспособность.

Третья цель платформы - быть надежным источником информации о вакансиях и компаниях для соискателей. Важно предоставлять детальные описания вакансий, полезные сведения о компаниях, а также отзывы о работодателях и зарплатные ожидания. Это позволит соискателям принимать более обоснованные решения о своей карьере и выборе работодателя.

Еще одна важная цель платформы - создать удобный и привлекательный пользовательский опыт для всех пользователей. Это включает в себя удобный интерфейс, простоту использования сервиса, быстрый доступ к нужной информации и возможность взаимодействия с другими пользователями. Такой подход сделает использование платформы приятным и продуктивным для всех ее участников.

И последняя, но не менее важная цель платформы - помочь соискателям в их профессиональном развитии и карьерном росте. Платформа может предоставлять дополнительные ресурсы и инструменты для помощи соискателям в поиске работы, развитии их профессиональных навыков и построении успешной карьеры. Это может быть обучающие материалы, курсы повышения квалификации, советы по построению карьеры и возможности для профессиональной сети.

В структуре веб-сервиса должны быть ясно выделены разделы для работодателей и соискателей. Для работодателей важно предоставить возможность размещения вакансий, фильтрации и анализа кандидатов, а также инструменты для проведения собеседований и управления рекрутинговыми процессами. Для соискателей необходимо предусмотреть удобный поиск вакансий, возможность создания и редактирования профиля, а также получение уведомлений о новых вакансиях и статусе их заявок.

Функционал веб-сервиса может также включать возможности для взаимодействия пользователей, например, комментарии и отзывы о компаниях, обмен сообщениями между работодателями и соискателями, а также функции социальной сети для создания профессиональных связей.

Важно также уделить внимание аналитике и отчетности. Веб-сервис должен предоставлять инструменты для анализа эффективности рекрутинговых кампаний, оценки качества подбора персонала и выявления трендов на рынке труда. Это поможет работодателям и рекрутерам принимать обоснованные решения и оптимизировать свои стратегии подбора.

Кроме того, важно учесть требования к безопасности данных и защите персональной информации пользователей, а также обеспечить надежную работу сервиса и его доступность в любое время.

В итоге, формирование структуры и функционала веб-сервиса должно быть направлено на создание удобного и эффективного инструмента для подбора сотрудников, который будет соответствовать потребностям как работодателей, так и соискателей, и помогать им добиваться своих целей в области рекрутинга.

## Определение состава требований к разрабатываемой системе

Очевидно, что каждый разработчик, начиная создавать проект желает сделать его максимально качественно. Но без определенного заранее плана сделать это весьма затруднительно. Следовательно, нужно определить состав требований к конечному продукту. Требования должны описывать каждое ответвление программы, точно и развернуто их раскрывать. Требования в большей степени зависят от анализа предметной области, который мы благополучно составили в предыдущей главе. Итак, основываясь на предыдущем анализе, мы должны максимально полно и четко сформулировать каждый критерий требований. Критерии должны политься на Функциональные и Нефункциональные, а также не менее важно отвести отдельную часть для проработки деятельности отдельных групп пользователей на веб-сервисе.

Для начала определим Функциональные требования. В этом пункте мы определим главные возможности программы с точки зрения пользователя. В нашем случае это:

Функциональные требования:

* Добавление вакансий в систему.
* Добавление резюме в систему.
* Регистрация / авторизация пользователей в системе.
* Наследование пользовательских ролей в системе Техником, на роли выявленные в анализе: администратора, пользователя системы (зарегистрирован), и конечно, пользователя-гостя системы (не зарегистрирован).

Итак, с функциональными требованиями разобрались, теперь, по плану, определим общие нефункциональные предпочтения. Именно предпочтения, так как тут у нас есть определенный выбор, и мы можем положиться на свой вкус, особенности и конечно же анализ.

Нефункциональные требования:

* Разрабатываемая система должна быть веб-приложением, причиной этому задание на данную работу.
* Back-end часть системы должна быть реализована на более подходящем нам языке программирования, определяется скорее опытным путем.
* Front-end часть системы должна быть реализована при помощи архитектур, с использованием шаблонов и компонентов, которые будут взаимодействовать с бэкендом.
* Система должна иметь удобный, интуитивно понятный, а главное простой пользовательский интерфейс.
* Источником собираемых данных будут сами пользователи системы, база данных будет заполняться из форм создания вакансий и резюме, через специализированные страницы веб-сайта.
* Система должна работать стабильно, без сбоев, особенно важно, чтобы система была доступна из глобальной сети интернет.
* Собранные данные должны храниться в базе данных.
* Интерфейс должен быть на русском языке, в будущем в большой долей вероятности будет доступен выбор языка, для внедрения системы в других странах.
* Система должна поддерживать возможность обновление информации о кандидатах и их резюме по запросу.

Выше мы определили основные требования, следующим шагом определим действия пользователей на сайте.

Чтобы продумать действия пользователя на веб-сервисе, для начала нужно стать тем самым рядовым пользователем системы. Представим, что мы в поиске работы и попали на первый попавшийся сайт по поиску работы. Зададим себе вопрос: что мы видим? Главная страница сайта. Страницы отображения списка вакансий и резюме. При посещении сайта нужно обязательно дать возможность гостям системы просматривать дозволенную им информацию без необходимости создания аккаунта. Роль Гость системы очень важна, к ним нужно относиться максимально лояльно, чтобы не спугнуть потенциального пользователя излишним заполнением информации. Используя диаграммы прецедентов, удалось создать схему, на которой видно разделение возможностей каждой из ролей и соответствующие им возможности, подробнее на рисунке 2.1. Диаграмма прецедентов подразделяет пользователей на акторов: Гость, Пользователь, Администратор (Техник).

Главные прецеденты: Идентификация, Регистрация, Заполнение данных о кандидатах, Создание вакансии.

Примыкающие прецеденты: Учет данных в системе, Импорт данных, Создание резюме, Создание резюме кандидата, Изменение статуса кандидата.

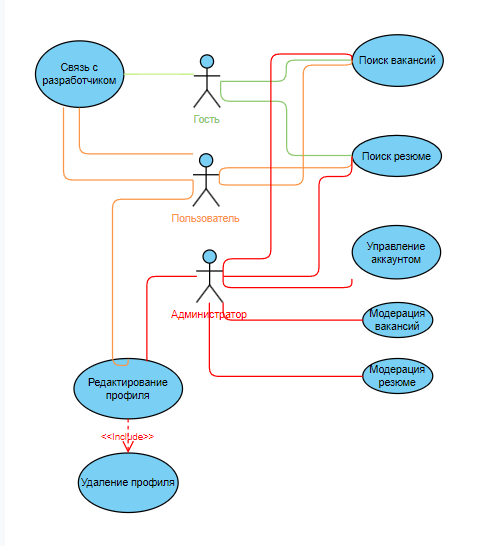


Рисунок 2.1. Диаграмма прецедентов системы Skill Sale

Определим прецедент идентификации. Присвоенное название –«Аутентификация пользователя». Действующее лицо - Пользователь. Прецедент характеризует идентификацию в системе. Условия срабатывания – исправно работает веб-сайт, открыта страница входа.

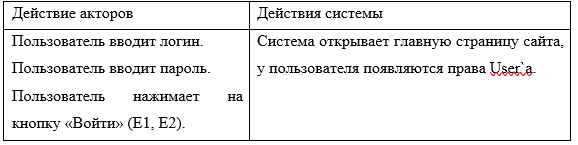


Рисунок 2.2. Прецедент «Идентификация»

Альтернативные пути развития событий:

1) Если логин или пароль не введены, то выводится сообщение «Пропущены логин или пароль», система переводит пользователя к пункту 1.

2) Если пользователя с введенным логином не существует в базе данных, или введенный пароль не соответствует паролю указанного пользователя, то выводится сообщение «Логин или пароль заполнены некорректно», система переводит пользователя к пункту 1.

Определим прецедент Регистрации. Взаимодействующее лицо – гость системы. Возможные действия – регистрация в системе. Условия срабатывания – исправно работает веб-сайт, текущая страница регистрации.

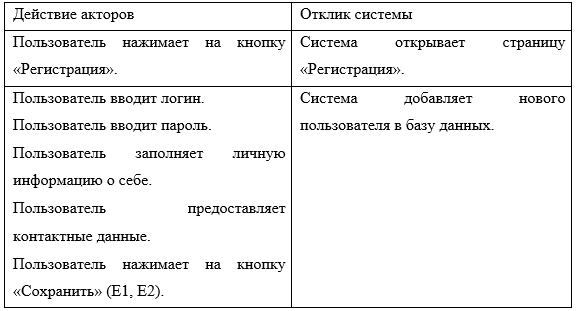


Рисунок 2.3. Прецедент «Регистрация»

Альтернативные пути развития событий:

1) Если обязательные поля ввода не заполнены или заполнены некорректно, система выводит сообщение «Данные заполнены некорректно», и возвращает пользователя к шагу 3.

2) Если пользователь с таким логином уже зарегистрирован в системе, то система выведет сообщение «Такой пользователь в системе уже есть. Логин пользователя должен быть уникальным». Возврат к позиции 3.

Определим прецедент создания вакансии. Действующее лицо – Пользователь, Техник. Акторы создают новую вакансию, запись в системе. Условия срабатывания - Пользователь идентифицирован и авторизован, открыта страница создания вакансии.

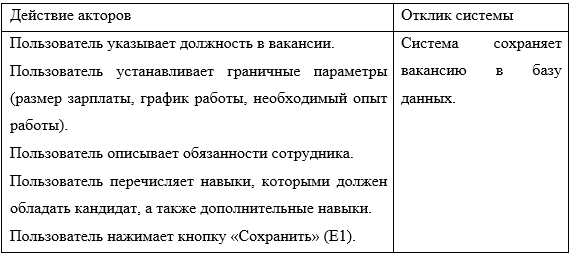


Рисунок 2.4. Прецедент «Создание вакансии»

Альтернативные пути развития событий:

1) Если введенные данные заполнены некорректно, система выводит соответствующее сообщение, и возвращает пользователя к шагу 1.

## Описание автоматизируемых бизнес-процессов

Автоматизация бизнес-процессов играет ключевую роль в современных организациях, стремящихся повысить свою эффективность, снизить затраты и улучшить качество обслуживания клиентов. Описание автоматизируемых бизнес-процессов является важным шагом в этом направлении, так как оно позволяет четко определить, какие задачи могут быть улучшены с помощью технологий, и выбрать наиболее подходящие методы для их автоматизации. Обычно, процесс описания начинается с идентификации текущих операций и задач, которые выполняются вручную. Этот этап включает в себя сбор данных, интервьюирование сотрудников и анализ существующих рабочих процессов. Основной целью является понимание, какие процессы наиболее подвержены ошибкам, занимают много времени или ресурсов, а также какие из них являются критическими для проекта.

Существует несколько подходов к описанию бизнес-процессов. Один из них – использование текстового описания, которое подробно описывает каждую операцию, ее входные и выходные данные, а также последовательность действий. Это позволяет создать четкую картину текущего состояния процесса и выявить узкие места и проблемы. Данный подход наиболее эффективен и может быть применен к нашему приложению. Нам важно определить бизнес-процессы, так как определение бизнес-процессов является критически важным этапом в управлении и развитии не только программных продуктов, но и довольно крупных организациях. Это внушает доверие и повышает значимость этого анализа. Он обеспечивает систематическое понимание того, как будет функционировать приложение, и позволяет описать логические пути входа-выхода состояний. Это в сою очередь предоставит возможности более подробно разобраться в системе для улучшения алгоритмов и оптимизации программного продукта. Другой вариант – визуализация процессов с помощью блок-схем или диаграмм потоков. Такие инструменты, как BPMN (Business Process Model and Notation) или UML (Unified Modeling Language), позволяют наглядно представить последовательность операций, взаимосвязи между ними и точки принятия решений. Визуальные диаграммы облегчают понимание процессов для всех участников и служат основой для дальнейшей автоматизации.

В результате описания автоматизируемых бизнес-процессов, мы сможем получить максимально возможную ясность в плане представления. Исходя из которых возможно будет составить карту мыслей о текущем состоянии и принять решения, обоснованные соответствующими диаграммами. Наглядность такого обоснования позволяет точно выделить процессы для автоматизации. Такой системный подход позволяет разработать эффективную стратегию автоматизации, направленную на оптимизацию работы и достижение бизнес-целей.

Как мы узнали выше, для таких целей существуют UML (Unified Modeling Language) диаграммы. Мое мнение таково: определения выглядит внушительно, создается впечатление о том, что составление таких диаграмм необходимо в любой деятельности человека. В данном случае это приоритетное решение для описания автоматизируемых бизнес-процессов. UML предоставляет мощный и универсальный инструмент, который помогает систематически подходить к анализу и оптимизации своих операций. Ознакомимся со способом использования. Описание бизнес-процессов с помощью UML начинается с создания диаграмм, которые иллюстрируют различные аспекты процессов. Одним из ключевых видов диаграмм, используемых для этого, является диаграмма деятельности. Она позволяет представить последовательность шагов в процессе, включая точки принятия решений, параллельные ветви и циклы. Такой визуальный подход помогает легко идентифицировать узкие места, дублирующиеся шаги и возможности для улучшения.

Другой важный инструмент UML – диаграммы прецедентов, которые фокусируются на взаимодействии пользователей с системой. Эти диаграммы помогают понять, какие роли и задачи задействованы в процессе, а также как различные участники взаимодействуют с системой. Это особенно полезно при определении требований к автоматизации и проектировании пользовательских интерфейсов. Использование UML позволяет не только визуализировать текущие процессы, но и моделировать их будущее состояние после внедрения автоматизации. С помощью UML можно создавать сценарии "как есть" и "как будет", что позволяет оценить влияние предлагаемых изменений и выбрать наилучшие решения для автоматизации. Такой подход помогает обеспечить согласованность и понимание среди всех участников проекта, от бизнес-аналитиков до разработчиков и конечных пользователей.

В итоге, было принято решение использовать UML для описания автоматизируемых бизнес-процессов, по нашим подсчетам, это позволит структурированно подходить к процессу автоматизации. Использование этого инструмента обеспечит лучшее понимание текущих процессов, выявление возможностей для улучшения и создание эффективного решения. Для начала определим главные бизнес-процессы:

Бизнес-процесс «Аутентификация пользователя» описывает процесс идентификации пользователя в системе Skill Sale. Его описание представлено на рисунке 2.5.

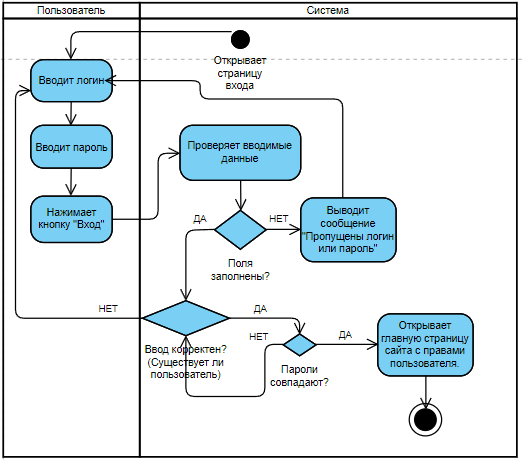


Рисунок 2.5. Диаграмма активностей «Аутентификация пользователя»

Процесс регистрации пользователя в системе с предоставлением прав «Пользователь». В проекте так же будет определена роль «Техник» системы. Роль «Техник» будет предоставлять определенному кругу лиц возможность управлять отображением вакансий и резюме на сервисе, во избежание недопониманий между пользователями системы. Активность регистрации представлена ниже на рисунке 2.6.

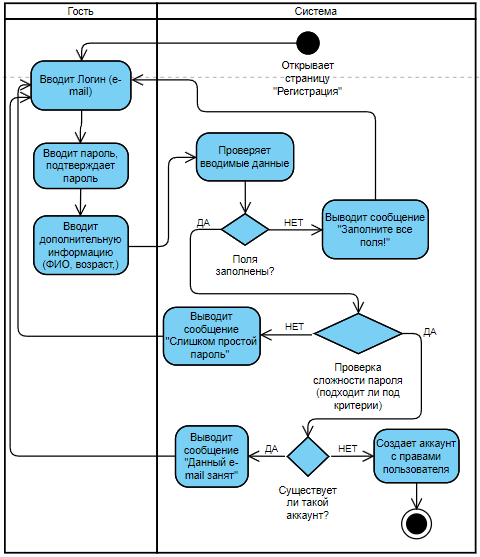


Рисунок 2.6. Диаграмма активностей «Регистрация»

При необходимости определения нового резюме в системе, будет выполняться бизнес-процесс «Создание резюме», эта активность была максимально упрощена, так как от пользователя требуется только ввод необходимой информации. С описанием активности можно ознакомиться ниже на рисунке 2.7.

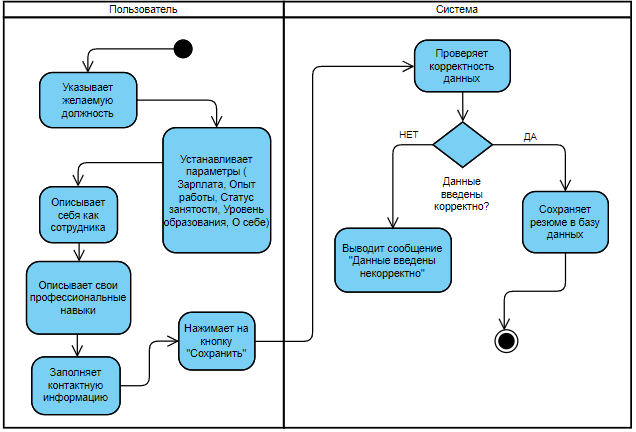


Рисунок 2.7. Диаграмма активностей «Создание резюме»

Проведенный анализ бизнес-процессов веб-сервиса по подбору сотрудников с использованием UML-диаграмм принес значительные результаты и ценные инсайты. Во-первых, визуализация процессов с помощью UML позволила получить четкое и структурированное представление о функционировании сервиса, начиная от получения заявки на подбор сотрудника до успешного найма кандидата. Мы ясно увидели последовательность действий, точки принятия решений и взаимодействие различных участников процесса.

Диаграммы прецедентов помогли определить, какие роли были задействованы в процессе (гость сайта, пользователи в виде сотрудников и соискателей) и как они взаимодействовали с системой. Это выявило ключевые моменты, требующие автоматизации или улучшения, такие как обработка заявок, проведение интервью и управление данными кандидатов.

Диаграммы активности показали последовательность выполнения задач и позволили выявить потенциальные узкие места, избыточные шаги и дублирующие действия. Это дало нам возможность оптимизировать процесс подбора сотрудников, сократив время на выполнение задач и повысив общую эффективность работы.

Использование диаграмм классов для моделирования структуры данных помогло лучше понять, какие данные необходимо было хранить и обрабатывать, а также как они были связаны между собой. Это оказалось полезным при разработке базы данных и интеграции различных модулей веб-сервиса.

Анализ с использованием UML также позволил провести симуляцию различных сценариев и оценить влияние изменений на общий процесс. Мы моделировали внедрение новых функций и автоматизацию отдельных этапов подбора сотрудников, чтобы увидеть, как это повлияет на скорость и качество работы.

В итоге, проведенный анализ обеспечил более глубокое понимание текущих бизнес-процессов, выявил области для улучшений и создал прочную основу для дальнейшей автоматизации и оптимизации веб-сервиса. Это привело к повышению производительности, улучшению качества обслуживания клиентов и, в конечном счете, к успешному и эффективному функционированию системы подбора сотрудников. Проведенный анализ предоставил детальный план по созданию системы подбора персонала. Определение функциональных и нефункциональных требований, а также визуализация бизнес-процессов с помощью диаграмм, позволили создать четкое представление о том, как должна работать система. Эти шаги обеспечивают фундамент для успешной реализации проекта, что позволит сократить время на поиск кандидатов и повысить эффективность подбора персонала.

## Определение поведения системы Skill Sale

В процессе проектирования веб-сервиса по подбору сотрудников мы уделяем особое внимание определению ключевых функциональных возможностей, которые обеспечат эффективность и удобство использования сервиса для наших пользователей.

Одной из первостепенных функциональных возможностей будет система регистрации и аутентификации пользователей. Этот функционал позволит пользователям создавать учетные записи и входить в систему для доступа к сервису, обеспечивая при этом безопасность данных. Этот функционал будет реализован с использованием современных методов шифрования и защиты данных, чтобы обеспечить высокий уровень безопасности. Пользователи будут иметь возможность создавать уникальные учетные записи, а их личная информация будет храниться и передаваться в зашифрованном виде, минимизируя риск утечки данных и несанкционированного доступа. Таким образом, мы гарантируем конфиденциальность и безопасность информации нашего сервиса, что является приоритетом для нас и наших пользователей.

Пользователи могут создавать и редактировать резюме и вакансии. Это поможет им подробно описать свои требования к сотрудникам и представить себя в качестве привлекательного работодателя для потенциальных кандидатов. Также они смогут создать своё резюме. Пользователи также смогут создавать и редактировать свои резюме, что поможет им представить свой профессиональный опыт, навыки и квалификацию в наилучшем свете. Разделы резюме будут структурированы таким образом, чтобы пользователи могли подробно описать свой опыт работы, образование, профессиональные достижения, а также другие важные аспекты своей карьеры. Это позволит им выделиться среди других кандидатов и привлечь внимание потенциальных работодателей.

Кроме того, пользователи смогут создавать вакансии и подробно описывать требования к кандидатам. Они смогут указывать не только профессиональные навыки и опыт работы, но и качества личности, которые являются важными для успешной работы на данной позиции. Это поможет привлечь кандидатов, наиболее подходящих для конкретной вакансии, и повысить эффективность процесса подбора персонала.

Кроме того, в сервисе будет реализован расширенный функционал поиска и фильтрации кандидатов. Пользователи смогут проводить поиск по различным критериям, таким как навыки, опыт работы и образование, что позволит им быстро находить подходящих кандидатов для своих вакансий. Помимо базовых критериев поиска, таких как навыки, опыт работы и образование, расширенный функционал поиска и фильтрации кандидатов включает в себя возможность уточнения по более специализированным параметрам. Это могут быть такие критерии, как уровень знания иностранных языков, сертификаты и лицензии, наличие определенного опыта в конкретных отраслях или с использованием определенных технологий.

Кроме того, пользователи смогут применять множественные фильтры одновременно для более точного и быстрого поиска. Например, будет возможность отфильтровать вакансии или резюме по опыту работы, уровень образования, город, а также пользователи смогут найти требуемое по уникальному идентификатору.

Также будет предоставлена возможность сохранения настроек поиска для последующего использования. Это позволит пользователям сохранить наиболее часто используемые комбинации фильтров и быстро применять их при новом поиске, экономя время и упрощая процесс подбора.

Важным дополнением к системе уведомлений и коммуникации будет возможность настройки предпочтений уведомлений. Пользователи смогут выбирать, какие типы уведомлений они хотели бы получать, а также предпочтительный способ доставки уведомлений: по электронной почте, через мобильное приложение или SMS.

Кроме того, встроенные инструменты коммуникации предоставят возможность пользователю устанавливать контакт с кандидатами напрямую через сервис. Это может включать в себя обмен сообщениями, проведение онлайн-интервью или отправку дополнительной информации о вакансии. Все это будет способствовать более быстрой и эффективной коммуникации между работодателями и кандидатами, что ускорит процесс подбора персонала.

Также стоит упомянуть, что система уведомлений будет обеспечивать двустороннюю коммуникацию, позволяя не только отправлять уведомления, но и получать обратную связь от пользователей. Это позволит нам быстро реагировать на запросы пользователей и улучшать сервис в соответствии с их потребностями и предпочтениями.

Наконец, мы планируем обеспечить интеграцию нашего сервиса с другими системами управления персоналом и отслеживания кандидатов. Это позволит нашим пользователям использовать сервис вместе с уже существующими инструментами, сокращая время и усилия, затрачиваемые на управление процессом подбора персонала.

Все эти функциональные возможности будут интегрированы в наш веб-сервис с целью обеспечения максимальной эффективности и удобства использования для наших пользователей.

## Выбор архитектуры системы Skill Sale

При разработке веб-приложения на первых этапах важно определиться с выбором архитектуры. На данный момент существуют довольно много вариантов, нам же следует выбрать наиболее подходящую, которая наиболее полно охватит потребности совмещая при этом простоту понимания структуры. Перечислим некоторые из них:

Архитектура веб-приложений определяет, как структурированы и взаимодействуют между собой различные компоненты системы. Существует несколько основных типов архитектур, каждая из которых имеет свои особенности и применяется в зависимости от конкретных задач и требований. Рассмотрим основные из них:

1. Многоуровневая архитектура

Говоря о многоуровневой архитектуре в голову сразу же, приходит мысль о чем-то сложном, на самом деле это не так. Вышеописанная архитектура является основой большинства приложений в интернете, что внушает доверие и это не с проста. Многоуровневая архитектура на самом деле представляет собой некий способ организации веб-приложений, представляющая собой некий функционал, разделенный на три основных логических уровня (или больше, в зависимости от сложности проекта и специфики задач). Это разделение является, впрочем, самым логичным и первостепенным так как при разделении проекта на слои интуитивно просто управлять отдельными компонентами, более того такой способ позволяет легко масштабировать приложение. В такой архитектуре на каждом уровне доступно более низкоуровневое разделение, разделение на модули.

Главная идея многоуровневой архитектуры, как мы разобрались, в делении системы на слои, каждый из которых отвечает за выполнение определенной функции, что подходит под основные критерии чистого кода – «Разделяй и властвуй». При таком разбитии проекта нам не нужно изменять старые компоненты системы чтобы обновить функциональность, достаточно лишь добавить новое, это колоссально облегчает поддержку кода и конечно позволяет довольно просто его развивать. Как мы узнали выше, обычно выделяют три слоя – это:

* уровень интерфейса пользователя (представления)
* уровень бизнес-логики
* уровень доступа к данным

Каждый из слоев занимается своим делом. К примеру, уровень интерфейса пользователя занимается взаимодействием с конечным пользователем, обычно это представления, реализованные при помощи фронтенд технологий. Он включает в себя все элементы, с которыми пользователь напрямую взаимодействует, то есть с интерфейсами, они в свою очередь состоят из форм или страниц вывода информации. Проще говоря, уровень обеспечивает отображение данных и обработку(валидацию) данных ввода пользователя.

Уровень бизнес-логики — это некая надстройка над контроллерами, обычно компоненты данного уровня называют сервисами. Главное предназначение уровня в отделении бизнес-логики от кода контроллеров, от уровня доступа к данным. Проще говоря то, как будет переработана информация, что произойдет с ней в будущем, какие алгоритмы для этого будут использоваться, всем этим занимается слой бизнес-логики. Подведя итог можно сказать, что для определения поведения системы, мы должны создать соответствующий слой. Реализовывать алгоритмы, правила обработки данных, в общем случае все, что может повлиять на решения, системы должно находиться именно в слое бизнес-логики. Важно помнить, что этот уровень взаимодействует как с интерфейсом пользователя, так и с уровнем данных, обеспечивая выполнение запросов и бизнес-процессов, но в то же время не является ни тем, ни другим.

Уровень данных как говорит само название управляет хранением и доступом к данным. Он взаимодействует с базами данных, обычно это дополнительные библиотеки, которые позволяют, не прибегая к написанию SQL- запросов руками, управлять данными в файловой системе и другими хранилищами данных, при условии поддержки. Этот уровень отвечает за сохранение, получение и изменение данных, обеспечивая их целостность и доступность.

Преимущества многоуровневой архитектуры довольно очевидны, во-первых, это конечно модульность. Модульность кода в разы облегчает процесс разработки, при этом написание unit-тестов занимает минимум времени. Такой проект, разделенный на слои, довольно неплохо масштабируется при необходимости, простота в том, что при обновлении компонентов мы обращаемся к нужному нам слою, часто изменять что-либо в другом слое не имеет необходимости. Например, если приложение сталкивается с высокой нагрузкой на уровне бизнес-логики, можно добавить больше серверов, обслуживающих этот уровень, без необходимости изменения других уровней, как было сказано ранее.

Кроме того, такая архитектура упрощает поддержку и обновление системы. Поскольку функциональные компоненты разделены на слои, изменения в одном уровне (например, обновление интерфейса пользователя) могут быть сделаны без значительных изменений в других уровнях (например, в бизнес-логике или данных), часто изменения происходят только в одном слое.

В целом, многоуровневая архитектура является гибким и эффективным способом организации веб-приложений, способствующим упрощению разработки, повышению производительности и облегчению масштабируемости. Многоуровневую архитектуру часто делят на два вида, но при необходимости их можно объединить в одно решение:

* Клиент-серверная архитектура: самая базовая модель, в которой клиент (например, веб-браузер) отправляет запросы серверу, а сервер обрабатывает их и возвращает ответы.
* Трехуровневая архитектура: включает три уровня: интерфейс пользователя (клиент), бизнес-логика (сервер), и база данных. Такое разделение помогает улучшить масштабируемость и управляемость приложением.

Каждый из этих типов архитектур имеет свои преимущества и ограничения. Выбор подходящей архитектуры зависит от специфических требований проекта и его целей. Важно тщательно анализировать все варианты, чтобы найти наилучшее решение для конкретной задачи. В нашем случае создаваемый проект будет использовать многослойную архитектуру, а конкретно – будет реализовано клиент-серверное приложение с трехуровневой архитектурой. Далее перейдем к определению минимального стека технологий, соответствующей выбранной архитектуре, возможностью расширения и модернизации.

## Обзор технологий для разработки веб-приложения Skill Sale

Итак, после выбора архитектуры у нас появились некие критерии выбора технологий. Для начала определим их для проведения сравнительного анализа.

Платформа для разработки должна быть:

* Поддерживаемой: наличие обновлений текущего года.
* Масштабируемой: Платформа должна поддерживать возможности масштабирования, чтобы приложение могло обрабатывать увеличивающуюся нагрузку и объем данных по мере роста количества пользователей системы.
* Производительной: Высокая производительность платформы важна для обеспечения быстрого отклика и стабильной работы приложения даже при высокой нагрузке.
* Безопасной: Платформа должна предлагать встроенные механизмы безопасности и соответствовать современным стандартам защиты данных, чтобы минимизировать риски утечек и атак.
* Документированной: Наличие обширной и качественной документации облегчает процесс разработки, позволяет быстрее решать возникающие проблемы и упрощает обучение новых разработчиков.

Для полноценного приложения нам необходимо иметь возможность работы с табличным видом хранения информации. Для решения этой задачи существуют базы данных. На основе анализа функционала приложения, более всего подходят реляционные базы данных. Будем ориентироваться на эту категорию среди множества других видов. Итак, база данных должна отвечать следующим критериям:

* База данных должна обеспечивать быструю обработку запросов и эффективное управление большими объемами данных, что особенно важно для нашего приложения, ориентируясь на перспективу будущего, возможны высокие нагрузки.
* Способность базы данных масштабироваться как вертикально (увеличение мощности отдельных серверов), так и горизонтально (добавление новых серверов) позволяет поддерживать рост приложения и его пользователей.
* База данных должна предоставлять механизмы резервного копирования и восстановления данных, а также поддерживать высокую доступность и устойчивость к сбоям.
* Наличие встроенных механизмов безопасности, таких как шифрование данных, управление доступом и аудит, помогает защитить данные от несанкционированного доступа и утечек.
* Наличие обширной и качественной документации, обучающих материалов и активного сообщества разработчиков в интернете позволяет быстро находить решения для возникающих проблем и эффективно использовать базу данных.

Мы определили критерии технологий для двух звеньев избранной архитектуры, следовательно нужно решить какие технологии избрать для последнего звена – клиента. Исходя из выбранной архитектуры можно сделать небольшой анализ, в плане выбора основы. Но в большинстве вариантов платформа разработки затрагивает и реализацию клиентского взаимодействия. Это проявляется в плане написания клиентской части приложения, к примеру в технологии .Net, есть фреймворк для создания веб-сервисов. В нем клиентская часть реализуется путем совмещения С# кода с технологиями гипертекстовых разметок HTML и стилей CSS. Наличие таких инновационных технологий позволяет наиболее динамично изменять вид страниц, и создавать шаблоны с возможностью встраивания некой логики прямо в представление. К слову, это наиболее приоритетный вариант развития клиентской части. Исходя из этого следуют довольно честные критерии:

* Платформа должна поддерживать последние версии HTML, CSS и JavaScript для обеспечения современного и функционального пользовательского интерфейса.
* Важно, чтобы выбранная технология легко интегрировалась с серверной частью приложения, обеспечивая плавный обмен данными и совместную работу клиентской и серверной логики.
* Клиентская технология должна обеспечивать быструю загрузку страниц и отзывчивость интерфейса, чтобы пользователи могли комфортно работать с приложением даже при высокой нагрузке.
* Встроенные механизмы защиты от распространенных угроз.
* Платформа должна предоставлять удобные инструменты для разработки, отладки и тестирования клиентского кода, что ускорит процесс разработки и повысит качество конечного продукта.
* Возможность легко расширять функциональность приложения и адаптировать его под специфические требования бизнеса поможет быстрее реагировать на изменения и развивать продукт.

Итак, мы довольно точно определили критерии отбора главной системы, теперь важно помнить о том, что нам необходимы некоторые библиотеки. Важность установки пакетов нельзя недооценивать, ведь они значительно способствуют сохранению времени. Это обосновывается тем, что большинство компонентов, соответствующих возникающим задачам, уже с большой долей вероятности уже решены другими разработчиками. В этом деле главное использовать подлинные источники, они обычно встроены в среду разработки. При отсутствии таких возможностей используется консоль диспетчера пакетов – обычная командная строка Windows. Перед установкой необходимо определить наиболее трудоемкие задачи, при условии, что разработка этих систем самому не имеет никакого смысла. Стало быть, определим такие задачи:

* Визуальные стили фронтенд компоненты. Так как объем работ довольно обширен, совсем неплохо было бы заниматься функциональной частью системы (бекендом) больше, чем, «оберткой». Но в то же время «лицо» сайта – это возможность привлечения потенциальных клиентов. Соответственно нужно найти способ удовлетворения обеих потребностей.
* Доступ к данным. Конечно, не плохо писать код взаимодействия с базой данных с нуля. Но проблема опять же во времени и чистоте кода. Зачем мастерить «велосипед», если кто-то его уже сделал и добровольно отдал в личное пользование. Тем более взаимодействие с базой временами требует «пластичности кода», ведь при масштабировании приложения нам придется изменять код сразу в нескольких слоях, что нарушает целостность и заповеди многослойной архитектуры.
* Система миграций. При изменении логики в слое доступа к данным, нередко возникает необходимость масштабирования или отката неких изменений в базе данных. Создать такую систему довольно сложно, а главное бессмысленно, при наличии соответствующих пакетов.

Итак, из выявленных критериев выше, следует то, что теперь нам известны наши ожидания, а это уже пол дела. Далее нам нужно найти соответствия между нашими «хотелками» и реальными проектами в сети интернет. Сейчас проведем обзор существующих технологий, затем утвердим конкретный набор, что составит наш минимальный стек.

1. Java Platform, Enterprise Edition предоставляет мощный набор API и runtime-среду для разработки и запуска крупных, масштабируемых, многослойных, надежных и защищенных сетевых приложений. Java EE поддерживается множеством серверов приложений, таких как Apache Tomcat, JBoss и GlassFish.
2. Node.js: Node.js — это серверная платформа для JavaScript, построенная на движке V8 от Google Chrome. Она позволяет разрабатывать высокопроизводительные и масштабируемые сетевые приложения, особенно веб-сервисы и API. Node.js отличается асинхронной, неблокирующей моделью ввода-вывода, что делает его идеальным для приложений, работающих в реальном времени.
3. Spring Framework: — это популярная платформа для разработки приложений на языке Java. Spring предоставляет всесторонний инфраструктурный каркас, поддерживающий создание всех типов приложений. Он особенно популярен для разработки корпоративных приложений благодаря своей мощной поддержке для обработки транзакций, безопасности и интеграции с другими системами.
4. Django: Django — это высокоуровневая веб-платформа на языке Python, ориентированная на быстрый и чистый дизайн. Она предоставляет множество встроенных инструментов, таких как аутентификация, админ-панель, формирование URL и управление сессиями, что позволяет разработчикам быстро создавать функциональные веб-приложения.
5. ASP.NET: Часть платформы .NET, предназначенная для разработки веб-приложений и сервисов. ASP.NET Core — это кроссплатформенная, высокопроизводительная и открытая версия ASP.NET, позволяющая разрабатывать и запускать приложения на Windows, macOS и Linux.

Эти платформы предоставляют разнообразные возможности и инструменты для разработки приложений, в частности веб-сайтов и сервисов, в большинстве случаев удовлетворяя потребности разработчиков в различных сферах программирования. Но, как мы видим под наши критерии в полном объеме попадает только фреймворк ASP.NET, надстройка над .NET платформой. Довольно неплохой вариант. Платформа долгое время пользуется успехом и зарекомендовала себя как наиболее актуальная. Имеет долгосрочную поддержку технологий и стабильно обновляется не менее 4 раз в год.

Следующим шагом, стало быть, нужно осмотреть технологии баз данных, в частности реляционных. Реляционные базы данных используют структурированный формат данных, который позволяет эффективно организовывать, управлять и извлекать данные с помощью языка SQL. Основные реляционные базы данных включают:

1. MySQL: Самое популярное решение среди реляционных баз данных с открытым исходным кодом. Она широко используется в веб-разработке благодаря своей надежности, простоте в использовании и производительности. MySQL поддерживает транзакции, репликацию, кластеризацию и имеет активное сообщество.
2. PostgreSQL: Мощная реляционная база данных с открытым исходным кодом, известная своей соответствием стандартам SQL и обширными возможностями. PostgreSQL поддерживает транзакции, хранимые процедуры, триггеры и сложные запросы. Она также предоставляет широкие возможности для расширения и масштабирования.
3. Microsoft SQL Server: Коммерческая реляционная база данных от Microsoft, широко используемая в корпоративных средах. SQL Server предлагает высокую производительность, надежность, безопасность и интеграцию с другими продуктами Microsoft. Он также включает мощные инструменты для анализа данных и бизнес-аналитики.

MS SQL Server обладает множеством преимуществ, соответствующих различным критериям для выбора реляционной базы данных. Во-первых, он обеспечивает высокую производительность благодаря оптимизированному движку, поддержке индексов и параллельной обработке запросов, что позволяет эффективно обрабатывать большие объемы данных и высокие нагрузки. Во-вторых, его надежность подтверждается мощными механизмами резервного копирования и восстановления данных, включая полные, дифференциальные и журнальные резервные копии, а также высокую доступность и отказоустойчивость.

В области безопасности MS SQL Server предоставляет широкие возможности, такие как шифрование данных, управление доступом на основе ролей и расширенные механизмы аудита, что помогает защитить данные от несанкционированного доступа и утечек. Кроме того, он легко интегрируется с другими продуктами Microsoft и различными внешними системами, что упрощает разработку и эксплуатацию приложений.

Главное, при разработке приложения подбора персонала использовать только те технологии, которые обеспечивают надежность, производительность и удобство использования системы.

## Конкретный выбор технологий для реализации

Итак, начнем с выбора языка программирования, из которого вытечет выбор платформы, на которой, собственно, и будем разрабатывать веб-сервис, то есть серверная часть.

Первым делом выберем язык программирования, это будет нашей основой основ. На данный момент существуют довольно много различных языков, на любой вкус. Тут можно полагаться на личные предпочтения, поскольку добрая половина языков в большей или меньшей степени имеют инструменты взаимодействия с веб программированием. По прохождению программы обучения в Международном университете Кыргызской Республики, мне довелось опробовать и в какой-то степени разрабатывать Программное Обеспечение на нескольких языках программирования. В том числе: C++, C#, Python, Pascal, Java. Основываясь на своем выборе и предпочтениях, я конечно же выберу язык C#, он является мощным и гибким, разработанным компанией Microsoft. Он прост и легок в изучении. Его сильная сторона - глубокая интеграция с .NET платформой, которая предоставляет богатый набор библиотек и инструментов для разработки широкого спектра приложений, в частности веб-сервисов, что нам как раз таки и требуется.

C# обладает строгой типизацией и объектно-ориентированным подходом, что помогает разработчикам писать надежный и легко поддерживаемый код. Язык активно развивается и поддерживается, регулярно получая обновления, которые включают новые функции и улучшения производительности. Благодаря этому разработчики могут использовать современные практики и инструменты, чтобы создавать эффективные и высокопроизводительные приложения. С его помощью можно легко интегрироваться с различными сервисами и технологиями, что делает его универсальным инструментом для решения самых разных задач в разработке программного обеспечения.

После определения основного языка программирования, на основе обзора технологий выше, можно подвести итог выбора платформы. При использовании языка программирования C#, естественно, платформой сборки будет .NET. Не сложный выбор, учитывая, что других более привлекательных вариантов нет. Помимо платформы, необходимо выбрать фреймворк. Для разработки веб-сайтов на языке C# существуют несколько фреймворков и платформ, подобных ASP.NET, которые предлагают богатый набор инструментов и возможностей для создания современных веб-приложений. Вот некоторые из них:

1. ASP.NET Core: это кроссплатформенный, высокопроизводительный фреймворк для создания современных облачных и веб-приложений. Он предлагает модульную архитектуру, поддержку контейнеров, интеграцию с популярными клиентскими фреймворками (такими как Angular и React), а также улучшенную производительность и масштабируемость. ASP.NET Core также включает в себя ASP.NET MVC и ASP.NET Web API, что позволяет легко создавать веб-сайты и веб-сервисы.
2. Blazor: это фреймворк от Microsoft, который позволяет создавать интерактивные веб-приложения с использованием C# вместо JavaScript. Blazor работает как на стороне клиента (Blazor WebAssembly), так и на стороне сервера (Blazor Server), предоставляя гибкость и возможность выбора наиболее подходящего подхода для конкретного проекта. Blazor упрощает разработку, позволяя использовать единый язык программирования для серверной и клиентской частей приложения.

На основе ASP.NET Core реализовано большинство решений, которые мы можем встретить в интернете. Он зарекомендовал себя как самый лучший инструмент среди прочих. Из представленного следует вывод, что лучшей и наиболее поддерживаемой платформой является ASP.NET Core. Не менее значимым в выборе оказался критерий, связанный с небольшим опытом работы, с данным инструментом.

Для улучшения визуальной составляющей представлений будет использоваться Bootstrap. Наиболее популярный из большинства инструментов, имеет мощный, расширяемый и многофункциональный интерфейсный инструментарий. Одним из ключевых преимуществ Bootstrap является его адаптивность. С ним просто создавать и настраивать при помощи инструмента Sass, готовую систему сеток и компонентов. А также позволяет воплощать проекты в жизнь с помощью мощных плагинов JavaScript.

Entity Framework Core будет использоваться для упрощения работы с базой данных. Этот ORM (Object-Relational Mapping) фреймворк позволяет работать с базой данных, используя объектно-ориентированный подход, что ускоряет процесс разработки и уменьшает количество кода, необходимого для взаимодействия с базой данных.

Git выбран для контроля версий и управления исходным кодом. Этот распределенный VCS (Version Control System) позволяет эффективно отслеживать изменения в коде, работать в команде и управлять различными версиями проекта.

Для работы с языком C# и другими программными продуктами от компании Microsoft, обычно используют среду разработки Visual Studio. Данная среда разработки – многофункциональный инструмент. В ней есть все что нужно для комфортной разработки: Средства работы с текстом различной сложности, поддержка кода других языков программирования, средство работы с базами данных.

Итак, подведя итоги, для конкретной реализации приложения подбора персонала был определен и утвержден стек технологий, включающий среду разработки Visual Studio, C# с ASP.NET Core (MVC), Bootstrap, Microsoft SQL Server и Sql Server Management Studio для возможности подключаться к базе непосредственно, Git и Entity Framework Core. Стек в полной мере охватывает и покрывает потребности разработки. Цель текущего анализа достигнута, следующая ступень – непосредственно использование этих инструментов для разработки нашего программного продукта.

# Глава 3. РАЗРАБОТКА ВЕБ-СЕРВИСА

Глава посвящена разработке веб-сервиса по подбору сотрудников, который призван решить основные проблемы, с которыми сталкиваются сотрудники и работодатели, мы привели из в главе анализа. Рассмотрим основные этапы разработки, начиная с анализа требований и заканчивая внедрением и тестированием готового продукта. Важное внимание будет уделено архитектуре системы, выбору технологий и инструментов, а также разработке интерфейсов, обеспечивающих удобство и эффективность работы пользователей.

Цель данной главы — предоставить читателю полное представление о процессе создания современного и функционального веб-сервиса, который будет призван улучшить качество и скорость найма сотрудников.

Как мы выяснили из предыдущей главы, реализуемый проект будет работать по принципу многослойной клиент-серверной архитектуры. Платформа разработки .NET Core, она обеспечит возможность создания кроссплатформенных приложений и удобную интеграцию с внешними API для сбора данных. Система Skill Sale представляет собой пример трехзвенной архитектуры приложения. Помимо клиентской и серверной частей, в структуру системы входит сервер баз данных, обеспечивающий надежное хранение и управление данными. Клиентская часть реализована с использованием фреймворка ASP.NET MVC, наиболее распространенный в случае разработки веб-сервисов.

Сервер баз данных отвечает за хранение и обработку данных, гарантируя целостность и доступность информации только лицам, имеющим доступ к ней – администратору базы данных и непосредственно из слоя доступа к данным в back-end отделе проекта. Сервер баз данных реализуется средствами технологий систем управления базами данных, в нашем случае за него работает Entity Framework Core, который подключается только с уровня сервера приложения при наличии соответствующей строки подключения в виде JSON-файла.

В концепции веб-приложения с трехзвенной архитектурой уровнем баз данных является система, на которой развернута база данных. В случае реализуемой системы такой СУБД является Microsoft SQL Server.

Серверная часть – основа приложения, здесь находится вся логика сервиса, отвечающая за, принятие входящих подключений, маршрутизация пользователя к нужным ему страницам, и не безответственна за доступ к информации из базы данных, такой как аккаунты пользователей и банку вакансий. Не стоит недооценивать важность кропотливого труда над этой частью. Все что связано с логикой, будет находиться именно здесь, следовательно крайняя точность не помешает. На серверной стороне присутствует слой бизнес-логики приложения, обеспечивающий безопасность и управление правами доступа, а также аутентификации пользователей. В нашем случае на уровне сервера находится приложение .Net Core, разработанное в среде разработки Visual Studio. Схема архитектуры приложения приведена на рисунке 3.1.



Рисунок 3.1. Схема трехзвенной архитектуры приложения

На сервере баз данных mssql будет развернута база данных под названием «Danonenone». На сервере приложений будет запущено веб-приложение, реализующее шаблон ASP.NET Core и использующее необходимые библиотеки для взаимодействия с базой данных, авторизаций и небольших надстроек для быстроты программирования. Взаимодействие между сервером баз данных и сервером приложений будет осуществляться через неявно передающиеся SQL-запросы, благодаря библиотеке Microsoft Entity Framework Core.

На клиентской стороне будет развернуто веб-приложение Skill Sale на базе фреймворка ASP.NET MVC с использованием библиотеки Bootstrap, для создания пользовательского интерфейса. Это приложение будет взаимодействовать с сервером приложений посредством HTTP запросов. В будущем возможен переход на HTTPS протокол, который позволит защитить конфиденциальную информацию от современных попыток хищения.

Решение Skill Sale пройдет деплой и будет развернуто на сервере some.com с использованием серверов IIS. Приложение .NET обычно развертывается в виде одного элемента, соответствующего названию исполняемому файлу с расширением .exe, или одного веб-приложения, выполняющегося в домене приложений служб IIS. Архитектура разработанного приложения содержит один проект. Поэтому вся логика приложения заключена в одном проекте, и после установки на сервере выполнится развертывание как один элемент. Используемый по умолчанию шаблон включает отдельные папки для обязанностей шаблона MVC (модели, представления и контроллеры), а также дополнительные папки для данных и служб.

Итак, мы определили структуру приложения, начнем разработку. Первым делом, откроем среду разработки Visual Studio, создадим шаблон проекта ASP.NET MVC (Модель-представление-контроллер), установим имя проекта и необязательные настройки. После завершения процессов среды мы увидим стандартный шаблон такого приложения. Во вкладке «обозреватель решений», была создана следующая структура:

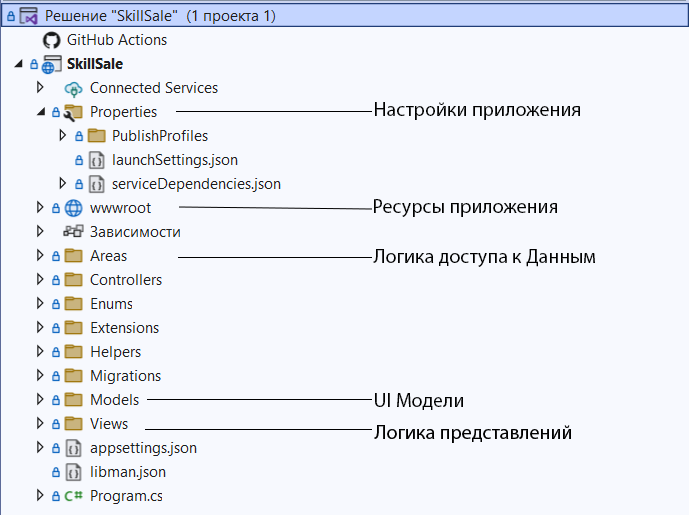


Рисунок 3.2 Файловая структура Skill Sale

## Реализация серверных компонентов

Для того, чтобы начать разрабатывать приложение, давайте ознакомимся с теоретической частью. Бэкенд системы — это серверная часть приложения, отвечающая за логику, обработку данных и взаимодействие с базой данных. Он принимает запросы от клиентской части, обрабатывает их, взаимодействует с базой данных и возвращает ответы клиенту. NET Core используемый как основа, предоставляет мощные инструменты для написания серверного кода, который может выполняться на различных операционных системах, включая Windows, Linux и macOS. Мы же будем писать под операционную систему Windows.

ASP.NET MVC — это фреймворк для разработки веб-приложений, который реализует паттерн "Model-View-Controller" (MVC). Этот паттерн разделяет приложение на три основных компонента.

* Model (Модель): содержит данные и логику работы с ними. Модель отвечает за управление данными приложения, их состояние и взаимодействие с базой данных.
* View (Представление): отвечает за отображение данных пользователю. Представление получает данные от контроллера и отображает их в виде HTML-страницы.
* Controller (Контроллер): обрабатывает пользовательские запросы, взаимодействует с моделью и выбирает подходящее представление для отображения данных. Контроллер является посредником между моделью и представлением.

Когда пользователь взаимодействует с веб-приложением (например, нажимает на кнопку или вводит данные в форму), запрос отправляется на сервер. Контроллер принимает этот запрос и обрабатывает его. В зависимости от типа запроса и логики приложения, контроллер может запросить данные из модели или сохранить новые данные в базу данных.

После обработки запроса контроллер выбирает соответствующее представление и передает ему необходимые данные. Представление, в свою очередь, генерирует HTML-код, который отправляется обратно на клиентскую сторону и отображается пользователю. Таким образом, пользователь видит актуальную информацию и результаты своих действий. Передача данных в представление осуществляется с помощью объектов модели. Контроллер формирует модель с необходимыми данными и передает ее в представление. В представлении данные модели отображаются с помощью синтаксиса Razor Pages, который позволяет вставлять C# код прямо в HTML.

Пример передачи данных:

* - Пользователь запрашивает список навыков.
* - Контроллер получает этот список из базы данных через модель.
* - Контроллер передает список в представление.
* - Представление генерирует HTML-код, отображающий список навыков пользователю.

Исходя из этого бэкенд системы Skill Sale, построенный на .NET Core и ASP.NET MVC, эффективно обрабатывает запросы пользователей, взаимодействует с базой данных и обеспечивает отображение актуальных данных через представления. Лично мне нравится, что эта архитектура делает код более структурированным и понятным, облегчая процесс разработки и улучшая масштабируемость приложения.

Мы уже определили какие пакеты библиотек нам необходимы для разработки ранее, а значит перейдем к их установке. Откроем вкладку «диспетчер NuGet-пакетов». Установим перечисленные ранее пакеты. Эти пакеты обеспечивают фундаментальные функциональные возможности, подключение сервисов аутентификации и работу с данными.

Microsoft.AspNetCore.Identity: Этот пакет от Microsoft предоставляет нам мощный и гибкий инструментарий для управления аутентификацией и авторизацией в нашем приложении. Мы можем легко настраивать права доступа, управлять пользователями и обеспечивать безопасный доступ к ресурсам.

Microsoft.EntityFrameworkCore: Entity Framework Core — это фреймворк для работы с данными, который позволяет нам абстрагироваться от деталей работы с базой данных и сосредоточиться на бизнес-логике. Благодаря EF Core, мы можем упростить доступ к данным, использовать объектно-ориентированный подход к работе с базой данных и обеспечить легкость в разработке.

Microsoft.EntityFrameworkCore.SqlServer: Этот пакет расширяет возможности Entity Framework Core, предоставляя инструменты для работы с базой данных SQL Server. Мы можем использовать все мощные функциональные возможности SQL Server для эффективного хранения и извлечения данных в нашем приложении.

Microsoft.EntityFrameworkCore.Tools: Набор инструментов для разработки с использованием Entity Framework Core. Эти инструменты позволяют нам создавать и управлять миграциями базы данных, генерировать модели из существующей схемы базы данных, а также обновлять базу данных в соответствии с изменениями в модели.

Интеграция этих NuGet-пакетов в наш проект обеспечивает нам не только надежность и безопасность, но и удобство в разработке и поддержке приложения. Мы можем быть уверены, что наш проект строится на надежном фундаменте, который готов к масштабированию и дальнейшему развитию.

При создании серверной части приложения, основываясь и полностью соответствуя проведенному анализу и конечно проектировании системы были разработаны следующие блоки: Model, Views, Controllers, Helpers, Extensions, Enums.

В блоке Models, что соответствует документации официального источника по ASP.NET были определены основные классы, наследованные от DbContext, с помощью которых будет выполняться генерация базы данных. В классах DbSet<T> были указаны основные таблицы элементов, которые позволят взаимодействовать с таблицами, а также установлена структура зависимостей между классами с каскадным удалением элементов, и прописано подключение в файле appsettings.json, к которому подключается SkillSaleContext. Готовая реализация класса SkillSaleContext представлен ниже:

**namespace** SkillSale**.**Data**;**

**public** class SkillSaleContext **:** IdentityDbContext**<**SkillSaleUser**>**

**{**

**public** SkillSaleContext**(**DbContextOptions**<**SkillSaleContext**>** options**)**

**:** **base(**options**)**

**{**

**}**

**public** DbSet**<**SkillSaleUser**>** SkillSaleUsers **{** **get;** **set;** **}**

**public** DbSet**<**Vacancy**>** Vacancies **{** **get;** **set;** **}**

**public** DbSet**<**Resume**>** Resumes **{** **get;** **set;** **}**

**public** DbSet**<**Feedback**>** Feedbacks **{** **get;** **set;** **}**

**protected** **override** void OnModelCreating**(**ModelBuilder modelBuilder**)**

**{**

**base.**OnModelCreating**(**modelBuilder**);**

modelBuilder**.**Entity**<**Vacancy**>()**

**.**HasOne**(**v **=>** v**.**Author**)**

**.**WithMany**(**u **=>** u**.**Vacancies**)**

**.**HasForeignKey**(**v **=>** v**.**AuthorId**)**

**.**IsRequired**()**

**.**OnDelete**(**DeleteBehavior**.**Restrict**);**

modelBuilder**.**Entity**<**Resume**>()**

**.**HasOne**(**r **=>** r**.**Author**)**

**.**WithMany**(**u **=>** u**.**Resumes**)**

**.**HasForeignKey**(**r **=>** r**.**AuthorId**)**

**.**IsRequired**()**

**.**OnDelete**(**DeleteBehavior**.**Restrict**);**

**}**

**}**

## Создание моделей для базы данных

Логично, что перед работой с базой данных, нам необходимо для начала ее создать. Обратимся с этой задачей к сайту some.com. После регистрации, мы можем создать бесплатную базу данных Microsoft SQL Server и выбрать актуальную версию. После этого нам будет доступна строка подключения (Connection String), которую мы используем далее при подключении. В структуре проекта нами созданного, находим файл конфигурации. В нашем случае он называется appsettings.json и напишем в него следующее:

"ConnectionStrings": {

"SkillSaleContextConnection": "Server=server\_addres; Database=database\_name; User Id= login; Password=password; TrustServerCertificate=true; "

}

Здесь, мы определили строку подключения к базе данных. ConnectionStrings, определяет список возможных подключений. Строка SkillSaleContextConnection уже в свою очередь предоставляет возможность описать данные для подключения. Строка подключения и все необходимые для ввода данные берем с сайта somee.

На основании проведенного анализа и проектирования, мы определили необходимость создания нескольких ключевых моделей для базы данных, которые будут автоматически управляться с помощью Entity Framework Core (EF Core) и Identity. Эти модели представляют основные сущности системы и обеспечивают структурированное хранение данных, необходимое для функционирования сервиса SkillSale.

Entity Framework Core (EF Core) является мощным инструментом для работы с базой данных в .NET, позволяющим разработчикам работать с базой данных посредством объектно-ориентированных моделей. При использовании EF Core в сочетании с Identity, система автоматически создает и управляет таблицами базы данных, которые соответствуют определенным моделям. Identity предоставляет встроенные механизмы для управления пользователями, ролями и аутентификацией, что существенно упрощает разработку и безопасность приложения.

Модели базы данных

Для эффективной реализации функциональности системы необходимо создать несколько моделей, каждая из которых будет представлять определенную сущность.

1. Resume (Резюме): Модель резюме будет включать информацию, необходимую для представления профессионального профиля пользователя. Это позволит пользователям добавлять и редактировать свои резюме, указывая навыки, опыт работы, образование и контактную информацию.
2. Vacancy (Вакансия): Модель вакансии будет аналогична модели резюме, но предназначена для работодателей. Она будет содержать данные о доступных вакансиях, включая название, описание, требования и дату публикации. Это позволит работодателям публиковать и управлять вакансиями, а соискателям — искать подходящие рабочие места.
3. Favorite (Избранное): Модель избранного будет использоваться для хранения информации о резюме и вакансиях, которые пользователи отметили, как избранные. Это позволит пользователям сохранять интересующие их позиции или кандидатов для быстрого доступа в будущем.

Автоматизация с помощью Entity Framework Core

Использование Entity Framework Core (EF Core) для автоматического создания и управления базой данных значительно упрощает процесс разработки. При использовании этой библиотеки мы можем определять модели с помощью классов C#, а EF Core будет отвечать за создание соответствующих таблиц в базе данных. Для добавления связей, основы реляционных баз данных, нам достаточно лишь использовать одну модель в другой. К примеру, создадим связь один-ко-многим:

public List<SkillSaleUser>? EmployerCandidates { get; set; }

В этом случае, мы имеем в модели список кандидатов на вакансию. Здесь в виде класса SkillSaleUser, выступает объект пользователя системы.

Миграции в EF Core позволяют легко обновлять структуру базы данных при изменении моделей, что обеспечивает гибкость и адаптируемость системы.

Интеграция с Identity обеспечивает управление пользователями, аутентификацию и авторизацию. Identity автоматически создает таблицы для хранения данных о пользователях и ролях, а также предоставляет методы для регистрации, входа в систему и управления учетными записями. Это позволит сосредоточиться на бизнес-логике приложения, не беспокоясь о реализации базовых функций безопасности.

Создание моделей базы данных для системы Skill Sale, таких как Resume, Vacancy и Favorite, является критическим шагом в разработке функционального и эффективного приложения, так как охватывают максимум возможностей работы с ними. Использование EF Core и Identity автоматизирует процесс создания и управления базой данных, обеспечивая при этом высокую производительность и безопасность системы. Эта структура позволит пользователям и работодателям легко взаимодействовать с сервисом, обеспечивая надежное хранение и доступ к необходимой информации.

На основе сформированной модели данных с помощью технологии .NET Core и диспетчера NuGet-пакетов была выполнена генерация базы данных на ранее созданном сервере Microsoft SQL Server. Для этого использовалась технология миграции (migrations), которая автоматически создала скрипты базы данных на основе разработанных моделей данных. Чтобы убедиться в том, что таблицы в базе данных были составлены правильно, создадим диаграмму объектов при помощи встроенных инструментов программы SQL Server Management Studio, (представлена на рисунке 3.3). На диаграмме присутствуют все таблицы, описанные выше, которые успешно создались и исправно функционируют.

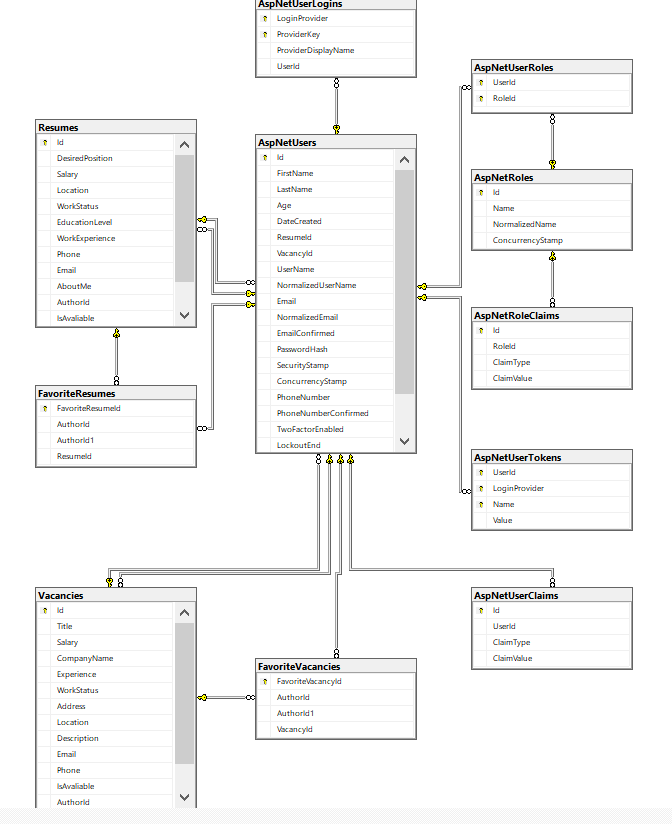


Рисунок 3.3. Диаграмма объектов основной базы данных

Механизм миграций в Entity Framework Core анализирует изменения в моделях данных с момента последней миграции и генерирует соответствующие SQL-скрипты для применения этих изменений к базе данных. Это позволяет поддерживать актуальность структуры базы данных в соответствии с изменениями в приложении без необходимости вручную писать SQL-скрипты. Миграция создается соответствующей командой в консоли диспетчера пакетов “Add-Migration название\_миграции”.

После создания миграции нужно применить изменения к базе данных с помощью команды "Update-Database" в консоли диспетчера NuGet. Это применит сгенерированные SQL-скрипты и обновит структуру базы данных в соответствии с последними изменениями в моделях данных. Получившиеся таблицы можно увидеть в обозревателе объектов (рисунок 3.4).

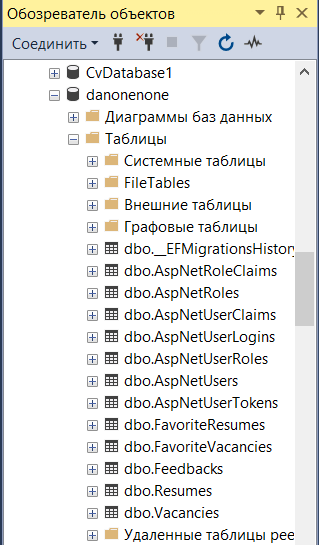


Рисунок 3.4. Сгенерированная база данных в СУБД Microsoft SQL Server

Система регистрации и авторизации разработана таким образом, чтобы пользователи могли легко создавать учетные записи и входить в систему. При регистрации нового пользователя система сохраняет его данные в базе данных при помощи Entity Framework. Эта технология предоставляет мощные инструменты для управления пользователями, включая создание, обновление и удаление учетных записей, а также управление паролями и ролями.

После регистрации пользователь может войти в систему, используя свой логин и пароль. Entity Framework Identity обеспечивает надежное хранение паролей, применяя хеширование и другие методы безопасности, что защищает данные пользователей от несанкционированного доступа.

Одной из ключевых особенностей системы является управление ролями. В проекте Skill Sale реализованы две основные роли: "admin" и "user". Пользователи с ролью "admin" обладают расширенными правами доступа и могут выполнять административные задачи, такие как модерирование вакансий и резюме, управление пользователями и их ролями. Это позволяет администраторам сайта эффективно контролировать контент и взаимодействие пользователей с системой.

Пользователи с ролью "user" имеют ограниченные права доступа, которые позволяют им использовать основные функции сайта, такие как создание и редактирование резюме, отклик на вакансии, а также доступ к личному профилю. Такое разграничение прав доступа обеспечивает безопасность и упрощает управление системой.

Entity Framework Identity также предоставляет механизмы для управления сессиями пользователей и защиты от несанкционированного доступа. Это включает функции для сброса паролей, двухфакторной аутентификации и блокировки учетных записей при подозрительной активности.

В целом, реализация авторизации и регистрации с помощью Entity Framework Identity обеспечивает надежность и безопасность системы, позволяя пользователям комфортно и безопасно взаимодействовать с веб-сервисом. Благодаря использованию современных технологий и подходов, проект Skill Sale предлагает своим пользователям функциональную и защищенную платформу для поиска вакансий и управления своими резюме.

Последний блок разработки back-end части системы – разработка контроллеров. Для каждого класса-модели был создан контроллер, поддерживающий пользовательскую авторизацию и запросы типов HttpGet, HttpDelete, HttpPut и HttpPost. Все основные методы, для доступа и изменения данных, были также успешно реализованы. Названиям методов контроллера соответствуют View-модели (представления), отвечающие за отображение информации в браузере клиента. Рассмотрим один из контроллеров, контроллер ProfileController.

**namespace** SkillSale**.**Controllers

**{**

**[**Authorize**]**

**public** class ProfileController **:** Controller

**{**

**private** **readonly** SkillSaleContext \_context**;**

**private** **readonly** UserManager**<**SkillSaleUser**>** \_userManager**;**

**public** ProfileController**(**SkillSaleContext context**,** UserManager**<**SkillSaleUser**>** userManager**)**

**{**

\_userManager **=** userManager**;**

\_context **=** context**;**

**}**

**[**HttpGet**]**

**public** **async** Task**<**IActionResult**>** Index**()**

**{**

var currentUser **=** **await** \_userManager**.**Users**.**FirstOrDefaultAsync**(**x **=>** x**.**UserName **==** User**.**Identity**.**Name**);**

//footer off

ViewBag**.**Footer **=** **false;**

var result **=** **await** \_context**.**SkillSaleUsers

**.**Where**(**x **=>** x**.**Id **==** currentUser**.**Id**)**

**.**Include**(**x **=>** x**.**Vacancies**)**

**.**ThenInclude**(**v **=>** v**.**CandidatesList**)**

**.**Include**(**x **=>** x**.**Resumes**)**

**.**ToListAsync**();**

**return** View**(**result**);**

**}**

**[**HttpGet**]**

**public** **async** Task**<**IActionResult**>** Portrait**(**string id**)** **{**

**if** **(**id **==** **null)**

**return** NotFound**();**

var user **=** **await** \_userManager**.**Users**.**FirstOrDefaultAsync**(**x **=>** x**.**Id **==** id**);**

var result **=** **await** \_context**.**SkillSaleUsers

**.**Where**(**x **=>** x**.**Id **==** user**.**Id**)**

**.**Include**(**x **=>** x**.**Resumes**)**

**.**ToListAsync**();**

**return** View**(**result**);**

**}**

**[**HttpGet**]**

**[**Authorize**]**

**public** **async** Task**<**IActionResult**>** FavoriteVacancies**()** **{**

//footer off

ViewBag**.**Footer **=** **false;**

var currentUser **=** **await** \_userManager**.**GetUserAsync**(**User**);**

**if** **(**currentUser **==** **null)**

**return** NotFound**();**

**await** \_context**.**Entry**(**currentUser**)**

**.**Collection**(**u **=>** u**.**FavoriteVacancies**)**

**.**Query**()**

**.**Include**(**fv **=>** fv**.**Vacancy**)**

**.**LoadAsync**();**

var favoriteVacancies **=** currentUser**.**FavoriteVacancies

**.**Select**(**fv **=>** fv**.**Vacancy**)**

**.**ToList**();**

**return** View**(**favoriteVacancies**);**

**}**

**[**HttpGet**]**

**[**Authorize**]**

**public** **async** Task**<**IActionResult**>** FavoriteResumes**()** **{**

//footer off

ViewBag**.**Footer **=** **false;**

var currentUser **=** **await** \_userManager

**.**GetUserAsync**(**User**);**

**if** **(**currentUser **==** **null)**

**{**

**return** NotFound**();**

**}**

**await** \_context**.**Entry**(**currentUser**)**

**.**Collection**(**u **=>** u**.**FavoriteResumes**)**

**.**Query**()**

**.**Include**(**fr **=>** fr**.**Resume**)**

**.**LoadAsync**();**

var favoriteResumes **=** currentUser**.**FavoriteResumes

**.**Select**(**fr **=>** fr**.**Resume**)**

**.**ToList**();**

**return** View**(**favoriteResumes**);**

**}}}**

Следующий этап – создание сервисов-помощников для уменьшения кода в контроллерах и упрощения работы с данными, что в свою очередь ускорит процесс разработки. Часто приходится работать с датой, поэтому создадим класс облегчающий этот процесс. Пример разработанного сервиса (DateHelper).

**namespace** SkillSale**.**Helpers

**{**

**public** static class DateHelper

**{**

**public** static int CalculateAge**(**DateTime birthDate**)**

**{**

int age **=** DateTime**.**Today**.**Year **-** birthDate**.**Year**;**

**if** **(**DateTime**.**Today **<** birthDate**.**AddYears**(**age**))**

age**--;**

**return** age**;**

**}**

**}**

**}**

Создание вспомогательного метода для вычисления возраста позволяет избежать повторения одного и того же кода в разных частях проекта. Теперь, мы можем просто вызывать метод CalculateAge, где это необходимо, что экономит время и уменьшает вероятность ошибок.

Также в проекте реализованы расширения, следующий код демонстрирует, как можно использовать расширения методов для улучшения работы с перечислениями (enums) в C#. Он находится в пространстве имен SkillSale.Extensions и содержит статический класс EnumExtensions с методом GetDisplayName.

**namespace** SkillSale**.**Extensions

**{**

**public** static class EnumExtensions

**{**

**public** static string GetDisplayName**(this** Enum **value)** **{**

var field **=** **value.**GetType**().**GetField**(value.**ToString**());**

var displayAttribute **=** field**?.**GetCustomAttribute**<**DisplayAttribute**>();**

**return** displayAttribute**?.**Name **??** **value.**ToString**();**

**}**

**}**

**}**

Представьте, что в вашем проекте вы часто работаете с перечислениями, где каждый элемент может иметь пользовательское имя для отображения. Обычно это достигается путем применения атрибутов Display к элементам перечисления. Вместо того чтобы писать один и тот же код для получения этих пользовательских имен каждый раз, вы создаете расширяющий метод GetDisplayName. Этот метод можно вызывать на любом значении перечисления, и он будет возвращать пользовательское имя, если оно задано, или само значение перечисления, если пользовательское имя не указано. Когда вы вызываете GetDisplayName на значении перечисления, метод использует рефлексию для получения метаданных о поле перечисления. Он ищет атрибут Display, который может содержать пользовательское имя. Если такой атрибут найден, метод возвращает его значение; если нет, возвращается строковое представление значения перечисления.

Это значительно упрощает и ускоряет разработку. Вместо повторного написания однотипного кода для получения пользовательских имен для каждого перечисления, вы просто используете этот метод. Это делает код чище, уменьшает количество ошибок и повышает производительность разработчиков, позволяя им сосредоточиться на более сложных задачах.

## Разработка front-end компоненты системы.

Сбор требований является критически важным этапом разработки любого программного обеспечения, особенно веб-сервиса. На этом этапе определяются функциональные и нефункциональные требования к системе, которые будут влиять на архитектуру, дизайн и реализацию проекта. Для нашего веб-сервиса, разработанного на платформе ASP.NET MVC, сбор требований направлен на создание простого и интуитивно понятного интерфейса, который обеспечивает эффективное взаимодействие пользователя с системой.

В приложении ASP.NET MVC на диске нет страницы, соответствующей пути в URL-адресе, введенном в адресной строке браузера. Ближе всего к странице в приложении MVC ASP.NET называется представлением. В приложении ASP.NET MVC входящие запросы браузера сопоставляются с действиями контроллера. Итогом работы контроллера обычно выступает представление, но не всегда. Действие контроллера может выполнять действия другого типа, например перенаправлять вас на другое действие контроллера или вовсе выполнять работу внутри, вызывая небольшие изменения на странице-инициаторе. Цель сбора требований - детально понять, что именно нужно конечным пользователям и заинтересованным сторонам от веб-сервиса. Это включает как функциональные возможности, так и пользовательский опыт (UX).

## Выявленные требований

Фронтенд компоненты, как ни странно, тоже обладают некоторыми требованиями. Такие требования лучше всего объявлять до процесса разработки. Они позволяют увидеть действо со стороны. Определим их.

Функциональные требования:

* Простой и интуитивно понятный интерфейс:
* Основные функции должны быть легко доступны и понятны пользователям с минимальным уровнем технической подготовки.
* Использование стандартных UI-паттернов для обеспечения знакомого опыта взаимодействия.

Адаптивный дизайн:

* Интерфейс должен корректно отображаться и функционировать на различных устройствах, включая настольные компьютеры, планшеты и смартфоны.
* Использование адаптивной верстки для обеспечения оптимального пользовательского опыта на экранах разных размеров.

Регистрация и авторизация пользователей:

* Удобные и безопасные формы регистрации и авторизации.
* Возможность восстановления пароля и интеграция с социальными сетями для упрощенной авторизации.

Личный кабинет пользователя:

* Доступ к персональной информации, настройкам учетной записи и истории взаимодействий.
* Возможность редактирования личных данных и предпочтений.

Интерактивные элементы и уведомления:

* Реализация интерактивных элементов, таких как модальные окна, всплывающие подсказки и уведомления.
* Обеспечение обратной связи пользователю о выполненных действиях и изменениях в системе.

Нефункциональные требования:

* Быстрое время загрузки страниц и отклик интерфейса на действия пользователя.
* Оптимизация кода и использование кэширования для улучшения производительности.
* Защита данных пользователя через использование шифрования и безопасных методов передачи данных.
* Внедрение механизмов защиты от атак, таких как SQL-инъекции и XSS.
* Простота навигации и минимизация количества действий, необходимых для выполнения задач.
* Проведение тестирования с пользователями для выявления и устранения проблем в UX.

На основании собранных требований были созданы прототипы интерфейса. Эти прототипы прошли несколько итераций, в ходе которых получалась обратная связь от пользователей и заинтересованных сторон. На основе этой обратной связи вносились необходимые изменения и уточнения.

## Проектирование интерфейса

Проектирование интерфейса (UI/UX) является ключевым этапом разработки фронтенд компоненты веб-сервиса, так как он определяет, как пользователи будут взаимодействовать с системой. На этом этапе создаются макеты и прототипы, которые визуализируют требования, собранные на предыдущем этапе. Цель проектирования интерфейса - создать интуитивно понятный, эстетически приятный и функциональный пользовательский интерфейс, который обеспечивает позитивный пользовательский опыт.

Цель проектирования интерфейса - создать такой дизайн, который будет удовлетворять потребности пользователей и соответствовать требованиям проекта. Основные задачи включают:

* Создание структурированной навигации, которая позволяет пользователям легко находить нужную информацию и функции.
* Разработка визуальной идентичности, которая соответствует бренду и создает положительное впечатление у пользователей.
* Обеспечение адаптивного дизайна, который корректно отображается на различных устройствах и экранах.
* Интеграция интерактивных элементов, которые делают взаимодействие с веб-сервисом удобным и приятным.

В проекте Skill Sale реализовано несколько контроллеров, каждый из которых отвечает за определенный функционал сайта. Контроллеры играют ключевую роль в обработке запросов пользователей и управлении соответствующими данными. Для каждого контроллера были созданы соответствующие представления, что обеспечивает удобное и интуитивное взаимодействие пользователей с веб-сайтом.

Контроллер `AppRolesController` отвечает за управление ролями на сервере. Он позволяет добавлять, редактировать и удалять роли, что обеспечивает гибкость в управлении правами доступа пользователей.

Методы и представления

* `Index()`: отображает список всех ролей.
* `Create()`: представление для добавления новой роли.
* `Edit(int id)`: представление для редактирования существующей роли.
* `Delete(int id)`: представление для удаления роли.

Контроллер `ControlPanelController` предоставляет уполномоченным лицам права модерировать вакансии и резюме пользователей. Это важный компонент для обеспечения качества и актуальности размещаемой информации.

Методы и представления:

* `Index()`: отображает панель управления с доступными действиями.
* `ModerateVacancy(int id)`: представление для модерирования вакансий.
* `ModerateResume(int id)`: представление для модерирования резюме.

Контроллер `FeedbacksController` позволяет пользователям сайта обращаться в техническую поддержку. Это важный аспект для поддержания высокого уровня обслуживания и быстрого решения проблем пользователей.

Методы и представления:

* `Index()`: отображает список всех обращений в техподдержку.
* `Create()`: представление для создания нового обращения.
* `Details(int id)`: представление для просмотра деталей обращения.
* `Delete(int id)`: представление для удаления обращения.

Контроллер `HomeController` отвечает за отображение главной страницы веб-сайта. Главная страница служит отправной точкой для пользователей, предоставляя основные навигационные элементы и информацию.

Методы и представления:

* `Index()`: отображает главную страницу с основными разделами и функциями сайта.

Контроллер `ProfileController` позволяет пользователям редактировать информацию о своем профиле. Это включает в себя обновление личных данных, контактной информации и других аспектов профиля пользователя.

Методы и представления:

* `Index()`: отображает информацию о профиле пользователя.
* `Edit()`: представление для редактирования профиля.
* `ChangePassword()`: представление для смены пароля.

Контроллер `ResumesController` управляет резюме пользователей, предоставляя функционал для создания, редактирования и удаления информации о резюме. Это ключевой элемент для пользователей, ищущих работу.

Методы и представления:

* `Index()`: отображает список всех резюме пользователя.
* `Create()`: представление для создания нового резюме.
* `Edit(int id)`: представление для редактирования резюме.
* `Delete(int id)`: представление для удаления резюме.

Контроллер `UsersRoleController` позволяет изменять роли пользователей, предоставляя права доступа к определенным страницам веб-сайта. Это важно для администрирования и контроля доступа.

Методы и представления:

* `Index()`: отображает список пользователей и их ролей.
* `Edit(int userId)`: представление для изменения роли пользователя.

Контроллер `VacanciesController` управляет вакансиями, предоставляя функционал для создания, редактирования и удаления информации о вакансиях. Это основной инструмент для работодателей, ищущих кандидатов.

Методы и представления:

* `Index()`: отображает список всех вакансий.
* `Create()`: представление для создания новой вакансии.
* `Edit(int id)`: представление для редактирования вакансии.
* `Delete(int id)`: представление для удаления вакансии.

Для представлений был разработан дизайн, соответствующий критериям в главе анализа предметной области. Дизайн выполнен в стиле «модальных окон», что упрощает взаимодействие с пользователем, а минималистичность интерфейсов обеспечивает легкость в работе.

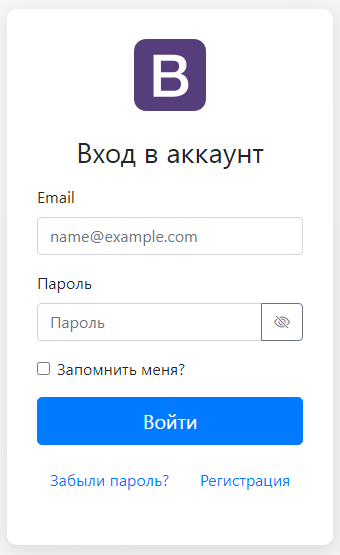


Рисунок 3.5. Страница входа в аккаунт

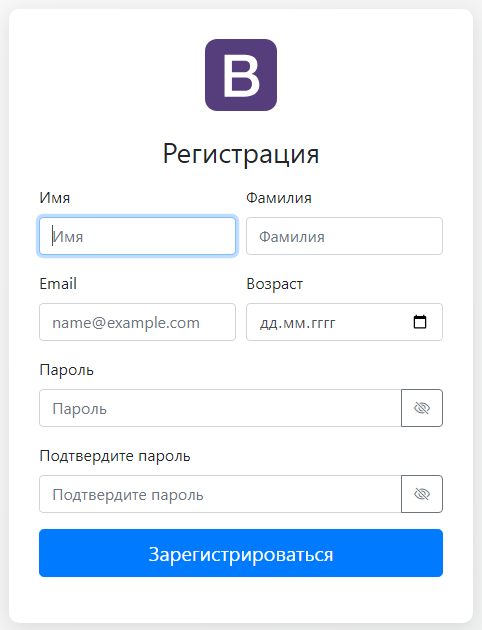


Рисунок 3.6. Страница регистрации пользователя.

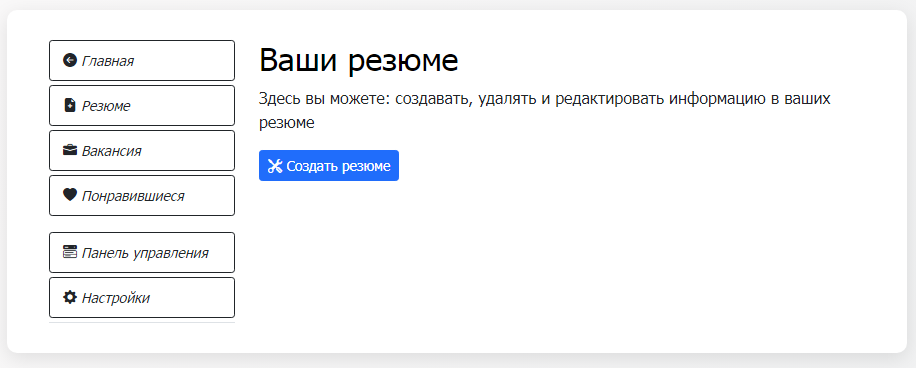


Рисунок 3.7. Панель взаимодействия с сайтом (создание, редактирование, удаление резюме).

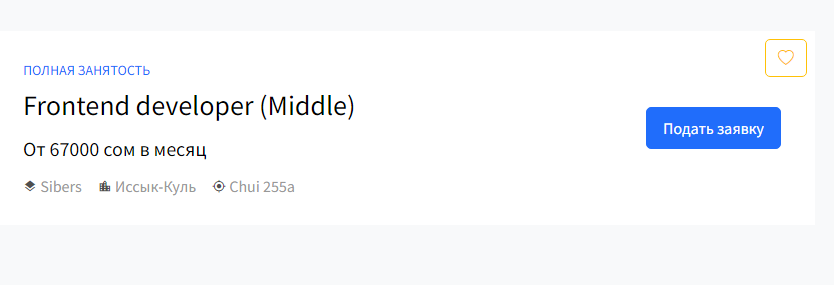


Рисунок 3.8. Вариант представления списка резюме.

Фактически, это представление ведет себя как обычная HTML-страница: здесь можно встретить все стандартные элементы разметки HTML, подключать стили и скрипты. Однако, оно не является чистой HTML-страницей, так как во время выполнения оно проходит компиляцию в сборки, а затем используется для формирования конечного HTML-кода, видимого пользователю в браузере. Это дополнение обеспечивает гибкость и мощь, позволяя встроить динамический функционал в статичные HTML-страницы. Для хранения представлений в проекте ASP.NET MVC предназначена папка Views:

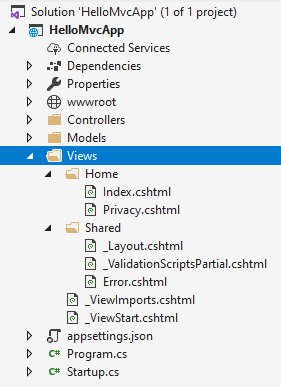


Рисунок 3.9. Структура папок проекта.

В этой папке уже есть некоторая подструктура. Во-первых, как правило, для каждого контроллера в проекте создается подкаталог в папке Views, который называется по имени контроллера и который хранит представления, используемые методами данного контроллера. Так, по умолчанию имеется контроллер HomeController и для него в папке Views есть подкаталог Home с представлениями для методов контроллера HomeController.

При необходимости мы можем добавлять в каталог Views какие-то свои представления, каталоги для представлений. И они необязательно должны быть связаны с контроллерами и их методами.  За работу с представлениями отвечает объект ViewResult. Он производит рендеринг представления в веб-страницу и возвращает ее в виде ответа клиенту. Чтобы возвратить объект ViewResult используется метод View:

**public** class HomeController **:** Controller

**{**

**public** IActionResult Index**()**

**{**

**return** View**();**

**}**

**}**

Представление в ASP.NET MVC может содержать не только стандартный код html, но и также вставки кода на языке C#. Для обработки кода, содержащего как элементы html, так и конструкции C#, используется движок представлений.

## Развертывание системы Skill Sale

Система Skill Sale была развернута на серверах компании somee:

* по завершению разработки проект был скомпилирован и выполнен деплой
* база данных развернута на внутреннем сервере баз данных some.com
* серверные части приложения развернуты на сервере компании some и доступна по адресу http://skillsale.somee.com
* клиентская часть развернута и настроена на взаимодействие с серверной частью с помощью технологии IIS.

# Экспериментальный раздел

Разработанный продукт представляет собой комплексное решение для подбора сотрудников.

Как и любая другая система, система Skill Sale нуждается в тестировании. Для этого был создан массив входных данных для проверки систем по нескольким критериям сразу. Были протестированы следующие функции системы:

* создание, редактирование, удаление вакансий из базы данных;
* создание, редактирование, удаление резюме из базы данных;
* создание, редактирование и удаление пользователей системы;
* регистрация и идентификация пользователей системы, обновление страниц;
* управление ролями пользователей;
* заполнение бланков на основании заработной платы, опыта, пользовательских навыков, указанных в системе и т.д.;

Вышеперечисленные функции были успешно протестированы наиболее достоверными массивами данных, система валидации ввода сработала успешно. В базу данных передались полностью верные значения переменных, ни одно из полей не было пропущено. О чем свидетельствуют рисунки 3.10 и 3.11, предоставленные ниже.

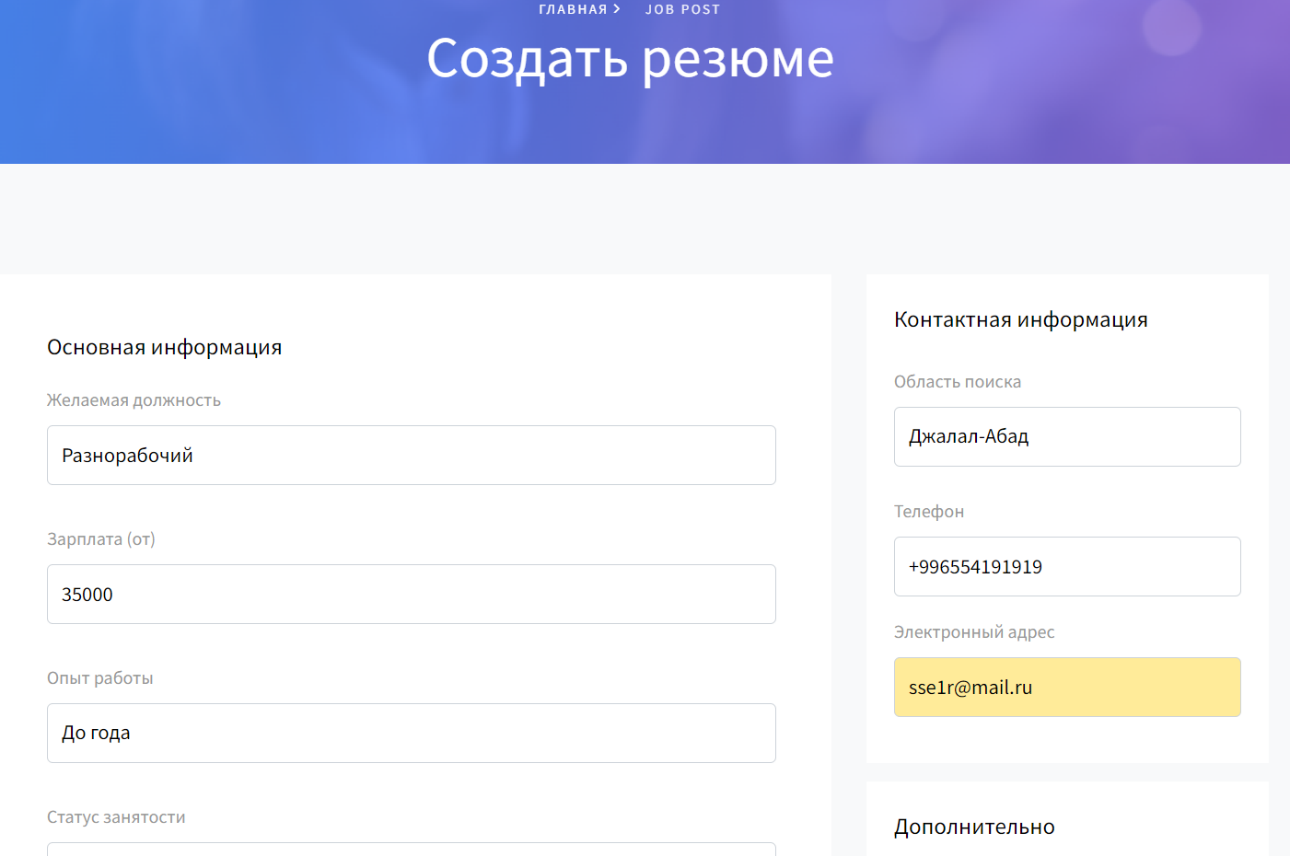


Рисунок 3.10. Ввод информации в систему (резюме).

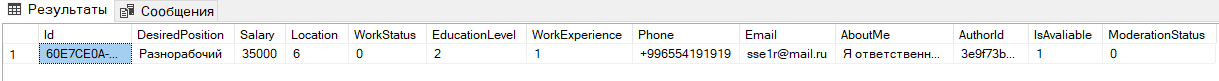


Рисунок 3.11. Хранение информации в базе данных.

Также перед объединением с back-end частью приложения все функции системы были протестированы на тестовом наборе данных, при помощи привлеченных тестировщиков. При тестировании, модели данных были заполнены данными для проверки стойкости системы. Была проверена работоспособность следующих элементов веб-приложения:

* компоненты страниц;
* навигационное меню;
* компоненты для редактирования данных и загрузки изображений;
* компоненты на возможность группировки, скроллинга, выбора элементов;
* компоненты таблицы на возможность сортировки отображаемых данных;

В результате проведенной реализации приложения была спроектирована архитектура системы и были разработаны основные блоки системы: части back-end и front-end, подготовлены модули системы, а также выполнены тестирование системы, рефакторинг кода, публикация и развертывание. Все компоненты системы интегрированы в единое целое, обеспечивая надежную и эффективную работу веб-сервиса. Было уделено особое внимание оптимизации производительности и безопасности, что позволило создать стабильное и безопасное приложение. Также была проведена работа по документированию всех этапов разработки и созданию пользовательской документации, что облегчает дальнейшее сопровождение и развитие системы. Результаты тестирования показали высокую надежность и производительность приложения в условиях реального использования.

# Заключение

В ходе выполнения выпускной квалификационной работы была достигнута основная цель: разработана программная система, автоматизирующая процесс подбора персонала. Разработка проводилась в среде MS Visual Studio с использованием современных технологий, таких как .NET Core, Entity Framework и Bootstrap. База данных была реализована с использованием системы управления базами данных (СУБД) Microsoft SQL Server. Для управления версионностью проекта применялась система контроля версий GitHub.

Разработанная система поддерживает работу трех типов пользователей: неидентифицированный гость, зарегистрированный пользователь и администратор. Неидентифицированные гости могут просматривать информацию о компаниях и их вакансиях. Зарегистрированные пользователи получают расширенные возможности: помимо просмотра компаний и вакансий, они могут публиковать свои резюме, просматривать резюме других соискателей, создавать отчеты по вакансиям и заполнять их через удобный интерфейс. Администраторы обладают полным доступом к системе, включая учет всех данных, регулярный сбор и очистку устаревших данных, отслеживание действий пользователей и создание статистических отчетов о работе системы.

Программная система была развернута на серверах компании somee и предназначена для персонального использования в учебных и исследовательских целях. Выполненная работа демонстрирует применение современных технологий и методологий разработки программного обеспечения, обеспечивая надежную и эффективную автоматизацию подбора персонала.В процессе работы над дипломным проектом я углубил свои знания в области разработки веб-приложений на основе ASP.NET MVC с использованием технологий .NET Core, Entity Framework Core и Identity. Целью моей дипломной работы было изучение и применение принципов разработки веб-приложений с целью создания системы для поиска работы и подбора персонала.

Благодаря выполнению проекта я достиг цели разработки системы SkillSale, которая представляет собой веб-приложение для эффективного поиска работы и подбора персонала. В процессе работы над проектом я освоил принципы MVC архитектуры, научился работать с базами данных и интегрировать различные технологии для создания полноценного приложения.

Работа над дипломным проектом также помогла мне развить навыки анализа требований, проектирования и реализации программного обеспечения, а также эффективного взаимодействия с командой. Полученный опыт и знания будут полезны для моего дальнейшего профессионального роста в области разработки программного обеспечения.

В рамках написания выпускной квалификационной работы были выполнены следующие задачи:

* Проанализирована предметная область подбора персонала, а также проведен сравнительный анализ существующих в этой области программных систем.
* Выявлены, сформулированы и формализованы основные требования к системе подбора персонала в IT‑компаниях с помощью UML диаграммы прецедентов.
* Описаны автоматизируемые системой бизнес-процессы в нотации UML диаграмм активностей.
* Спроектирована концептуальная модель предметной области, выполнено описание поведения системы в нотации UML диаграмм последовательностей, спроектированы основные классы разрабатываемой системы в нотации UML диаграмм классов, выполнено проектирование модели.
* Реализованы клиентская часть системы на языке HTML, CSS, JavaScript, фреймворке ASP.NET MVC и серверная часть приложения на языке C#, фреймворк .Net Core.
* Проведено тестирование разработанной системы SkillSale.
* Написана основная документация для разработанной системы Skill Sale.

Пути развития программы:

* Реализация возможности отправления уведомлений на привязанную почту пользователя с приглашением идентифицироваться в системе.
* Реализация системы доставки уведомлений об откликах, и состоянии вакансии.
* Реализация возможности обмена тестами для тестирования потенциальных сотрудников внутри системы, перед собеседованием.
* Реализация системы указываемых навыков (тэгов) для более точного отбора сотрудников.
* Реализация возможности общения между соискателями и работодателями.
* Реализация нейросети для подбора максимально соответствующих работодателя и сотрудника.

# Библиографический список

1. Блохина О., Петрова Е. HRM-системы в управлении персоналом. – Саранск: «Системное управление», 2009. 8 с.
2. Водовозова А.Б., Резникова О.С. НИЗКИЙ УРОВЕНЬ МОТИВАЦИИ ПЕРСОНАЛА. В сборнике: СОВРЕМЕННЫЕ ТЕХНОЛОГИИ УПРАВЛЕНИЯ ПЕРСОНАЛОМ. сборник научных трудов. Симферополь, 2016. С. 135-139
3. Буч Г. Рамбо Д., Якобсон И. Язык UML. Руководство пользователя. – М.: ДМК Пресс, 2006
4. Дергачев А.М. Проблемы эффективного использования сетевых сервисов // Научно-технический вестник СПбГУ ИТМО, 2011. № 1 (71). С. 83–87.
5. Плаксин М.А. Тестирование и отладка программ для профессионалов будущих и настоящих. М.: БИНОМ. Лаборатория знаний, 2015.
6. Роббинс Д.Н. HTML5, CSS3 и JavaScript. Исчерпывающее руководство. 4-е изд. М.: Эксмо, 2014.
7. Орунбасарова Ж., Кошимова М. Функции и роль HR-менеджера в современной организации. – Международный экономический форум, 2012.
8. Подбор персонала // Talantix. 2018. URL: https://talantix.ru/?utm\_source=article&utm\_medium=referral&utm\_campaign=504207 (дата обращения: 07.12.2024).
9. Руководство по ASP.Net Core 2 [Электронный ресурс]. URL: https://metanit.com/sharp/aspnet5/ (Дата обращения: 10.04.2024).
10. Справочник «Паттерны проектирования» [Электронный ресурс]. URL: http://design-pattern.ru/patterns/ (Дата обращения: 12.04.2024)
11. .NET Framework [Электронный ресурс]. URL: http://accord-framework.net/ (Дата обращения: 30.04.2024)
12. ASP.NET Core MVC Представления. https://metanit.com/sharp/aspnet5/7.1.php (Дата обращения: 11.04.2024).
13. Разработка программной системы автоматизации подбора сотрудников в IT-компаниях. https://revolution.allbest.ru/programming/01179332\_0.html (Дата обращения: 11.04.2024).