Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Магнитогорский государственный технический университет   
им Г.И. Носова»

Многопрофильный колледж

Отделение №2 Информационные технологии и транспорт

**КУРСОВОЙ ПРОЕКТ**

**РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

по ПМ.05. Проектирование и разработка информационных систем

МДК.05.02. Разработка кода информационных систем

на тему: Разработка автоматизированной информационной системы по поиску работы

Исполнитель: Хайсаров.В.И . студент 3 курса, группа ИспВ-20-1

Руководитель: Зорина И.Г., преподаватель МпК

Работа допущена к защите «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_ г. \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Работа защищена «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_ г. с оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Магнитогорск, 2023

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Магнитогорский государственный технический университет   
им. Г. И. Носова»

Многопрофильный колледж

**КУРСОВОЙ ПРОЕКТ**

Тема Разработка автоматизированной информационной системы по поиску работы

Студент Хайсаров.Владислав.Ильнурович

Задание

Исходные данные:

по заданной предметной области:

1. выполнить моделирование информационной системы;
2. спроектировать и разработать базу данных;
3. разработать серверную и клиентскую части приложения;
4. разграничить права доступа;
5. выполнить тестирование информационной системы.

Состав и содержание проекта:

1. построение UML-диаграмм: вариантов использования, последовательности, состояния, деятельности, классов; потоков данных;
2. логическое проектирование, описание концептуальной модели данных, разработка базы данных в выбранной СУБД;
3. разработка интерфейса приложения, логики работы приложения: оформление заявок на услуги, вывод отчетной документации и т.д.;
4. администрирование приложения, разработка панели администратора;
5. разработка комплекта тестирования информационной системы.

Срок сдачи: « » 20 г.

Руководитель: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /И.Г. Зорина

« » 2023 г.

Задание получил: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / В.И.Хайсаров

« » 2023г.

Магнитогорск, 2023

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 4](#_Toc139537939)

[1 ПРОЕКТИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ 6](#_Toc139537940)

[1.1 Анализ предметной области 8](#_Toc139537941)

[1.2 Моделирование информационной системы 11](#_Toc139537942)

[1.3 Логическое и физическое проектирование баз данных 16](#_Toc139537943)

[1.4 Выбор и обоснование средств разработки 21](#_Toc139537944)

[2. РАЗРАБОТКА ИНФОРМАЦИОННОЙ СИСТЕМЫ 26](#_Toc139537945)

[2.1 Создание базы данных 26](#_Toc139537946)

[2.1 Логическая и физическая структура сайта 36](#_Toc139537947)

[2.3 Разработка интерфейса пользователя 38](#_Toc139537948)

[2.5 Разработка выходных документов 43](#_Toc139537949)

[2.6 Разграничение прав доступа 47](#_Toc139537950)

[2.6 Руководство пользователя 49](#_Toc139537951)

[2.7 Тестирование информационной системы 55](#_Toc139537952)

[ЗАКЛЮЧЕНИЕ 60](#_Toc139537953)

[СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ 61](#_Toc139537954)

[ПРИЛОЖЕНИЕ А 62](#_Toc139537955)

# ВВЕДЕНИЕ

В настоящее время поиск работы является одной из наиболее актуальных и важных задач для многих людей. Рынок труда постоянно меняется, и чтобы найти подходящую работу, необходимо быть готовым к постоянному обучению и развитию своих профессиональных навыков.

Создание сайта по поиску работы является актуальным в настоящее время из-за нескольких причин. Сайт может быть полезен для любой организации или частного лица, которые хотят представить себя в интернете или использовать его возможности для достижения своих целей. Во-первых, современные технологии позволяют упростить процесс поиска работы и сделать его более эффективным. На сайте можно разместить свое резюме и получить доступ к большому количеству вакансий, что значительно увеличивает шансы на трудоустройство.

Во-вторых, создание сайта по поиску работы позволяет работодателям быстро и удобно находить подходящих кандидатов на вакансии. Это экономит время и сокращает затраты на поиск персонала.

Кроме того, сайт по поиску работы может предоставлять дополнительные услуги, такие как консультации по трудоустройству, обучение навыкам поиска работы, а также информацию о рынке труда и требованиях работодателей.

Актуальность создания сайта в контексте поиска работы заключается в том, что он может помочь потенциальному работодателю получить более полное представление о кандидате на вакансию. На сайте можно разместить информацию о своем опыте работы, образовании, профессиональных достижениях и навыках. Это позволит работодателю быстрее и проще принять решение о приглашении на собеседование.

В целом, создание сайта в контексте поиска работы может быть полезным для тех, кто хочет выделиться на фоне других кандидатов и повысить свои шансы на получение желаемой вакансии.

Таким образом, создание сайта по поиску работы является актуальным и полезным для всех участников рынка труда, и может значительно улучшить процесс трудоустройства.

Целью данной работы является изучение основных аспектов поиска работы, а также выработка рекомендаций и советов для тех, кто хочет успешно найти работу.

Для достижения данной цели были поставлены следующие задачи:

* составить анализ предметной области;
* создать модель информационной системы;
* разработать базу данных
* разработать интерфейс пользователя
* разработать выходные документы;
* создать разграничение прав доступа;
* разработать руководство пользователя;
* протестировать приложение.

Практическая значимость курсового проекта заключается в том, что разработанный сайт по поиску работы может быть использован как удобный и понятный инструмент для соискателей вакансий и работодателей.

В современном мире существует множество веб-платформ и приложений, предназначенных для поиска работы, однако разработанный вами сайт обладает определенными преимуществами. Он предоставляет пользователям удобный и интуитивно понятный интерфейс, который помогает соискателям вакансий найти подходящие вакансии и отправить свои резюме работодателям. Также, работодатели могут использовать сайт для размещения вакансий и нахождения подходящих кандидатов.

# 1 ПРОЕКТИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ

Информационная система - это совокупность программных и аппаратных средств, которые используются для сбора, хранения, обработки и передачи информации в рамках организации или предприятия.

Проектирование базы данных - это процесс создания структуры и организации данных в базе данных. Он включает в себя определение требований, создание концептуальной, логической и физической моделей, реализацию базы данных, тестирование и оптимизацию, а также развитие и поддержку.

Определение требований - это первый этап проектирования базы данных. Он включает в себя определение целей и задач, которые должна решать база данных, а также определение данных, которые будут храниться в базе данных. Это может включать в себя определение типов данных, ограничений, связей между таблицами и других факторов.

Создание концептуальной модели - это следующий этап проектирования базы данных. На этом этапе создается общая структура базы данных, которая будет отражать сущности, связи между ними и атрибуты. Это может включать в себя создание диаграммы сущность-связь или других моделей.

Создание логической модели - это следующий этап проектирования базы данных. На этом этапе создается модель, которая будет отражать отношения между таблицами, а также определение ключей и связей между таблицами. Это может включать в себя создание диаграммы ER или других моделей.

Создание физической модели - это следующий этап проектирования базы данных. На этом этапе создается модель, которая будет отражать способ хранения данных на диске, определение типов данных и индексов. Это может включать в себя определение таблиц, индексов, ограничений и других аспектов.

Реализация базы данных - это следующий этап проектирования базы данных. На этом этапе создаются таблицы, индексы, ограничения и триггеры, а также наполняется база данных начальными данными.

Тестирование и оптимизация - это следующий этап проектирования базы данных. На этом этапе проверяется работоспособность базы данных, оптимизируются запросы и индексы для повышения производительности.

Развитие и поддержка - это последний этап проектирования базы данных. На этом этапе обновляется и дорабатывается база данных в соответствии с изменяющимися требованиями бизнеса и поддерживается ее работоспособность.

Нормализация базы данных - это процесс оптимизации структуры базы данных, который позволяет устранить избыточность данных и повысить эффективность их хранения и обработки. Этот процесс включает в себя разделение таблиц на более мелкие и связывание их между собой, чтобы избежать дублирования данных и обеспечить более эффективный доступ к информации.

Иначе говоря, это процесс организации данных в базе данных, который позволяет устранить избыточность и неоднозначность данных, что повышает эффективность хранения и обработки данных. Вот основные этапы нормализации базы данных:

Первая нормальная форма (1НФ): в этой форме каждое поле должно содержать только одно значение, а все повторяющиеся группы данных должны быть выделены в отдельную таблицу.

Вторая нормальная форма (2НФ): в этой форме каждое поле должно зависеть только от первичного ключа таблицы, а не от других полей.

Третья нормальная форма (3НФ): в этой форме каждое поле должно зависеть только от первичного ключа таблицы, а не от других полей, которые не являются частью первичного ключа.

Нормальная форма Бойса-Кодда (НФБК): в этой форме каждый неключевой атрибут должен зависеть только от ключа целиком, а не от его части.

Четвертая нормальная форма (4НФ): в этой форме каждый многозначный атрибут должен быть зависимым только от ключа целиком, а не от его частей.

Пятая нормальная форма (5НФ): в этой форме каждый многозначный атрибут должен быть зависимым только от ключа целиком, а не от других многозначных атрибутов.

Каждый этап нормализации базы данных позволяет устранять избыточность и неоднозначность данных, что повышает эффективность хранения и обработки данных. Однако, не всегда необходимо достигать 5НФ, так как это может привести к усложнению структуры базы данных и замедлению ее работы.

# 1.1 Анализ предметной области

В качестве предметной области выбрана компания по поиску работы. Система необходима для поиска работников для предприятий и трудоустройству населения.

От работодателей поступает вакансия, на которую агентство по трудоустройству ищет работника.

Название вакансии, должно содержать только информацию, относящуюся к названию должности. Такая информация, как рекламные слоганы, график работы, название компании, продукта, сервиса, улицы, станции метро, вид занятости, по которому вы принимаете на работу, должна быть вынесена в текст вакансии

Какая специализация необходима и разряд данной специализации, ожидаемый уровень дохода для соискателя, где будет проходить работа, будет расписано описание вакансии иначе говоря какие требования к данной работе и условия необходимы, требуемый опыт работы для данной вакансии, обязанности сотрудника на этой работе, необходимые ключевые навыки.

Тип занятости подразумевает:

– полная занятость;

– частичная;

– проектная работа;

– или разовое задание;

– волонтерство;

– стажировка.

Режим работы:

– производство только в выходные дни;

– или сменами по 4-6 часов в день;

– и можно начинать работу после 16:00.

В графике работы может быть указано:

– полный день;

– сменный график;

– гибкий график;

– удаленная работа;

– вахтовый метод.

В вакансии еще имеется информация о работодателе, организации и контактные данные.

Дополнительно в вакансии может содержаться требование к знаниям различных языков, должно быть указано: необходимый язык, и уровень владения языком. Еще может храниться информация о наличии водительского удостоверения, то есть категория прав и дата получения.

После отправки заполненной вакансии, данную вакансию могут просматривать соискатели, и откликаться на данные вакансии. И если, соискатель заинтересует работодателя то предложит пройти собеседование.

Соискатель в свою очередь, чтобы работодатель обратил внимание, составляют собственное резюме.

В резюме хранится информация о личных данных соискателя:

– фамилии;

– Имени;

– отчестве;

– номер телефона;

– дата рождения;

– город проживания;

– пол;

– наличие гражданства.

Опыт работы в какой-либо сфере, желаемая должность которую ищет, заработную плату которая ему необходима.

Готов ли соискатель к переезду в какой-либо город или область, желаемая занятость, может быть полная либо частичная, проектная работа, волонтерство стажировка.

Указывается информация о образование, включает в себя: уровень образования, в каком учебном заведение обучался, на каком факультете, специализация, год окончания учебного заведения

Дополнительно можно указать знания различных языков здесь должно быть указан необходимый язык, и уровень владения этим языком. Еще может храниться информация о наличии водительского удостоверения, то есть категория прав и дата получения

После отклика на вакансию работодатели видят в сообщениях информацию о соискателе в откликах на вакансию, что работодатели видят, не открывая резюме:

– Название резюме (указываете это сами, когда создаете или редактируете резюме на сайте);

– Возраст (но только если указали в резюме дату рождения, а указывать это, не обязательно);

– Зарплату, на которую претендуете;

– Общую продолжительность опыта работы: это сумма всех периодов работы, перечисленных в резюме (например, если указаны три места работы и в каждом трудились по одному году, то будет написано — «опыт работы 3 года»);

– Последнее или текущее место работы из резюме: наименование компании-работодателя, продолжительность работы (если работаете до сих пор, то видно месяц и год начала работы там, а также фразу «по настоящее время») и должность.

Далее соискатели могут открыть резюме и полностью ознакомиться с данными в полном объеме.

Обычно работодатели смотрят отклики один-два раза в день, в выделенное для этого время. Изучают список, выбирают несколько подходящих кандидатов и открывают их резюме: у этих откликов появляется отметка «Просмотрен». Есть менеджеры, которые специально нажимают кнопку «Отметить как просмотренный» — чтобы соискатели видели, что их резюме не потерялось во входящих. Но когда откликов много, то это занимает значительное количество рабочего времени, и часть резюме все-таки остается без внимания.

Чтобы отсеивать по резюме было проще, работодатели иногда используют простой трюк: в конце текста вакансии задают вопрос, ответ на который просят прислать в сопроводительном письме. В этом случае смотрят только на резюме с сопроводительными письмами.

Количество показов — это сколько раз резюме появлялось в поисковой выдаче у работодателей, когда искали кандидатов по базе резюме.

Дело в том, что работодатели ищут сотрудников двумя способами:

– Размещают вакансию и разбирают поступившие на нее отклики;

– Ищут подходящих кандидатов в базе резюме;

Показы резюме связаны со вторым способом.

Если соискатели заинтересованы, то откликаются на данную вакансию. После отклика работодатель рассматривает резюме соискателя.

Компания по трудоустройству знает, от кого поступила вакансия, какой специалист необходим предприятию. В вакансии хранится сведения о предприятиях с данными, сведения от предприятия (вакансии), сведения о работниках, ищущих работу, сведения о специальностях, по которым могут работать работники для того, чтобы осуществить поиск по специальности. Основными функциями агентства по трудоустройству являются обслуживание клиентов и предприятий, регистрация новых клиентов и предприятий, подбор работников по вакансиям, подготовка отчетов для руководства.

# 1.2 Моделирование информационной системы

Моделирование – это процесс создания абстрактной модели реальной системы или процесса, с целью изучения свойств и поведения. Моделирование информационной системы включает в себя создание модели, которая отображает информационные потоки, функциональные блоки и связи между ними. Эта модель может быть использована для анализа и оптимизации работы информационной системы, а также для принятия решений по улучшению. В результате моделирования может быть разработана более эффективная и оптимальная информационная система, что позволяет снизить затраты на эксплуатацию и повысить качество предоставляемых услуг.

Диаграмма вариантов UML – это инструмент для моделирования системы, который отображает взаимодействие ее компонентов с актерами (пользователями системы) в различных сценариях использования.

Диаграмма вариантов помогает определить функциональные требования к системе и описать ситуации, в которых пользователь взаимодействует с ней. Это позволяет разработчикам лучше понимать, как система должна работать и как взаимодействовать с пользователем. Диаграмма вариантов может использоваться на стадии проектирования, а также как инструмент для общения между разработчиками и заказчиком проекта.

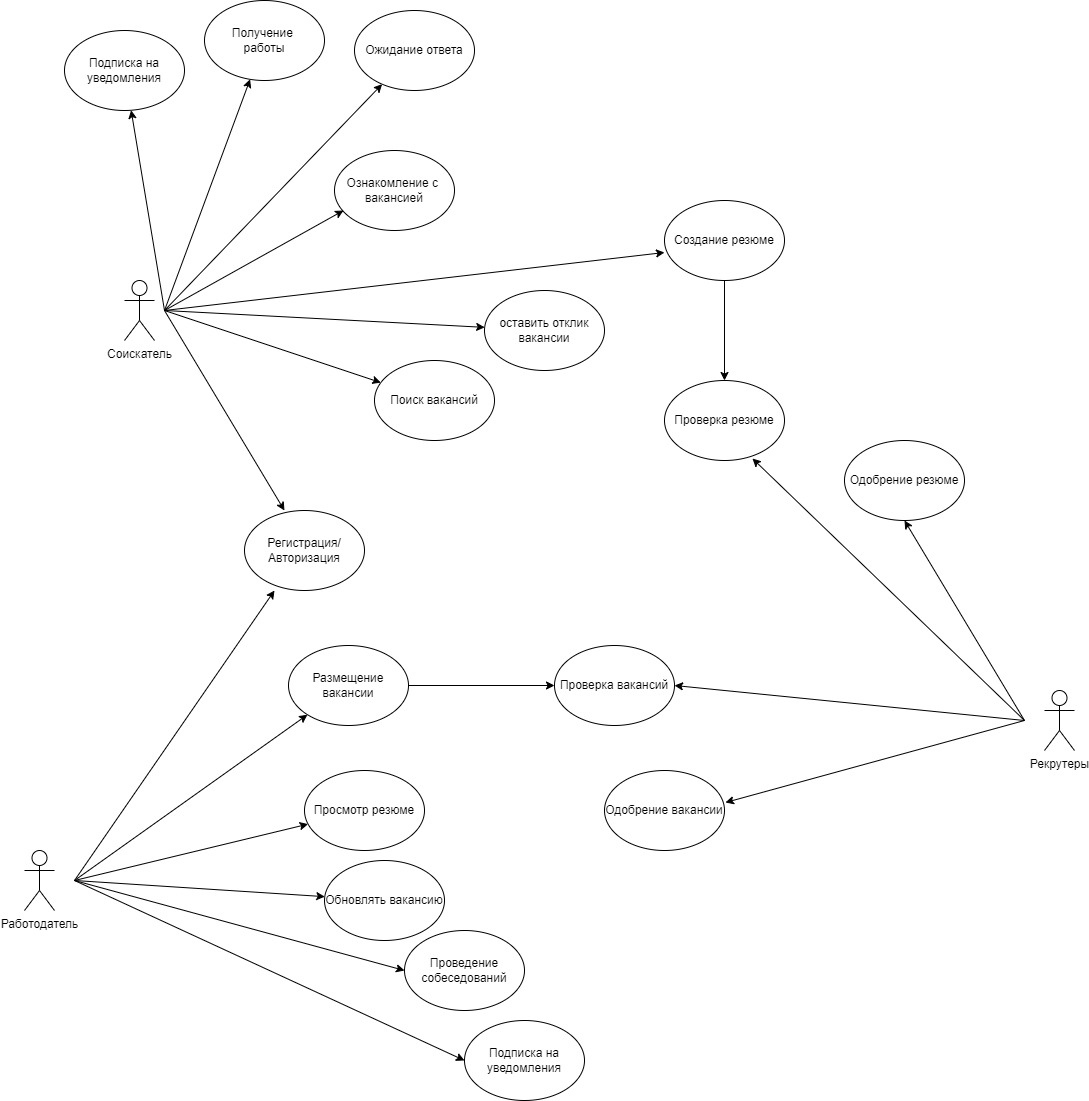
Диаграмма вариантов (рисунок 1).

Рисунок 1 – Диаграмма вариантов

Диаграмма состояний UML – это инструмент моделирования системы, который отображает изменения состояний объектов в различных ситуациях.

Диаграмма состояний помогает описать поведение объекта в системе, где объект может находиться в различных состояниях и переходить между ними в зависимости от входных данных или событий. Эта диаграмма может помочь разработчикам лучше понять, как объекты будут взаимодействовать в системе, и как изменения состояний объектов будут влиять на работу системы.

Диаграмма состояний часто используется на стадии анализа и проектирования системы, а также в качестве инструмента общения между заказчиком и разработчиками. Она может быть особенно полезна для систем, где объекты могут иметь сложное поведение и должны переходить между многими состояниями.

Диаграмма состояний (рисунок 2).

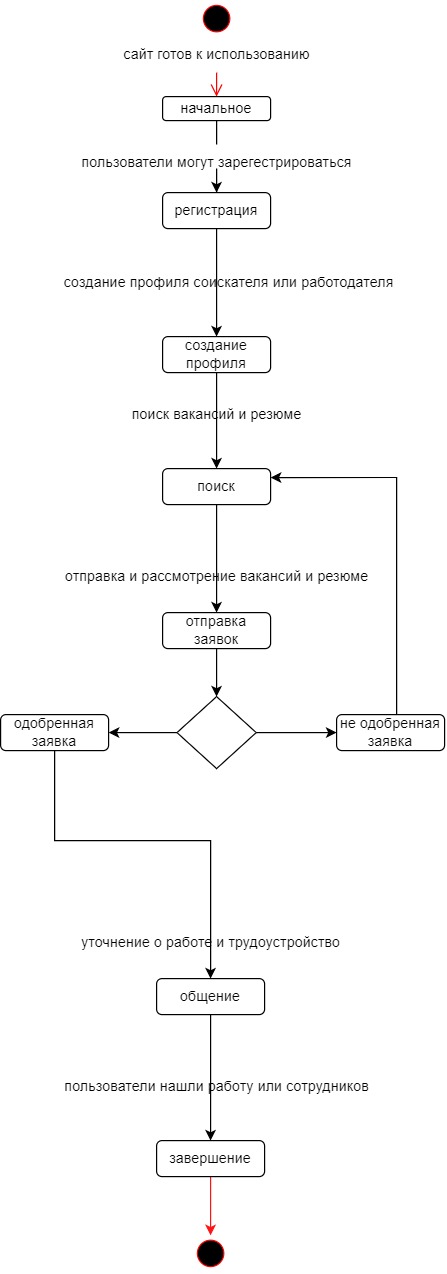


Рисунок 2 –Диаграмма состояний

Диаграмма деятельности UML – это инструмент моделирования системы, который отображает последовательность действий или операций, выполняемых в рамках процесса или деятельности в системе

Диаграмма деятельности помогает описать процессы и действия, которые будут выполняться в системе, и предоставляет визуальное представление последовательности выполнения действий. Она может использоваться для моделирования многих видов процессов, включая бизнес-процессы, процессы разработки программного обеспечения и т. д.

Диаграмма деятельности может быть полезна на разных стадиях разработки системы, от определения требований до тестирования и сопровождения. Она может помочь разработчикам и другим участникам проекта лучше понимать процессы и их взаимосвязь, а также использоваться для документирования процессов и общения между различными заинтересованными сторонами.

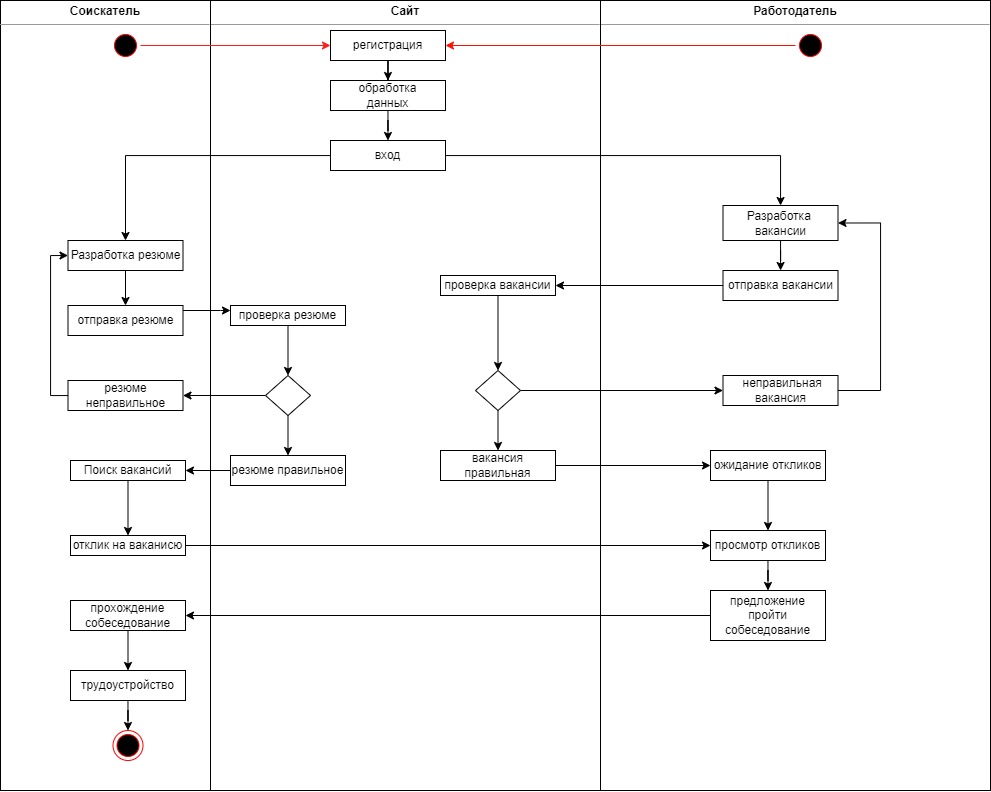
 Диаграмма деятельности (рисунок 3)

Рисунок 3 –Диаграмма деятельности

Диаграмма последовательности UML – это инструмент моделирования системы, который позволяет показать, как объекты взаимодействуют друг с другом в рамках определенной операции или процесса.

Диаграмма последовательности отображает последовательность шагов, которые объекты выполняют во время операции, и учитывает время, необходимое для выполнения каждого шага. Она используется для описания сценариев использования системы, где объективное поведение играет ключевую роль, а также для моделирования взаимодействия между объектами в системе.

Диаграмма последовательности может быть полезна для разработчиков, чтобы лучше понять взаимодействие между объектами и понять, как действовать в конкретных ситуациях. Она может помочь в разработке приложений, где важно понимание взаимодействия между объектами, особенно если это сложное взаимодействие.

Как и другие диаграммы UML, диаграмма последовательности может использоваться на разных этапах процесса разработки, от определения требований до создания кода и тестирования.

Диаграмма последовательности (рисунок 4)

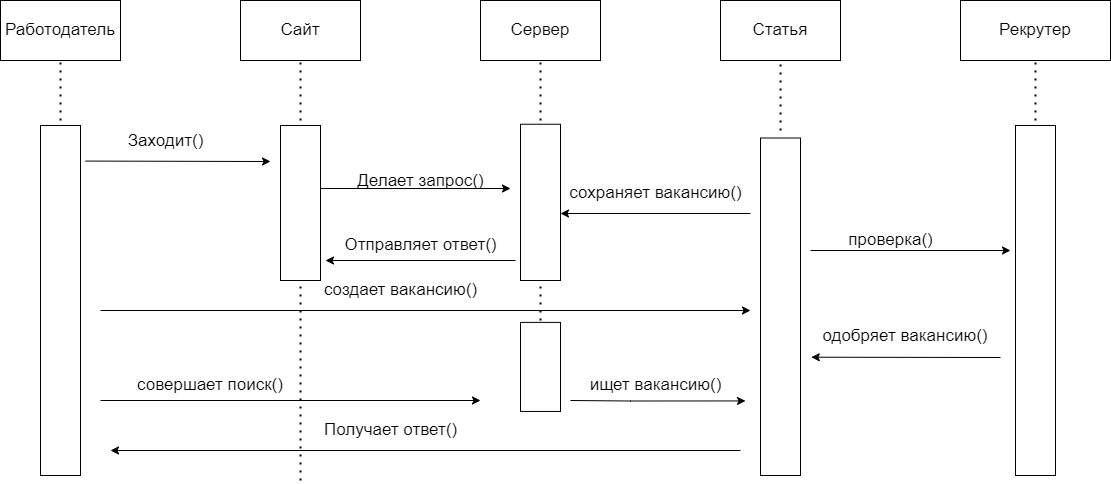


Рисунок 4 –Диаграмма последовательности

Диаграмма IDEF0 – это инструмент моделирования, который используется для моделирования процессов бизнеса, систем и организационных структур. IDEF0 была разработана в рамках Института технологии информации и является одной из самых старых и наиболее распространенных методологий моделирования, используемых в инженерии и управлении.

Диаграмма IDEF0 позволяет описывать бизнес-процессы и системы на высоком уровне абстракции, отображая взаимодействия между функциями и деятельностями, выполняемыми в рамках организации или системы. Она призвана помочь описать взаимодействия между людьми, технологиями, объектами и процессами.

Диаграмма IDEF0 может использоваться на ранних стадиях проекта, включая сбор требований и анализ бизнес-процессов, и может помочь в определении изменений в системе и проектировании оптимальных бизнес-процессов. Она также может быть использована в процессе документирования процессов и общения между участниками проекта.

В целом, диаграмма IDEF0 является мощным инструментом, который позволяет описывать сложные бизнес-процессы и системы на высоком уровне абстракции, что делает ее полезной для решения многих задач в области управления проектами и моделирования бизнес-процессов.

Диаграмма IDEF0 (рисунок 5)

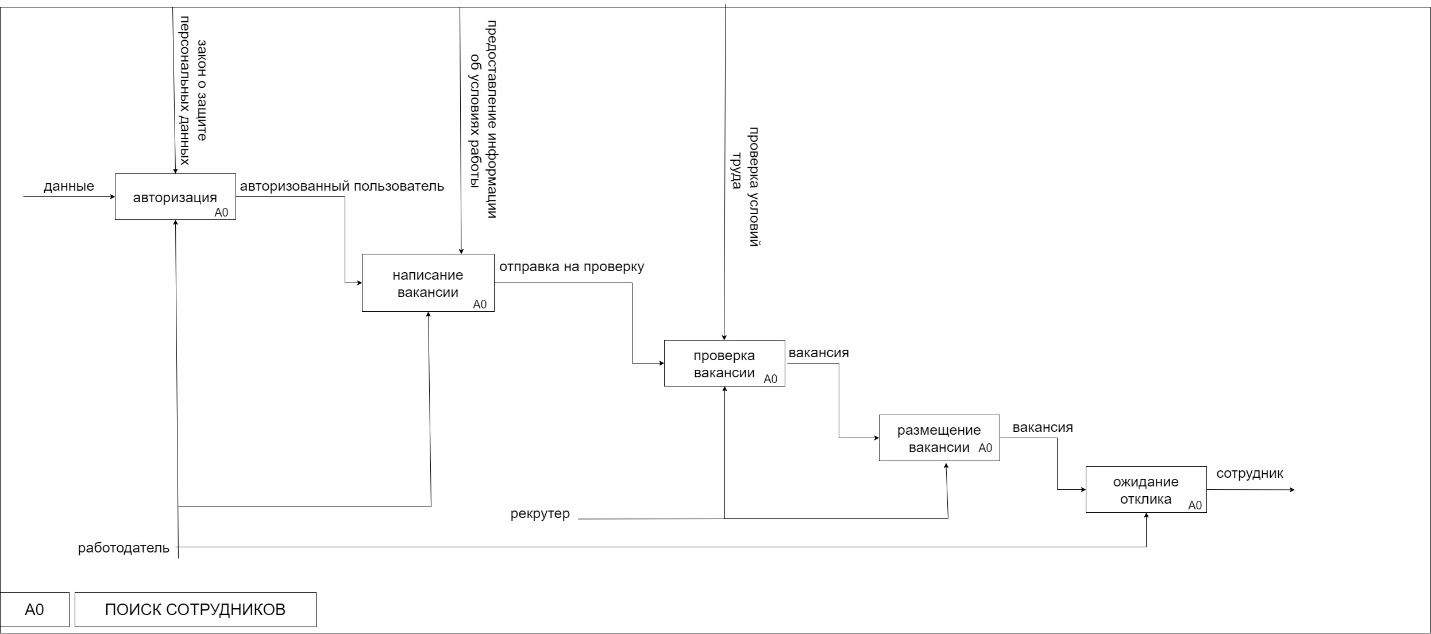


Рисунок 5 –Диаграмма IDEF0

# 1.3 Логическое и физическое проектирование баз данных

Логическое проектирование – создание схемы базы данных на основе конкретной модели базы данных, например, реляционной модели данных. Для реляционной модели данных даталогическая модель – набор схем отношений, обычно с указанием первичных ключей, а также «связей» между отношениями, представляющих собой внешние ключи. Преобразование концептуальной модели в логическую модель, как правило, осуществляется по формальным правилам. Этот этап может быть в значительной степени автоматизирован. На этапе логического проектирования учитывается специфика конкретной модели данных, но может не учитываться специфика конкретной СУБД.

Физическое проектирование базы данных - процесс создания описания реализации базы данных на вторичных запоминающих устройствах с указанием структур хранения и методов доступа, используемых для организации эффективной обработки данных

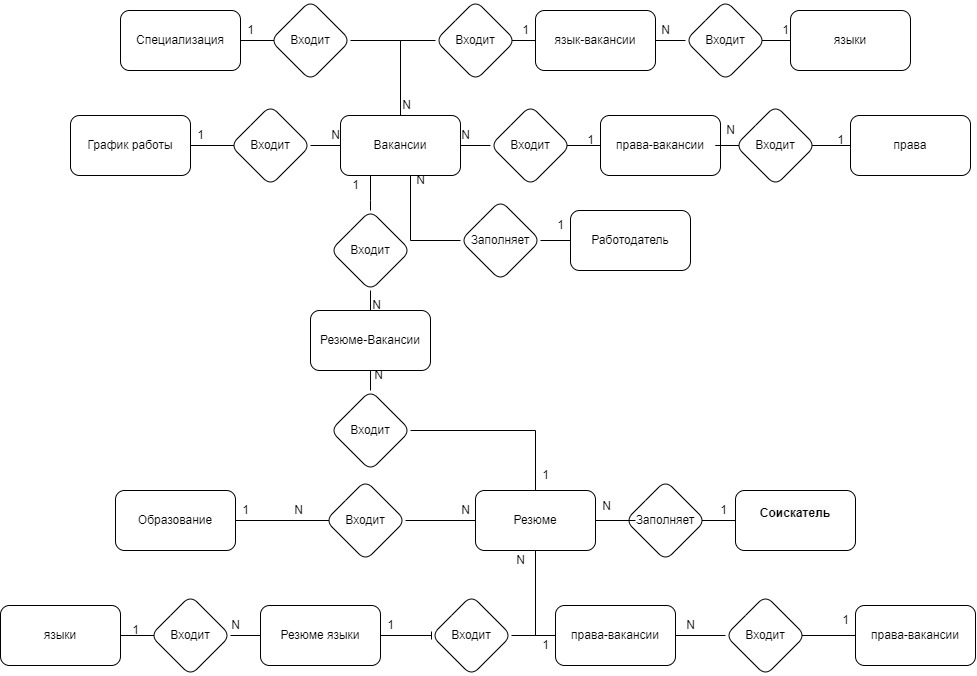
Физическое проектирование является третьей фазой процесса создания проекта базы данных, при выполнении которой проектировщик принимает решения о способах реализации разрабатываемой базы данных. Во время предыдущей фазы проектирования была определена логическая структура базы данных (т.е. набор ее сущностей, связей и атрибутов). Хотя эта структура не зависит от конкретной целевой СУБД, она создавалась с учетом выбранной модели хранения данных, например, реляционной, сетевой или иерархической (рисунок 6).

Рисунок 6 – Диаграмма сущность-связь

Сущность связь- модель данных, позволяющая описывать концептуальные схемы предметной области. С её помощью можно выделить ключевые сущности и обозначить связи, которые могут устанавливаться между этими сущностями. Во время проектирования баз данных происходит преобразование схемы, созданной на основе ER-модели, в конкретную схему базы данных на основе выбранной модели данных (реляционной, объектной, сетевой или др.).

Сущность в базе данных – это любой объект в базе данных, который можно выделить исходя из сути предметной области для которой разрабатывается эта база данных. Разработчик базы данных должен уметь правильно определять сущности.

Связь работает путем сопоставления данных в ключевых столбцах, обычно столбцах (или полях), которые имеют одно и то же имя в обеих таблицах. В большинстве случаев связь соединяет основной ключ или уникальный столбец идентификатора для каждой строки, от одной таблицы к полю в другой таблице.

Таблица applicant (соискатель) содержит информацию о соискателях и является главной по отношению к таблице resume(резюме).

Структура таблицы applicant:

* id – код категории (ключевое поле);
* Surname –фамилия соискателя;
* name – имя соискателя;
* patronymic– отчество соискателя;
* telephone –телефон соискателя;
* date\_birth – дата рождения;
* city\_residence – город проживания;
* paul-пол соискателя;
* citizenship –гражданство;
* role - роль;

Таблица education (образование) содержит информацию о образовании соискателя и является главной по отношению к таблице resume(резюме).

Структура таблицы education:

* id – код категории (ключевое поле);
* level –уровень образования;
* educational\_institution – Учебное заведение;
* faculty – Факультет;
* specialization – Специализация;
* year\_graduation – Год окончания;

Таблица [employer](http://127.0.0.1/openserver/phpmyadmin/index.php?route=/sql&pos=0&db=WORK&table=employer) (работодатель) содержит информацию о работодатели и является главной по отношению к таблице vacancy (вакансия).

Структура таблицы [employer](http://127.0.0.1/openserver/phpmyadmin/index.php?route=/sql&pos=0&db=WORK&table=employer):

* id – код категории (ключевое поле);
* organization – название организации;
* number – номер телефона;
* role – роль;
* Email – еmail;
* Password – пароль

Таблица language-resume(языки-резюме) содержит информацию о языке-резюме и является подчиненной по отношению к таблице resume(резюме).

Структура таблицы language-resume:

* id – код категории (ключевое поле);
* id\_resume – код резюме (поле связи с таблицей резюме);
* id\_language – код языка (поле связи с таблицей языки);

Таблица language-vacancies(языки-вакансий) содержит информацию о языке-вакансий и является подчиненной по отношению к таблице vacancy (вакансия).

Структура таблицы language-vacancies:

* id – код категории (ключевое поле);
* id\_vacancies – код вакансии (поле связи с таблицей вакансии);
* id\_language – код языка (поле связи с таблицей языки);

Таблица languages(языки) содержит информацию о языках и является подчиненной по отношению к таблицам language-resume(языки-резюме) и language-vacancies(языки-вакансий).

Структура таблицы languages:

* id – код категории (ключевое поле);
* languages – название языка;
* level – уровень владения языком;

Таблица resume (резюме) содержит информацию о резюме и является подчиненной по отношению к таблицам applicant (соискатель) и education (образование), и главной к таблице language-resume(языки-резюме), right-resume(права-резюме), resume-vacancy (резюме вакансии)

Структура таблицы resume:

* id – код категории (ключевое поле);
* work\_experience – опыт работы;
* desired\_position– желаемая должность;
* desired\_salary – желаемая зарплата;
* move – Переезд;
* busyness – Занятость;
* work\_schedule – График работы;
* work\_permit – разрешение на работу;
* availability\_car – Наличие автомобиля;
* id\_applicant – код соискателя (поле связи с таблицей соискателя);
* id\_education – код образование (поле связи с таблицей образование);

Таблица resume - vacancy (резюме вакансии) является подчиненной по отношению к таблицам resume (резюме) и vacancy (вакансия).

Структура таблицы education:

* id – код категории (ключевое поле);
* id\_resume –код резюме (поле связи с таблицей резюме);
* id\_vacancy – код вакансии (поле связи с таблицей вакансии);

Таблица right(права) содержит информацию о правах и является подчиненной по отношению к таблицам right-resume(права-резюме) и right -vacancies(права-вакансии).

Структура таблицы right:

* id – код категории (ключевое поле);
* category – категория прав;
* date\_rec –дата получения прав;

Таблица right-resume(права-резюме) содержит информацию о правах резюме и является подчиненной по отношению к таблице resume(резюме).

Структура таблицы right-resume:

* id – код категории (ключевое поле);
* id\_resume – код резюме (поле связи с таблицей резюме);
* id\_right– код прав (поле связи с таблицей права);

Таблица right -vacancies (права вакансий) содержит информацию о правах вакансий и является подчиненной по отношению к таблице vacancy (вакансия).

Структура таблицы language-vacancies:

* id – код категории (ключевое поле);
* id\_vacancies – код вакансии (поле связи с таблицей вакансии);
* id\_right– код прав (поле связи с таблицей права);

Таблица [specialization](http://127.0.0.1/openserver/phpmyadmin/index.php?route=/sql&pos=0&db=WORK&table=specialization) (специализация) содержит информацию о специализации вакансий и является главной по отношению к таблице vacancy (вакансия).

Структура таблицы [specialization](http://127.0.0.1/openserver/phpmyadmin/index.php?route=/sql&pos=0&db=WORK&table=specialization):

* id – код категории (ключевое поле);
* title– название специализации;
* discharge– разряд специализации;

Таблица vacancy (вакансия) содержит информацию о вакансии и является главной по отношению к таблицам right -vacancies (права вакансий), language-vacancies(языки-вакансий), resume-vacancy (резюме вакансии), [specialization](http://127.0.0.1/openserver/phpmyadmin/index.php?route=/sql&pos=0&db=WORK&table=specialization) (специализация) и подчиненной к таблицам [work-schedule](http://127.0.0.1/openserver/phpmyadmin/index.php?route=/sql&pos=0&db=WORK&table=work-schedule) (график работы), [employer](http://127.0.0.1/openserver/phpmyadmin/index.php?route=/sql&pos=0&db=WORK&table=employer) (работодатель).

Структура таблицы vacancy:

* id – код категории (ключевое поле);
* title – Название вакансии;
* id\_specialization – код специализации (поле связи с таблицей специализация);
* income\_level – Уровень дохода;
* place\_work – место работы;
* description\_vacancy – Описание вакансии;
* work\_experience – Требуемый опыт работы;
* employee\_responsibilities – Обязанности сотрудника;
* key\_skills – Ключевые навыки;
* id\_work\_schedule – код Графика работы (поле связи с таблицей график работы);
* id\_employer – код работодателя (поле связи с таблицей работодатель);

Таблица [work-schedule](http://127.0.0.1/openserver/phpmyadmin/index.php?route=/sql&pos=0&db=WORK&table=work-schedule) (график работы) содержит информацию о графике работе вакансии и является главной по отношению к таблице vacancy (вакансия).

Структура таблицы [specialization](http://127.0.0.1/openserver/phpmyadmin/index.php?route=/sql&pos=0&db=WORK&table=specialization):

* id – код категории (ключевое поле);
* operating\_mode– режим работы;
* work\_schedule– график работы;

В результате анализа предметной области была сформулирована логическая структура, схема проектируемой базы данных (рисунок 7). Построенная логическая модель в дальнейшем будет востребована на этапе физического проектирования, а также, например, представлять любые изменения, вносимые в базу данных.

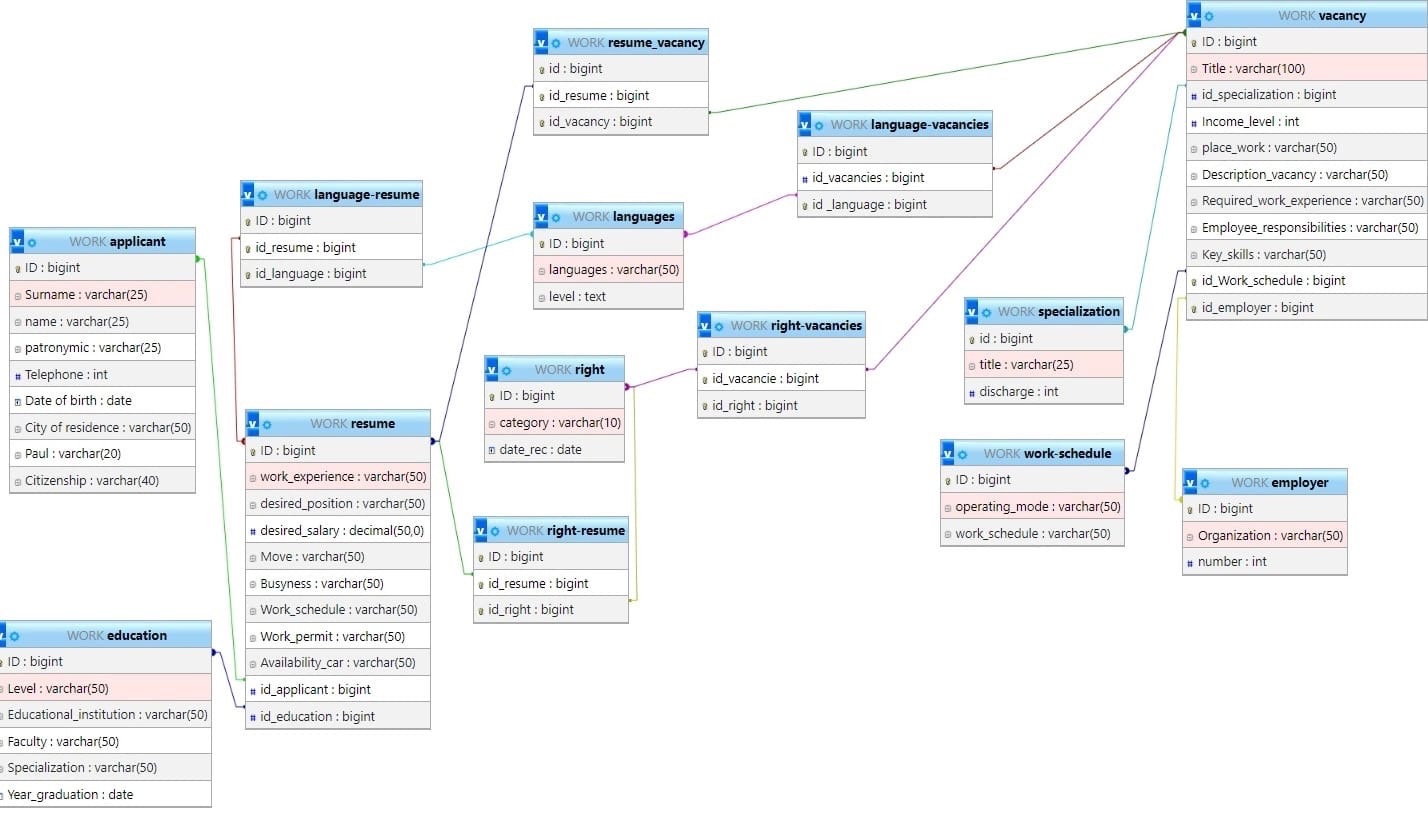


Рисунок 7 Схема базы данных

# 1.4 Выбор и обоснование средств разработки

Для разработки информационной системы необходимо выбрать средства разработки, которые позволят создать эффективную и удобную систему. Решил выбрать средства разработки Php и JavaScript и MySQL. Эти средства разработки являются одними из наиболее популярных и распространенных в веб-разработке. Ниже приведены основания для выбора каждого из них:

PHP - это язык программирования, который используется для создания динамических веб-сайтов и приложений. Он может выполнять различные задачи, такие как обработка форм, работа с базами данных, отправка и получение электронной почты и многое другое.

PHP помогает при создании сайта, потому что он позволяет создавать динамические страницы, которые могут изменяться в зависимости от действий пользователя или других переменных. Это позволяет создавать интерактивные сайты, которые могут предоставлять пользователю более персонализированный опыт.

Кроме того, PHP имеет большое сообщество разработчиков, которые создают библиотеки и фреймворки для упрощения разработки веб-приложений. Это делает PHP одним из наиболее популярных языков программирования для веб-разработки.

Php требуется для выполнения следующих задач:

– Взаимодействие с базами данных: с помощью PHP подключаться к базам данных, извлекать данные и использовать их на веб-странице.

– Создание пользовательского интерфейса: использовать PHP для создания интерактивных элементов на сайте, таких как кнопки, меню и модальные окна.

– Обработка сессий: PHP буду использовать для обработки сессий и хранения информации о пользователе, которая может быть использована на следующих страницах.

– Создание динамических страниц: Веб-страницы могут быть изменены с помощью PHP-скриптов. Это может использоваться для генерации динамических страниц на основе информации пользователя или других факторов.

Используя PHP во время создания сайта, могу создать большое количество интерактивных и динамических элементов для пользователей.

JavaScript - это язык программирования, который используется для создания интерактивных веб-сайтов и приложений. JavaScript позволяет добавлять динамические элементы на веб-страницы, такие как анимации, валидацию форм, обработку событий и многое другое.

JavaScript может быть использован для создания различных компонентов веб-сайта, таких как слайдеры, выпадающие меню, модальные окна и т.д. Он также может быть использован для создания веб-приложений, которые работают на стороне клиента, таких как игры и социальные сети.

В целом, JavaScript является неотъемлемой частью веб-разработки и помогает создавать более интерактивные и функциональные веб-сайты.

JavaScript требуется для выполнения следующих задач:

– Валидацию форм: JavaScript может использоваться для проверки данных, введенных в форму пользователем, и обеспечения их правильности.

– Создание анимации: JavaScript может использоваться для создания анимированных элементов на веб-страницах.

– Управление стилями и содержимым: JavaScript может изменять содержимое и стили элементов на веб-странице, что позволяет сделать сайт динамичнее и привлекательнее для пользователей.Vue.js (произносится "вью джей эс") - это прогрессивный JavaScript-фреймворк, который используется для разработки пользовательских интерфейсов веб-приложений. Он является одним из наиболее популярных фреймворков JavaScript и обеспечивает создание одностраничных приложений (SPA) и многокомпонентных интерфейсов.

Основные преимущества Vue.js:

Легкость использования: Vue.js предоставляет простую и интуитивно понятную синтаксическую структуру, что делает его отличным выбором для начинающих разработчиков. Он также легко интегрируется с существующими проектами и другими библиотеками JavaScript.

Реактивность: Vue.js обладает реактивной архитектурой, которая автоматически отслеживает изменения данных и обновляет соответствующие компоненты без необходимости явного обновления. Это позволяет создавать динамические интерфейсы, которые моментально откликаются на пользовательские действия.

Компонентный подход: Vue.js основан на компонентах, которые являются независимыми и переиспользуемыми блоками кода. Компоненты позволяют создавать иерархическую структуру приложения, что упрощает его разработку, тестирование и поддержку.

Гибкость: Vue.js предоставляет гибкие инструменты для создания пользовательского интерфейса. Он поддерживает использование шаблонов HTML, компонентные стили и директивы, что позволяет разработчикам выбирать подходящий под их потребности способ разработки.

Vue.js также обладает обширной экосистемой, включающей в себя плагины, библиотеки и инструменты разработки, которые значительно упрощают процесс разработки и расширяют возможности фреймворка.

MySQL – это система управления реляционными базами данных (СУБД), которая используется для хранения и организации данных веб-сайтов и приложений. Я выбрал MySQL, потому что это одна из самых популярных и надежных СУБД, которая имеет открытый исходный код и поддерживает большое количество функций.

MySQL обладает высокой производительностью и масштабируемостью, что делает его идеальным выбором для веб-приложений любого размера. Он также легко интегрируется с другими языками программирования, такими как PHP, что делает его очень удобным для использования в веб-разработке.

Кроме того, MySQL имеет обширную документацию и большое сообщество разработчиков, которые создают библиотеки и расширения для упрощения работы с базами данных. Это делает MySQL одним из наиболее популярных выборов для веб-разработчиков.

MySQL требуется для выполнения следующих задач:

– Управление базой данных. Использовать MySQL для создания, изменения и управления базами данных, которые будут использоваться на веб-сайте.

– Хранение данных. Использовать MySQL для хранения и организации данных, которые будут использоваться на веб-сайте, таких как пользовательские данные и данные о продуктах.

– Выполнение запросов. Использовать MySQL для выполнения запросов к базе данных, чтобы извлечь и изменить данные.

– Управление пользовательскими данными. Использовать MySQL для хранения пользовательских данных, таких как логины, пароли и другие данные, используемые на веб-сайте.

– Работа с большим объемом данных. MySQL может быть использован для управления большим объемом данных, что позволяет использовать его для масштабирования веб-сайта при необходимости.

PHP, JavaScript и MySQL – это три основных компонента, используемых при создании динамических веб-сайтов. PHP является серверным языком программирования, который используется для создания динамических страниц и обработки данных на стороне сервера. JavaScript используется для создания интерактивных элементов на веб-страницах и обработки событий на стороне клиента. MySQL - это система управления базами данных, которая используется для хранения и управления данными на сервере.

Вместе эти три компонента позволяют создавать мощные веб-приложения и сайты с динамическим содержимым. PHP обрабатывает запросы от клиентов и генерирует динамические страницы, используя данные из базы данных MySQL. JavaScript добавляет интерактивность и динамические элементы на страницы, такие как анимации, выпадающие меню и модальные окна.

Кроме того, PHP и MySQL позволяют создавать различные функциональные компоненты сайта, такие как формы обратной связи, авторизация пользователей, управление контентом и многое другое. JavaScript может использоваться для улучшения пользовательского опыта, например, с помощью AJAX, который позволяет обновлять содержимое страницы без перезагрузки.

Комбинация PHP, JavaScript и MySQL является широко используемой тройкой в веб-разработке. PHP используется на сервере для генерации HTML-страниц и обработки данных, которые отправляют клиенты. JavaScript используется на стороне клиента для создания интерактивности веб-страниц и отправки асинхронных запросов на сервер. MySQL используется для хранения данных, которые используются на сайте.

Примером применения PHP, JavaScript и MySQL может быть создание сайта интернет-магазина. PHP может быть использован для генерации страниц с продуктами и обработки заказов. JavaScript может быть использован для создания интерфейса взаимодействия с пользователем, например, для создания выпадающих списков или формы заказа. MySQL может быть использована для хранения информации о продуктах и заказах пользователей.

В целом, использование PHP, JavaScript и MySQL вместе позволяет создавать более функциональные, интерактивные и динамические веб-сайты, которые улучшают пользовательский опыт и повышают эффективность работы сайта.

В заключение, PHP, JavaScript и MySQL -–это мощные инструменты, которые могут помочь успешно создать веб-сайт или веб-приложение. Их использование может обеспечить высокую производительность и надежность, а также расширить возможности для создания интерактивного пользовательского опыта.

# РАЗРАБОТКА ИНФОРМАЦИОННОЙ СИСТЕМЫ

# 2.1 Создание базы данных

База данных - это организованная структура, предназначенная для хранения, изменения и обработки взаимосвязанной информации, преимущественно больших объемов. Базы данных активно используются для динамических сайтов со значительными объемами данных — часто это интернет-магазины, порталы, корпоративные сайты. В данном случае это база данных для детского сада.

Одной из целей данного курсового проекта явилось создание базы данных. База данных должна обеспечивать удобность и отвечать за скорость ориентирования на сайте, чтобы вся информация была структурирована и доступна.

Данная база данных представляет собой информацию о работодателях и их вакансиях и соискателей с их резюме

Создание таблицы «соискатель». Характеристики таблицы представлены в   
таблице 1.

CREATE TABLE `applicant` (

` id ` bigint NOT NULL,

`surname` varchar (25) NOT NULL,

`name` varchar (25) NOT NULL,

`patronymic` varchar (25) NOT NULL,

`telephone` int NOT NULL,

`date\_birth` date NOT NULL,

`city\_residence` varchar (50) NOT NULL,

`paul` varchar (20) NOT NULL,

`citizenship` varchar (40) NOT NULL

PRIMARY KEY (`id `);

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

Таблица 1- Характеристики полей таблицы «applicant»( соискатель)

|  |  |  |
| --- | --- | --- |
| Имя столбца | Тип данных | Свойство поля |
| id | BIGINT(11) | Ключевое поле |
| Обязательное поле |
| счетчик |
| surname | VARCHAR(25) | Обязательное поле |
| name | VARCHAR(25) | Обязательное поле |
| patronymic | VARCHAR(25) | Обязательное поле |
| telephone | INT | Обязательное поле |
| Размер поля MAX |
| date\_birth | Date | Обязательное поле. |
| city\_residence | VARCHAR(50) | Обязательное поле |
| paul | VARCHAR(20) | Обязательное поле |
| citizenship | VARCHAR(40) | Обязательное поле |
| role | VARCHAR(25) | Обязательное поле |
| Email | VARCHAR(50) | Обязательное поле |
| password | VARCHAR(255) | Обязательное поле |

Создание таблицы «Образование» Характеристики таблицы представлены в таблице 2

CREATE TABLE `education` (

` id ` bigint NOT NULL,

`level` varchar(50) NULL,

`educational\_institution` varchar (50) CHARACTER SET utf8mb4 COLLATE utf8mb4\_0900\_ai\_ci NULL,

`faculty` varchar(50) NULL,

`specialization` varchar(50) NULL,

`year\_graduation` date NULL

PRIMARY KEY (`id `);

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

Таблица 2 Характеристики полей таблицы «education»( Образование)

|  |  |  |
| --- | --- | --- |
| Имя столбца | Тип данных | Свойство поля |
| id | BIGINT | Ключевое поле |
| Обязательное поле |
| счетчик |
| level | VARCHAR(50) | Не обязательное поле |
| educational\_institution | VARCHAR(50) | Не обязательное поле |
| faculty | VARCHAR(50) | Не обязательное поле |
| specialization | VARCHAR(50) | Не обязательное поле |
| year\_graduation | DATE | Не обязательное поле |

Создание таблицы «Работодатель» Характеристики таблицы представлены в таблице 3

CREATE TABLE `employer` (

` id ` bigint NOT NULL,

`organization` varchar(50) NOT NULL,

`number` int NOT NULL

PRIMARY KEY (`id `);

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

Таблица 3 Характеристики полей таблицы «employer»( Работодатель)

|  |  |  |
| --- | --- | --- |
| Имя столбца | Тип данных | Свойство поля |
| id | BIGINT | Ключевое поле |
| Обязательное поле |
| счетчик |
| organization | VARCHAR(50) | Обязательное поле |
| number | INT | Обязательное поле |
| Размер поля MAX |
| role | VARCHAR(25) | Обязательное поле |
| Email | VARCHAR(50) | Обязательное поле |
| password | VARCHAR(255) | Обязательное поле |

Создание таблицы «Язык-резюме» Характеристики таблицы представлены в таблице 4

CREATE TABLE `language-resume` (

` id ` bigint NOT NULL,

`id\_resume` bigint NULL,

`id\_language` bigint NULL

PRIMARY KEY (`id `);

FOREIGN KEY (`id\_resume`, `id\_language`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

Таблица 4 Характеристики полей таблицы «language-resume» (Язык-резюме)

|  |  |  |
| --- | --- | --- |
| Имя столбца | Тип данных | Свойство поля |
| id | BIGINT | Ключевое поле |
| Не обязательное поле |
| счетчик |
| id\_resume | BIGINT | Не обязательное поле |
| id\_language | BIGINT | Не обязательное поле |

Создание таблицы «Язык-вакансия» Характеристики таблицы представлены в таблице 5

CREATE TABLE `language-vacancies` (

` id ` bigint NOT NULL,

`id\_vacancies` bigint NULL,

`id \_language` bigint NULL

PRIMARY KEY (`id `);

FOREIGN KEY (`id\_ vacancies `, `id\_language`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

Таблица 5 Характеристики полей таблицы «language-vacancies» (Язык-вакансия)

|  |  |  |
| --- | --- | --- |
| Имя столбца | Тип данных | Свойство поля |
| id | BIGINT | Ключевое поле |
| Не обязательное поле |
| счетчик |
| id\_vacancies | BIGINT | Не обязательное поле |
| id\_language | BIGINT | Не обязательное поле |

Создание таблицы «Язык» Характеристики таблицы представлены в таблице 6

CREATE TABLE `languages` (

` id ` bigint NOT NULL,

`languages` varchar (50) NULL,

`level` varchar (50) NULL

PRIMARY KEY (`id `);

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

Таблица 6 Характеристики полей таблицы «language» (Язык)

|  |  |  |
| --- | --- | --- |
| Имя столбца | Тип данных | Свойство поля |
| id | BIGINT | Ключевое поле |
| Обязательное поле |
| счетчик |
| languages | VARCHAR(50) | Не обязательное поле |
| level | VARCHAR(50) | Не обязательное поле |

Создание таблицы «Резюме» Характеристики таблицы представлены в таблице 7

CREATE TABLE `resume` (

`id` bigint NOT NULL,

`work\_experience` varchar (50) CHARACTER SET utf8mb4 COLLATE utf8mb4\_0900\_ai\_ci NOT NULL,

`desired\_position` varchar (50) CHARACTER SET utf8mb4 COLLATE utf8mb4\_0900\_ai\_ci NOT NULL,

`desired\_salary` decimal (50,0) NOT NULL,

`move` varchar (50) NOT NULL,

`busyness` varchar (50) NOT NULL,

`work\_schedule` varchar (50) CHARACTER SET utf8mb4 COLLATE utf8mb4\_0900\_ai\_ci NOT NULL,

`work\_permit` varchar (50) CHARACTER SET utf8mb4 COLLATE utf8mb4\_0900\_ai\_ci NULL,

`availability\_car` varchar (50) CHARACTER SET utf8mb4 COLLATE utf8mb4\_0900\_ai\_ci NULL,

`id\_applicant` bigint NOT NULL,

`id\_education` bigint NOT NULL

PRIMARY KEY (`id `);

FOREIGN KEY (`id\_applicant `, ` id\_education `)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

Таблица 7 Характеристики полей таблицы «resume» (Резюме)

|  |  |  |
| --- | --- | --- |
| Имя столбца | Тип данных | Свойство поля |
| id | BIGINT | Ключевое поле |
| Обязательное поле |
| счетчик |
| work\_experience | VARCHAR(50) | Обязательное поле |
| desired\_position | VARCHAR(50) | Обязательное поле |
| desired\_salary | DECIMAL (50,0) | Обязательное поле |
| move | VARCHAR(50) | Обязательное поле |
| busyness | VARCHAR(50) | Обязательное поле |
| work\_schedule | VARCHAR(50) | Обязательное поле |
| work\_permit | VARCHAR(50) | Не обязательное поле |
| availability\_car | VARCHAR(50) | Не обязательное поле |
| id\_applicant | BIGINT | Обязательное поле |
| id\_education | BIGINT | Обязательное поле |

Создание таблицы «Резюме-Вакансия«Характеристики таблицы представлены в таблице 8

CREATE TABLE `resume\_vacancy` (

`id` bigint NOT NULL,

`id\_resume` bigint NULL,

`id\_vacancy` bigint NULL

PRIMARY KEY (`id `);

FOREIGN KEY (`id\_vacancy `)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

Таблица 8 Характеристики полей таблицы «resume\_vacancy» (Резюме-Вакансия)

|  |  |  |
| --- | --- | --- |
| Имя столбца | Тип данных | Свойство поля |
| id | BIGINT | Ключевое поле |
| Обязательное поле |
| счетчик |
| id\_resume | VARCHAR(50) | Ключевое поле |
| Не обязательное поле |
| счетчик |
| id\_vacancy | BIGINT | Не обязательное поле |

Создание таблицы «права» Характеристики таблицы представлены в таблице 9

CREATE TABLE `right` (

` id ` bigint NOT NULL,

`category` varchar(10) NULL,

`date\_rec` date NULL

PRIMARY KEY (`id `);

)ENGINE= InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

Таблица 9 Характеристики полей таблицы «right» (права)

|  |  |  |
| --- | --- | --- |
| Имя столбца | Тип данных | Свойство поля |
| id | BIGINT | Ключевое поле |
| Обязательное поле |
| счетчик |
| category | VARCHAR(10) | Не обязательное поле |
| date\_rec | DATE | Не обязательное поле |

Создание таблицы «права-резюме» Характеристики таблицы представлены в таблице 10

CREATE TABLE `right-resume` (

` id ` bigint NOT NULL,

`id\_resume` bigint NULL,

`id\_right` bigint NULL

PRIMARY KEY (`id `);

FOREIGN KEY (`id\_right `, ` id\_resume `)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

Таблица 10 Характеристики полей таблицы «right-resume» (права-резюме)

|  |  |  |
| --- | --- | --- |
| Имя столбца | Тип данных | Свойство поля |
| id | BIGINT | Ключевое поле |
| Обязательное поле |
| счетчик |
| id\_resume | BIGINT | Не обязательное поле |
| id\_right | BIGINT | Не обязательное поле |

Создание таблицы «права-вакансии» Характеристики таблицы представлены в таблице 11

CREATE TABLE `right-vacancies` (

`id` bigint NOT NULL,

`id\_vacancie` bigint NULL,

`id\_right` bigint NULL

PRIMARY KEY (`id `);

FOREIGN KEY (`id\_vacancie `, ` id\_right `)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

Таблица 11 Характеристики полей таблицы «right-vacancies» (права- вакансии)

|  |  |  |
| --- | --- | --- |
| Имя столбца | Тип данных | Свойство поля |
| id | BIGINT | Ключевое поле |
| Обязательное поле |
| счетчик |
| id\_vacancie | BIGINT | Не обязательное поле |
| id\_right | BIGINT | Не обязательное поле |

Создание таблицы «специализация» Характеристики таблицы представлены в таблице 12

CREATE TABLE `specialization` (

`id` bigint NOT NULL,

`title` varchar (25) NULL,

`discharge` int NULL

PRIMARY KEY (`id `);

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

Таблица 11 Характеристики полей таблицы «specialization» (специализация)

|  |  |  |
| --- | --- | --- |
| Имя столбца | Тип данных | Свойство поля |
| id | BIGINT | Ключевое поле |
| Обязательное поле |
| счетчик |
| title | VARCHAR(25) | Не обязательное поле |
| discharge | INT | Не обязательное поле |
| Размер поля MAX |

Создание таблицы «Вакансия» Характеристики таблицы представлены в таблице 13

CREATE TABLE `vacancy` (

`ID` bigint NOT NULL,

`Title` text NOT NULL,

`id\_specialization` bigint NOT NULL,

`income\_level` int NOT NULL,

`place\_work` varchar (50) CHARACTER SET utf8mb4 COLLATE utf8mb4\_0900\_ai\_ci NOT NULL,

`description\_vacancy` varchar (50) CHARACTER SET utf8mb4 COLLATE utf8mb4\_0900\_ai\_ci NOT NULL,

`work\_experience` varchar (50) CHARACTER SET utf8mb4 COLLATE utf8mb4\_0900\_ai\_ci NOT NULL,

`employee\_responsibilities` varchar (50) CHARACTER SET utf8mb4 COLLATE utf8mb4\_0900\_ai\_ci NOT NULL,

`key\_skills` varchar (50) CHARACTER SET utf8mb4 COLLATE utf8mb4\_0900\_ai\_ci NOT NULL,

`id\_Work\_schedule` bigint NOT NULL,

`id\_employer` bigint NOT NULL

PRIMARY KEY (`id `);

FOREIGN KEY (`id\_specialization `, ` id\_Work\_schedule `, ` id\_employer`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

Таблица 13 Характеристики полей таблицы «vacancy» (Вакансия)

|  |  |  |
| --- | --- | --- |
| Имя столбца | Тип данных | Свойство поля |
| id | BIGINT | Ключевое поле |
| Обязательное поле |
| счетчик |
| title | TEXT | Обязательное поле |
| Размер поля MAX |
| id\_specialization | BIGINT | Обязательное поле |
| income\_level | INT | Обязательное поле |
| Размер поля MAX |
| place\_work | VARCHAR(50) | Обязательное поле |
| description\_vacancy | VARCHAR(50) | Обязательное поле |
| required\_work\_experience | VARCHAR(50) | Обязательное поле |
| employee\_responsibilities | VARCHAR(50) | Обязательное поле |
| key\_skills | VARCHAR(50) | Обязательное поле |
| id\_Work\_schedule | BIGINT | Обязательное поле |
| id\_employer | BIGINT | Обязательное поле |

Создание таблицы «График работы» Характеристики таблицы представлены в таблице 14

CREATE TABLE `work-schedule` (

`id` bigint NOT NULL,

`operating\_mode` varchar(50) CHARACTER SET utf8mb4 COLLATE utf8mb4\_0900\_ai\_ci NOT NULL,

`work\_schedule` varchar(50) CHARACTER SET utf8mb4 COLLATE utf8mb4\_0900\_ai\_ci NOT NULL

PRIMARY KEY (`id `);

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

Таблица 14 Характеристики полей таблицы «work-schedule» (График работы)

|  |  |  |
| --- | --- | --- |
| Имя столбца | Тип данных | Свойство поля |
| id | BIGINT | Ключевое поле |
| Обязательное поле |
| счетчик |
| operating\_mode | VARCHAR(50) | Обязательное поле |
| work\_schedule | VARCHAR(50) | Обязательное поле |

# Логическая и физическая структура сайта

Логическая структура сайта - это система ссылочного взаимодействия между страницами виртуального ресурса. Она должна продумываться на самых ранних этапах разработки сайта и являться основой функциональности дизайна.

Логическая структура сайта - это не совокупность разрозненных страниц и графических ресурсов, а заранее структурированный контент с перекрестными ссылками между всеми страницами ресурса. Если структура реализована правильно, то эффективность веб-ресурса будет высокой.

Если структура сайта разработана правильно, то она будет служить для посетителя проводником к информации, выложенной на ресурсе, и являющейся главным фактором, определяющим поведение пользователя. Разрабатывается она, как правило, один раз, и в последующем остается без изменений, поэтому данному разделу сайтостроения необходимо уделить особое внимание. Дело в том, что после индексации поисковиками вашего веб-сайта, адреса его страниц заносятся в базу данных поисковой системы. Если вы измените путь доступа хотя бы к некоторым разделам уже после индексации, то есть вероятность, что ваш ресурс просто потеряется для ПС, и не будет выводиться на прежней позиции по запросу поисковика. Многие специалисты сайтостроения рекомендуют использовать линейный пример структуры сайта, который очень прост для понимания и легко индексируется.

Физическая структура подразумевает алгоритм размещения физических файлов по поддиректориям папки, в которой опубликован ваш сайт. (рис. 8).



Рисунок 8 – Физическая структура сайта

Таким образом выглядит директория с базой данных и функциями по ее управлению (рис. 9)

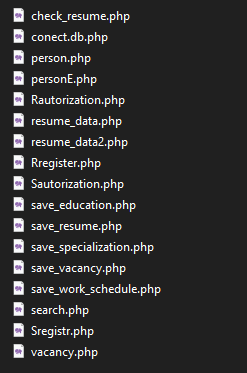


Рисунок 9 – Директория с базой данных

# 2.3 Разработка интерфейса пользователя

Веб-интерфейс - веб-страница или совокупность веб-страниц, предоставляющая пользовательский интерфейс для взаимодействия с сервисом или устройством посредством протокола HTTP и веб-браузера. Веб-интерфейсы получили широкое распространение в связи с ростом популярности всемирной паутины и соответственно — повсеместного распространения веб-браузеров. (рис. 10)

Одним из основных требований к веб-интерфейсам является их одинаковый внешний вид и одинаковая функциональность при работе в различных браузерах.

Классическим и наиболее популярным методом создания веб-интерфейсов является использование HTML с применением CSS и JavaScript'a. Однако различная реализация HTML, CSS, DOM и других спецификаций в браузерах вызывает проблемы при разработке веб-приложений и их последующей поддержке. Кроме того, возможность пользователя настраивать многие параметры браузера (например, размер шрифта, цвета, отключение поддержки сценариев) может препятствовать корректной работе интерфейса.

Другой подход заключается в использовании Adobe Flash, Silverlight или Java-апплетов для полной или частичной реализации пользовательского интерфейса. Поскольку большинство браузеров поддерживает эти технологии (как правило, с помощью плагинов), Flash- или Java-приложения могут выполняться с легкостью. Так как они предоставляют программисту больший контроль над интерфейсом, они способны обходить многие несовместимости в конфигурациях браузеров, хотя несовместимость между Java или Flash реализациями на стороне клиента может приводить к различным осложнениям.

UI-дизайн – User Interface, дословно переводится, как «Пользовательский интерфейс». Это определенный процесс визуализации, который позволяет реализовать прототип сайта, приложения или другого веб-ресурса. В первую очередь, UI включает активную работу над графической составляющей интерфейсом. Создаются анимации, иллюстрации, кнопки и другие элементы сайта, включая шрифты, цвета, формы.

Первоначально определяется пользовательский опыт и исследуется целевая аудитория. В зависимости от потребностей посетителей определяется цветовая палитра, формы, а также структура размещения объектов. Интерфейс пользователя должен быть понятным, и одна из главных задач UI-дизайнера – определить, удобен ли сайт визуально, легко ли попасть по кнопкам, хорошо ли читается текст.

Что такое интерфейс? Это набор, инструмент графической структуры приложения. Он полностью состоит из кнопок, текстов, изображений, полей, и прочих элементов. От их внешнего вида, удобства использования зависит конверсия сайта. Поэтому интернет магазины с плохим интерфейсом пользователя просто теряют своих клиентов.

UI-дизайнер делает так, чтобы взаимодействие пользователя с сайтом было гармоничным и эффективным. Проектирование пользовательского интерфейса строится на создании понятного визуала, чтобы посетитель сразу понял, как пользоваться продуктом (сайтом, интернет магазином).

Проектирование интерфейсов нужно не только для визуального оформления сайта. В первую очередь, это возможность удобно взаимодействовать с приложением и улучшать пользовательский опыт. UI нужен для выполнения следующих задач:

Привлечение посетителей. Пользователь оценивает внешний вид сайта и удобство его использования в течение нескольких секунд. Если посетитель сразу находит то, зачем он пришел, его устраивает визуализация и вообще, взаимодействие понятное и простое, он останется. В противном случае сайт просто закрывается, и посетитель уходит к конкурентам.

Ускорение превращения посетителя в клиента. Если форма заказа простая, сайт стильно оформлен, вызывает доверие и положительные эмоции – повышается конверсия продаж. Клиенту приятнее находиться на страницах такого ресурса, нежели на сайте с перегруженным или неграмотным пользовательским интерфейсом.

Адаптация под различные устройства. Разработка интерфейса подразумевает создание адаптивного дизайна, который одинаково хорошо будет смотреться на различных устройствах. Владельцы некоторых интернет магазинов не задумываются, как их ресурс будет отображаться на мобильных устройствах, хотя зачастую, преимущественная часть трафика идет именно со смартфонов.

Повышение уровня доверия. Создание фирменных цветов, шрифтов и других графических элементов делает интернет магазин узнаваемым и повышает уровень доверия пользователей. Постоянство – залог хорошего взаимодействия с клиентом.

UI нужен для того, чтобы наладить хорошие взаимоотношения с посетителем. Только так он становится клиентом. Зачастую владельцы интернет магазинов стараются сэкономить, отказываясь от услуг дизайнера и разработки пользовательского интерфейса, но такое решение приводит к большой ошибке. Сайт готов, реклама настроена, а продаж нет. А все потому, что пользователям просто неудобно или не нравится находиться на сайте. Но даже в таком случае можно все исправить.

Многие путают UI и UX. Грань действительно тонкая, как правило, происходит взаимодействие UI и UX. Последний нужен для того, чтобы понять пользователя, а UI помогает применить полученные данные и сделать так, чтобы посетитель легко разобрался с продуктом.

Во время разработки пользовательского интерфейса необходимо детально изучить целевую аудиторию и удовлетворить все ее потребности. Поэтому разработка UI состоит из нескольких этапов:

Опираясь на исследования целевой аудитории, проанализировав потребности, поведение и желания клиента, дизайнер ставит задачи. Продумывается концепция пользовательского интерфейса, будущие элементы, цвета. Этот этап очень важен, и занимает значительную часть времени. Но именно во время разработки концепции решаются наиболее важные задачи.

Данный этап еще до разработки пользовательского интерфейса позволяет понять, как клиент будет видеть сайт. Тут же вносятся все изменения. Первоначально размечается расположение всех графических элементов, включая кнопки, формы, меню. Далее подбирается цветовая палитра с обязательным условием – все цвета должны быть гармоничны между собой. Подбираются изображения, их стилистика, и в итоге – создается макет.

Когда разрабатывается интерфейс приложения, дизайнер ориентируется на специальные правила оформления элементов, а также регламент, созданный компаниями Google и Apple. Поэтому труд немного облегчается.

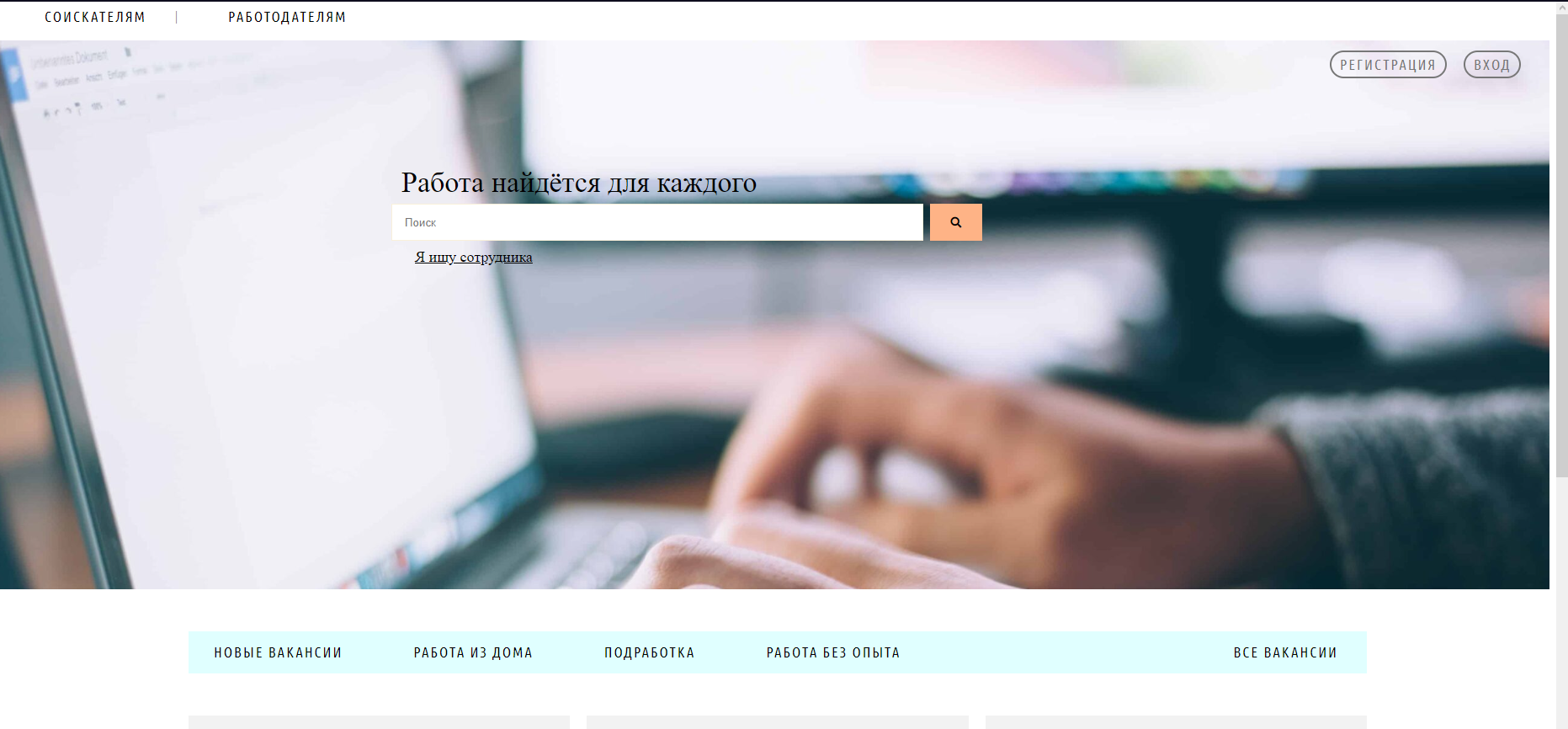


Рисунок 10 – Интерфейс на примере курсового сайта

**2.4 Разработка запросов**

Разработка запросов посвящена созданию и выполнению запросов на языке PHP для работы с базами данных. В PHP запросы используются для извлечения, вставки, обновления и удаления данных из базы данных.

Для работы с базами данных в PHP вы используете расширение MySQLi (MySQL Improved). MySQLi предоставляет удобные методы для выполнения запросов и обработки результатов. Данный код демонстрирует регистрацию пользователя и добавления данных о нем в базу данных MySQL с использованием расширения MySQLi.

<?php

header("Access-Control-Allow-Origin: http://localhost:5173"); // Разрешение доступа к этому скрипту только с указанного домена

header("Access-Control-Allow-Headers: Content-Type"); // Разрешение использования заголовка Content-Type

header("Content-Type: application/json"); // Установка типа контента на application/json

require\_once 'conect.db.php'; // Подключение файла с подключением к базе данных

if(isset($\_POST['registr'])){ // Проверка, была ли отправлена форма с именем "registr"

$organization=$\_POST['organization']; // Получение значения поля "organization" из POST-запроса

$number=$\_POST['number']; // Получение значения поля "number" из POST-запроса

$email=$\_POST['Email']; // Получение значения поля "Email" из POST-запроса

$password=$\_POST['password']; // Получение значения поля "password" из POST-запроса

$hashed\_password = hash('sha256', $password); // Хеширование пароля с использованием алгоритма SHA-256

// Подготовленный запрос для выборки пользователя с указанной электронной почтой

$stmt = $conn->prepare("SELECT \* FROM employer WHERE email = ?;");

$stmt->bind\_param("s", $email);

$stmt->execute();

$result = $stmt->get\_result(); // Получение результата запроса

if ($result->num\_rows === 0) { // Если пользователь с такой электронной почтой не существует

// Подготовленный запрос для добавления нового пользователя в таблицу "employer"

$stmt = $conn->prepare("INSERT INTO employer (organization, number, email, password, role) VALUES (?, ?, ?, ?, 'employer')");

$stmt->bind\_param("ssss", $organization, $number, $email, $hashed\_password);

if ($stmt->execute()) { // Если данные были успешно добавлены в базу данных

$role = 'employer'; // Задание значения переменной роли

$id\_app = $stmt->insert\_id; // Получение ID вставленной записи

echo json\_encode(["success" => true, "role" => $role, "userId" => $id\_app, "data" => "Авторизация успешна"]);

// Вывод JSON-ответа с данными об успешной авторизации

} else {

// Если данные не были добавлены, вернуть "error"

echo "error";

}

} else {

// Если пользователь с такой электронной почтой уже существует, вернуть "email\_exists"

echo "email\_exists";

}

$stmt->close(); // Закрытие подготовленного запроса

}

$conn->close(); // Закрытие соединения с базой данных

header("Access-Control-Allow-Origin: \*"); // Разрешение доступа к этому скрипту с любого домена

# 2.5 Разработка выходных документов

Организационно-технологическое решение задачи должно быть разработано таким образом, чтобы в одних и тех же документах исходной информации, представляемых на ЭВМ, и в документах выходной информации, выдаваемых с ЭВМ, отражалось многообразие особенностей производственных мощностей различных видов производств. Алгоритм решения задачи должен позволять.

Входных документов Выходных документов

Отраслевые модели успешно применяются для построения входных документов. Выходные документы после печати на.

Повышению эффективности работы с информацией способствуют специальные приемы динамического (ускоренного) чтения текстов. Метод динамического чтения позволяет воспринимать текст со скоростью более 300 слов в минуту при усвоении порядка 70 % материала. Советскими специалистами Л. В. Власовым и В. Н. Гладковым разработан специальный алгоритм ускоренного чтения организационно-распорядительных документов, используемых на предприятиях, в учреждениях и организациях нашей страны, включающий опорные блоки документа, на которые секретарь должен направить основное внимание при чтении вид документа, автор документа, выходные данные документа, заголовок к тексту, содержание текста, введение (мотивы и цели написания документа), доказательство необходимости предлагаемого действия, изложение существа дела (предлагаемое действие, исполнители, сроки исполнения)

Информацию, необходимую для решения задач в условиях АСУ, распределяют по 2 группам документов входным в ЭВМ и выходным. Все сказанное выше о требованиях к документам, обрабатываемым в АСУ, относится к входным документам. Выходные документы должны отражать итоги решения задач с помощью ЭВМ и давать результаты в виде, необходимом пользователю системы. Количество получаемых документов и их объем могут быть существенно уменьшены, если идет не полная информация, а лишь та, которая характеризует отклонения от плана или нормативов. При подготовке выходных форм нужно иметь в виду и должностное положение потребителя информации АСУ министру или, например, директору необходимы более укрупненные данные, чем начальнику главка или соответственно начальнику цеха.

Настраиваемость на формы выходных документов (выходных сообщений).

При большом объеме обрабатываемых секретарем документов может быть применен метод ускоренного чтения документов по блокам. Он основывается на том, что состав реквизитов и тексты различных документов в известной мере унифицированы. Так, главным составным элементом приказа, распоряжения является распорядительная часть письма, докладной записки - основная часть. Унификация документации позволяет усовершенствовать алгоритм восприятия, который в окончательном виде включил в себя следующие блоки автор документа выходные данные заголовок содержание введение (мотивы и цели написания документа) доказательство необходимости предполагаемого действия изложение существа дела, исполнители, сроки исполнения.

Выходные сведения располагаются на титульном листе, обороте титульного листа, последней странице документа (выходные и выпускные данные).

Принципиально иная схема направления потоков информации в условиях функционирования АСУ. Дело в том, что в условиях АСУ информация об управляемом объекте не собирается органом управления, а поступает непосредственно на ЭВМ, точнее в информационно-вычислительный центр (ИВЦ). Там она собирается, систематизируется и используется для решения отдельных задач. Результаты решения (выходные документы) передаются управляемому объекту.

Информационное обеспечение автоматизированной системы управления технологическими процессами на АЗС - совокупность исходных данных, закодированных соответствующим образом, и форм входных и выходных документов.

Первичные документы - это документы, содержащие исходную запись сведений, полученных в процессе исследований, разработок, наблюдений, анализа или других видов человеческой деятельности независимо от ее характера или тематики. Деление документов на первичные и вторичные в значительной степени условно, поскольку один и тот же документ может содержать сведения, относящиеся как к первой, так и второй группе. Так, считающиеся безусловно первичными видами документов - отчеты по НИР, проектные документы, монографии и т.д., могут содержать наряду с оригинальными сведения или данные ранее включенные в другие первичные или даже вторичные документы.

Вторичные документы - это документы (отчеты), полученные в результате аналитико-синтетической и логической переработки сведений или данных, содержащихся в первичных документах. Примерами вторичных документов являются справочные и энциклопедические издания, рефераты и реферативные издания, библиографические издания, указатели и списки, обзоры (за исключением т.н. аналитических обзоров) и т.п.

В процессе обработки этих документов в компании создаются такие выходные отчеты как: заказы на поставку, получение денежных средств на расчетном счете, получение денежных средств в кассу, готовый проект для монтажа или строительства объектов, отчет о разгрузке материалов на склад или строительный объект, отчет о изменение работы филиала в соответствии законов Российской Федерации, отчет о согласование проекта на монтаж или строительство, прайс-лист компании на продажу продукции, отчеты о изменение деятельности в соответствие корпоративной стратегии на рынке и т.д.

Входные документы: заказ, квитанция о поступлении товара, квитанция о поступлении денежных средств.

К выходным документам относится: отчет о поступлении товара, отчет о продажах, о поступлении денежных средств.

Входные и выходные документы должны, как правило, выполняться на типовых, специализированных и стандартных бланках, что позволяет использовать для их заполнения.

Реквизиты входных и выходных документов наиболее полно удовлетворяют систему расчетов по заработной плате на предприятиях и могут быть использованы без изменений.

При проектировании форм входных и выходных документов возникают две проблемы. Первая связана с проектированием содержания документов, включающего состав реквизитов и показателей, которые будут в него входить.

Используемые в системе формы входных и выходных документов должны быть не только полными, позволяющими охватывать все аспекты задачи, но и экономными, исключающими непроизводительные затраты ручного труда и машинного времени.

Сравнительно легко оценить число входных и выходных документов, которые используются при обработке известного объема данных.

Полный состав и структура записи входных и выходных документов по всему комплексу решаемых задач и видам изданий должен быть определен в ходе технического и рабочего проектирования.

Были разработаны и внедрены формы входных и выходных документов, инструкции и графики работы, позволяющие связать технические средства и людей, удаленных на расстояния до 1000 км в единую систему человек - машина, обеспечивающую сбор и обработку информации. Для приема телемеханической информации было создано устройство согласования. Для воспроизведения информации разработаны специальный телеэкран и дистанционные пульты запроса руководства.

Выходным документом данного веб–приложения является резюме соискателя, который скачивает работодатель после одобрения отклика на выборанную соискателем вакансию. Резюме представляет из себя pdf файл, который можно скачать. Содержимое документа представляет из себя информацию о соискателе, а так же его контактных данных.(рисунок 11)

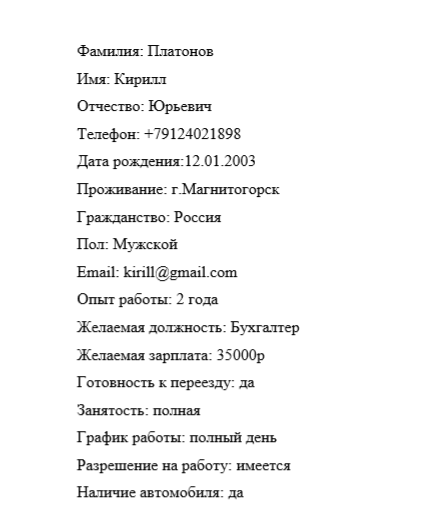


Рисунок 11 – Резюме соискателя

# 2.6 Разграничение прав доступа

Права доступа позволяют участникам иметь доступ к определенным функциям сайта курса или сайта проекта, в зависимости от их ролей и того, какие функции назначил им владелец или администратор сайта.

Роли - это наборы прав доступа. Некоторые роли дают участнику простой доступ к чтению материала, в то время как другие позволяют производить изменения в работе сайта и инструментов, например, добавление участников, изменение материала или прав доступа для других ролей. Все роли не эквивалентны.

При создании сайта, или при получении готового, вы получаете роль с наибольшими правами доступа и самый широкий уровень допуска. После этого вы можете выбрать (в рамках дозволенного администратором), какие инструменты или функции (например, Форум, Календарь, Ресурсы) должны присутствовать на сайте. Для каждого из этих инструментов можно настроить права доступа, чтобы не допустить совершения участниками действий, которые выходят за рамки их ролей.

Обзор ролей участников и установление собственной роли смотри более подробно в Роли участников.

Основные возможности

Роли могут давать различные возможности для работы с разными инструментами. Тем не менее, в большинстве случаев, создатель сайта проекта или преподаватель на сайте курса имеет все права доступа и может добавлять или удалять материал с сайта.

Пользователи, которые добавлены на уже существующий сайт, имеют более ограниченные в правах доступа роли. Права доступа этих ролей позволяют взаимодействие с сайтом, например, отправку сообщений в чате, но не позволяют совершать некоторых других действий, например, загружать файлы с помощью инструмента "Ресурсы" или заполнять Расписание.

Тем не менее, настройки для каждого инструмента различны. Администратор сайта определяет, какие роли установить и какие права доступа им предоставить. Роли и связанные с ними права доступа будут различны на разных сайтах.

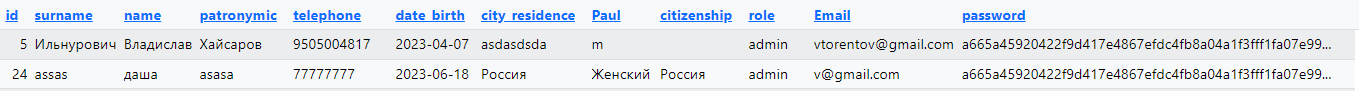
 Таблица пользователей и их ролей представлена ниже (Рис. 12)

Рисунок 12 – Таблица пользователей

Форма авторизации представлена на рисунке 13

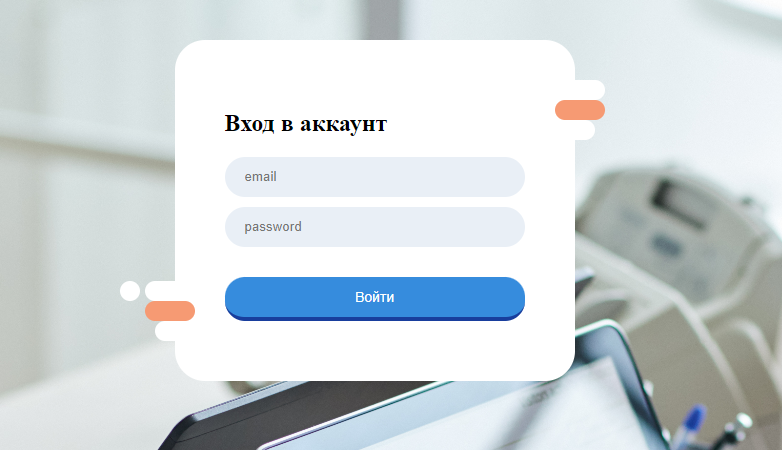


Рисунок 13 – Форма авторизации

Так же на сайте присутствует форма регистрации, которую можно наблюдать на рисунке 14.

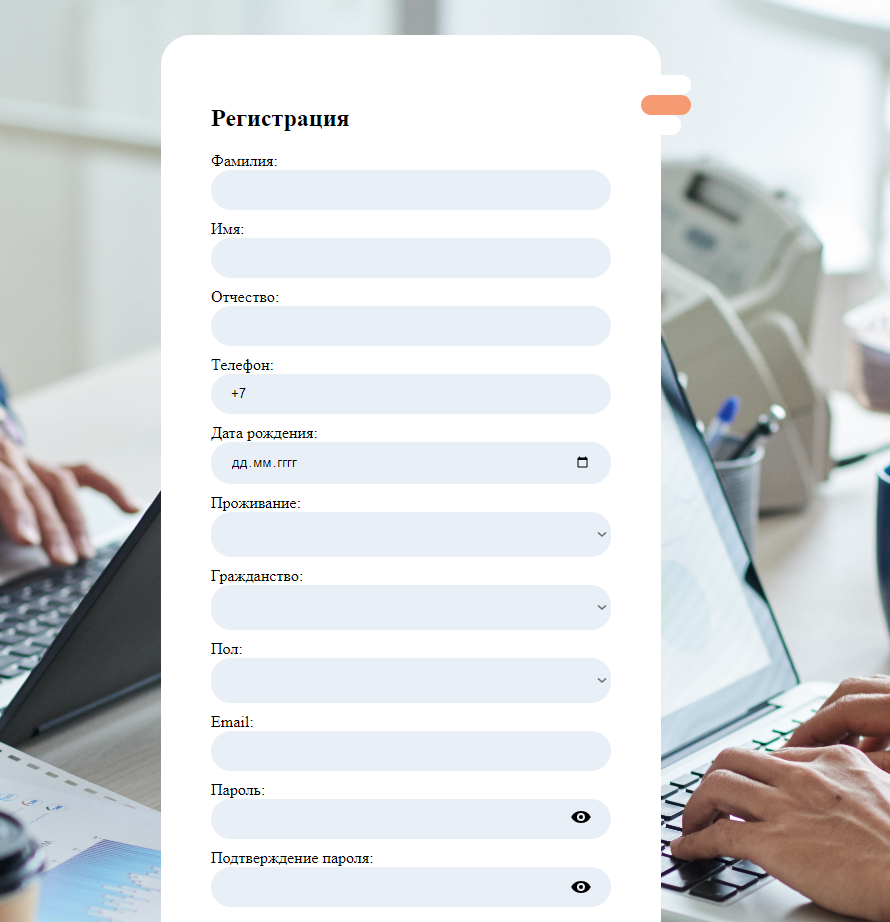


Рисунок 14 – Форма регистрации

# 2.6 Руководство пользователя

Инструкция по использованию программы (или просто «Инструкция пользователю», или «Руководство для пользователя») - это выдержка из полной документации, предназначенная для эксплуатации программы. Она представляет собой независимый документ для пользователя программы, в котором описывается: что делает программа и как им пользоваться.

«Инструкция пользователю» должна содержать всю необходимую для пользователя информацию и должна быть ему понятна без дополнительных материалов (без обращения к другим спецификациям). Следовательно, необходимая для этой инструкции информация переписывается полностью из соответствующих спецификаций.

Первая часть инструкции является описательной и должна содержать:

* наименование программы;
* краткое описание программы;
* перечень выполняемых программой функций;
* краткую характеристику метода (или методов) решения поставленной задачи, его досто­инство и недостатки;
* полную библиографическую ссылку на полное описание метода;
* описание входных и выходных данных.
* описание структуры базы данных (если она имеется), всех ее таблиц в словесной (вербальном) форме.

Вторая часть документа должна описывать порядок работы с программой. Она должна содержать описание всех режимов работы программы, а также содержание всех печатей и диагностических сообщений, которые выдаются по ходу выполнения программы.

Следует помнить, что пользователь по своей квалификации не является программистом и поэтому его работа с программой описывается на понятном ему языке и достаточно подробно, а именно:

* как запустить программу;
* как продолжить работу с программой (описывается подробный интерактивный режим работы пользователя с программой);
* подготовка и ввод исходных данных в программу;
* как реагировать на запросы программы;
* как вести работу в исключительных ситуациях;
* как реагировать на ошибки;
* как восстановить работу программы в случае аварийного его завершения;
* как получить требуемый результат;
* как правильно закончить работу с программой (запланированный программой выход);
* другие сведения, необходимые пользователю программы.

Руководство для обычного пользователя

При заходе на сайт у обычного пользователя имеются ряд ограничений. Например, незарегистрированный пользователь не может подать заявку на поступление в детский сад, а также оставлять отзывы на сайте. Он не может зайти в свой личный кабинет, так как его попросту нет. Для того чтобы зарегистрироваться или авторизироваться нужно пожать в меню навигации на соответствующую кнопку «Вход». Это можно увидеть на рисунке 15.



Рисунок 15 − Кнопка для входа в систему

После того, как пользователь зарегистрировался, он может создать резюме (рисунок 16) или вакансию (рисунок 17) смотря от своей роли.

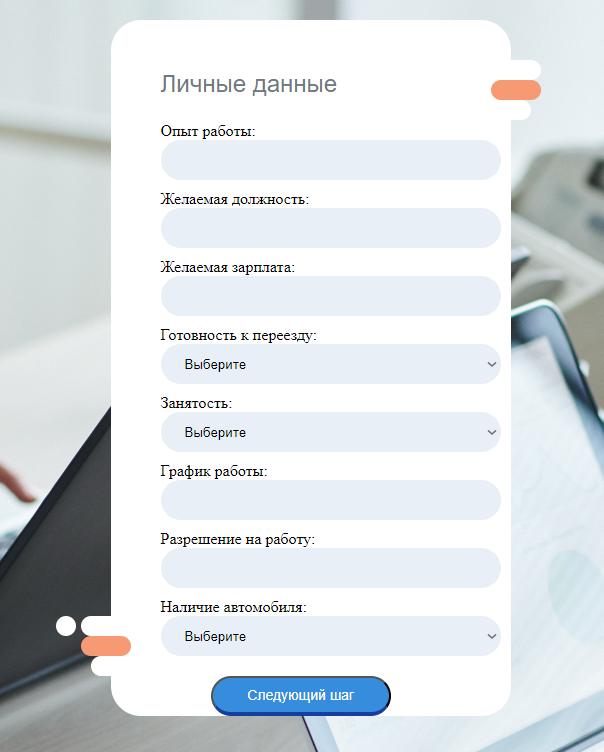


Рисунок 16 – Создание резюме

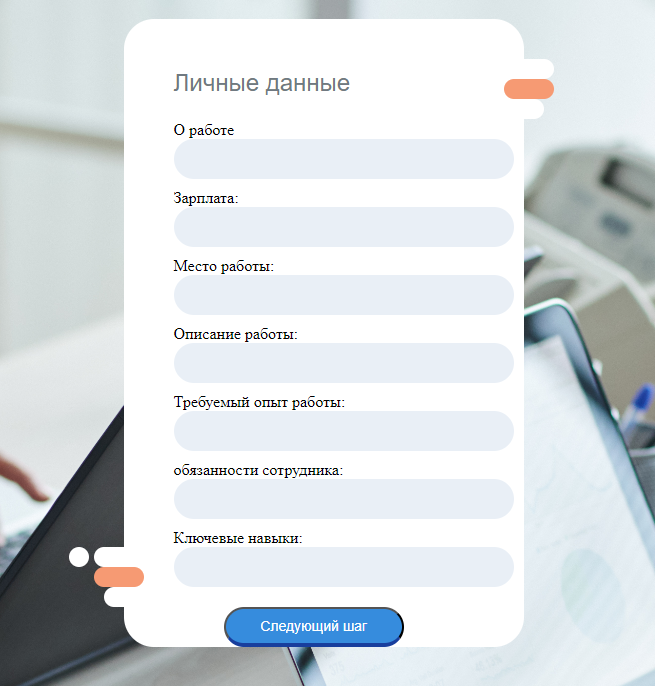


Рисунок 17 – Создание вакансии

Далее он может зайти в личный кабинет (рисунок 18) в нем может посмотреть свои личные данные (рисунок 19) а также изменить их при необходимости, свое резюме либо вакансию может тоже рассмотреть (рисунок 20).

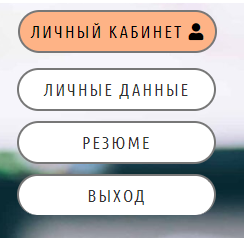


Рисунок 18 – Личный кабинет

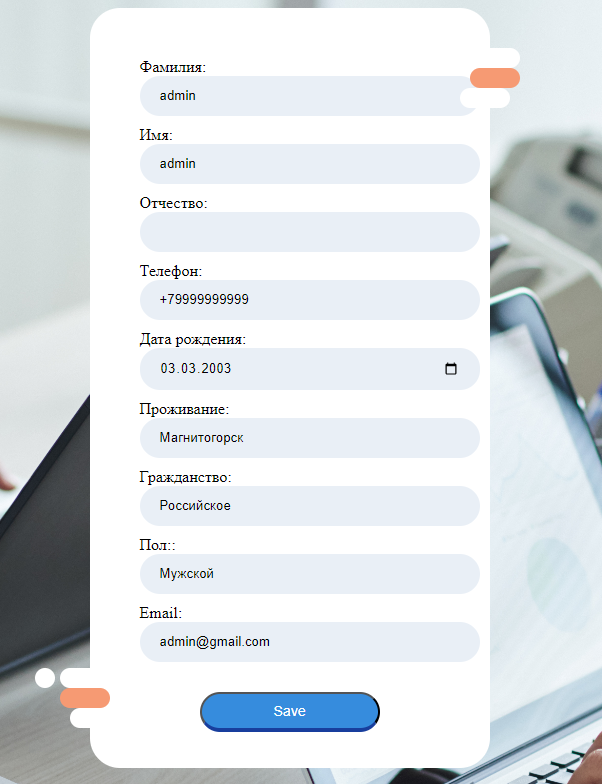


Рисунок 19 – Личные данные

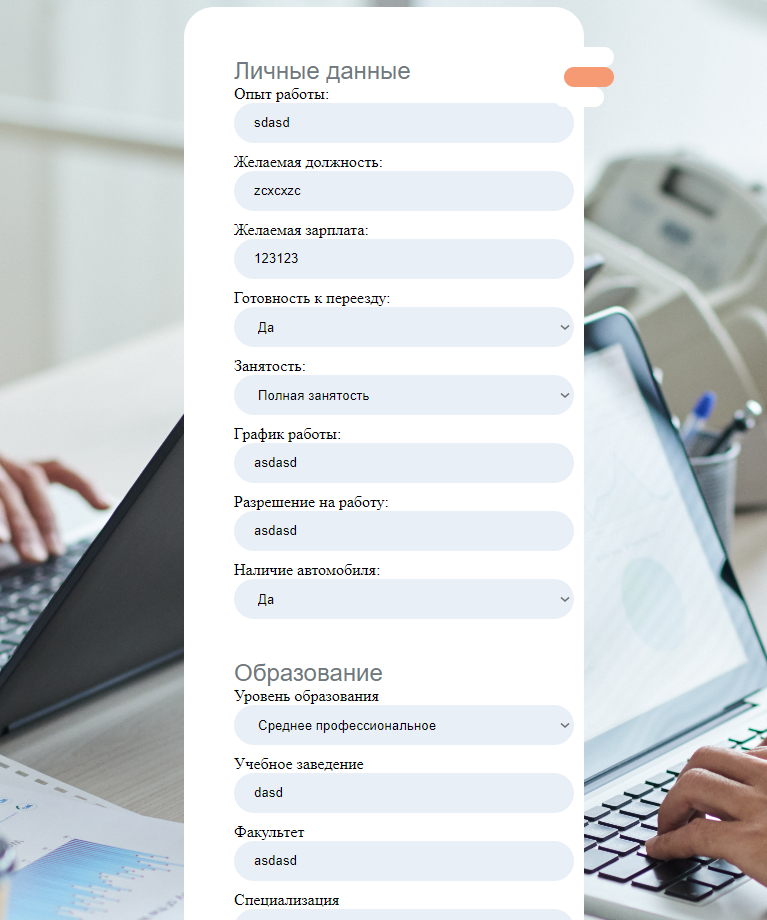


Рисунок 20 – Изменение резюме

При переходе на интересующую вакансию выходит подробная информация о ней (рисунок 21)

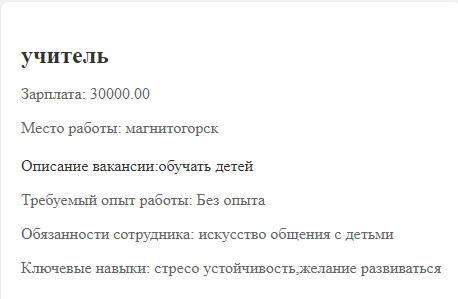


Рисунок 21 – Информация о вакансии

# 2.7 Тестирование информационной системы

Тестирование - очень важный и трудоемкий этап процесса разработки программного обеспечения, так как он позволяет выявить подавляющее большинство ошибок, допущенных при составлении программ.

Процесс разработки программного обеспечения предполагает три стадии тестирования:

* автономное тестирование компонентов программного обеспечения
* комплексное тестирование разрабатываемого программного обеспечения
* системное или оценочное тестирование на соответствие основным критериям качества.

Для повышения качества тестирования рекомендуется соблюдать следующие основные принципы:

* предполагаемые результаты должны быть известны до тестирования;
* следует избегать тестирования программы автором;
* досконально изучать результаты каждого теста;
* необходимо проверять работу программы на неверных данных;
* вероятность наличия необнаруженных ошибок в части программы пропорциональна числу ошибок, уже обнаруженных в этой части.

Формирование набора тестов имеет большое значение, поскольку тестирование является одним из наиболее трудоемких этапов (от 30 до 60 % общей трудоемкости) создания программного продукта. Существуют два принципиально разных подхода к формированию тестовых наборов – структурный и функциональный.

Структурный подход базируется на том, что известны алгоритмы работы программы. В основе структурного тестирования лежит концепция максимально полного тестирования всех маршрутов программы. Для формирования тестов программу представляют в виде графа, вершины которого соответствуют операторам программы, а дуги представляют возможные варианты передачи управления.

При структурном тестировании данной части программы необходимо сформировать как минимум пять тестовых наборов таким образом, чтобы обойти весь граф.

Функциональный подход основывается на том, что алгоритм работы программного обеспечения не известен. Тесты строят, опираясь на функциональные спецификации. Программа рассматривается как «черный ящик», и целью тестирования является выяснение обстоятельств, в которых поведение программы не соответствует требованиям.

Для тестирования модулей программного обеспечения возможно применение восходящего и нисходящего тестирования.

Восходящий подход предполагает, что каждый модуль тестируют отдельно, затем собирают протестированные модули в более крупные модули программы и тестируют их. Так продолжает пока не будет собран весь программный продукт. Данный подход имеет существенные недостатки. Во-первых, при восходящем тестировании серьезные ошибки в функциональности, алгоритмах и интерфейсе могут быть обнаружены только на завершающей стадии работы над проектом. Во-вторых, для того, чтобы тестировать модули нижних уровней, необходимо разработать специальные тестирующие программы, которые обеспечивают вызов интересующих нас модулей с необходимыми параметрами.

При нисходящем тестировании проверка начинается с главного модуля. К нему постепенно подключаются модули более низких иерархических уровней. Если модули нижних уровней не работоспособны, то вместо них подключаются программы-заглушки. Таким образом, тестируется некоторая модель группы программ с имитаторами-заглушками. Основной недостаток нисходящего тестирования – необходимость создания программ-заглушек. Основным достоинством данного метода является возможность на ранних стадиях разработки получения рабочую версии программного продукта.

На практике чаще всего комбинируют восходящий и нисходящий подходы к тестированию. Модули верхних уровней тестируют нисходящим способом, а модули нижних уровней - восходящим. Такой подход позволяет с одной стороны начать с тестирования интерфейса, с другой - обеспечивает качественное автономное тестирование модулей низших уровней.

На этапе тестирования решается задача верификации соответствия программной реализации требованиям, сформированным на первых двух этапах жизненного цикла ИС. Полную проверку программного продукта гарантирует исчерпывающее тестирование, предполагающее обязательное включение в тесты всех наборов исходных данных, всех вариантов их обработки и большое количество тестовых вариантов. Такая проверка ИС предполагает большое количество тестеров и может занять много времени.

Опытные отладчики обнаруживают ошибки путём сравнения шаблонов тестовых выходных данных с выходными данными тестируемых систем. Чтобы определить местоположение ошибки, необходимы знания о типах ошибок, шаблонах выходных данных, языке и процессе программирования. Очень важны знания о процессе разработке программного обеспечения.

Среди методов тестирования информационных систем, коих существует множество, выделим следующие, как наиболее часто используемые:

* Модульное тестирование (Unit test);
* Анализ покрытия кода тестами (Test coverage);
* Анализ стилистики кода
* Анализ производительности (Benchmark, Profiling).

Анализ стилистики, производительности, интеграционное и прочие виды тестирования здесь рассматриваться не будут.

Как в ходе выполнения проекта, так и при развитии системы, изменения в ее алгоритмы вносятся в соответствии с техническими заданиями, хотя данный документ может иметь различные названия. В технических заданиях, как правило, описываются критерии приемки: как мы и заказчик, убедимся в том, что заказанный функционал работает верно.

Автоматизированное тестирование лишено данных недостатков. Конечно, возникает отдельная задача кодирования данного теста на основании критериев приемки, но это скорее дисциплинирующий фактор.

Инструменты автоматизированного тестирования

В зависимости от платформы, применяются различные инструменты тестирования. Рассмотрим те инструменты, которые можно применить на платформе 1С:

Для Unit-тестирования можно применять систему «1С: Сценарное тестирование 8», хотя сценарное тестирование и отличается от блочного.

Для проверки соблюдения стандартов 1С существует инструмент «Автоматизированная проверка конфигураций».

Для анализа производительности можно применять хорошо себя зарекомендовавший «Корпоративный инструментальный пакет 8».

К сожалению, инструмента анализа покрытия кода 1С тестами пока не встречал.

Приведенные инструменты пусть и не являются идеальными, тем более если речь идет о различных этапах жизненного цикла систем, но позволяют с чего-то начать, автоматизировать наиболее трудоемкие задачи рутинного характера.

К плюсам, относятся:

Повышение качества разрабатываемого функционала.

Снижение трудозатрат на повторное тестирование. Тем более, что тестировать полезно каждую сборку целевой системы.

Повышение привлекательности компании для заказчиков. Применение технологий автоматизированного тестирования является показателем высокой культуры разработки и практически мировой тенденцией.

Также разработанные тесты могут быть монетизированы как продукт интеллектуальной деятельности. Например, по завершению проекта тесты могут быть переданы заказчику если заказчик планирует развивать систему силами собственного ИТ-подразделения.

Таблица 15 – Тестирование методом «черного ящика»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Номер | Назначение теста | Ожидаемый результат | Фактический результат | Итог |
| 1 | Нажать кнопку «Войти» на главной странице | Откроется окно входа в систему | Открылось окно входа | Выполнено |
| 2 | Войти в личный кабинет, нажав на вкладку | Раскроется список | Раскроется список | Выполнено |
| 3 | Нажать кнопку «Выход» | Выйдет из системы и перейдет на главную страницу | Страница обновилась, вместо кнопки выход появилась кнопка «Вход». Кнопка личного кабинета пропала. | Выполнено |
| 4 | Нажать кнопку «Регистрация» во вкладке «Авторизация» | Появятся формы для регистрации | Появились формы для регистрации | Выполнено |
| 5 | Нажать «Регистрация» после заполнения данных в формы регистрации | Пользователь добавится в базу данных и пользователь попадет в личный кабинет | Новый пользователь добавился в базу данных. Пользователь попал на другую страницу | Выполнено |
| 6 | Изменение личных данных | Редактирование личных данных | Данные изменились,  Обновилась страница | Выполнено |
| 7 | Изменение вакансии | Редактирование вакансии | Данные изменились,  Обновилась страница | Выполнено |
| 8 | Изменение резюме | Редактирование резюме | Данные изменились,  Обновилась страница | Выполнено |
| 9 | Поиск вакансий | Вставляем текст и ищем | Нашел вакансию | Выполнено |
| 10 | Отклик на вакансию | Нажимаем на кнопку отклик | Отклик успешно отправлен | Выполнено |

# ЗАКЛЮЧЕНИЕ

В результате была разработана база данных «WORK», в которой использовались такие таблицы как «applicant», «education», «employer», «[language-resume](http://127.0.0.1/openserver/phpmyadmin/index.php?route=/sql&db=WORK&table=language-resume&pos=0)», «[language-vacancies](http://127.0.0.1/openserver/phpmyadmin/index.php?route=/sql&db=WORK&table=language-vacancies&pos=0)», «[languages](http://127.0.0.1/openserver/phpmyadmin/index.php?route=/sql&db=WORK&table=languages&pos=0)», «[resume](http://127.0.0.1/openserver/phpmyadmin/index.php?route=/sql&db=WORK&table=resume&pos=0)», «[resume\_vacancy](http://127.0.0.1/openserver/phpmyadmin/index.php?route=/sql&db=WORK&table=resume_vacancy&pos=0)», «[right](http://127.0.0.1/openserver/phpmyadmin/index.php?route=/sql&db=WORK&table=right&pos=0)», «[right-resume](http://127.0.0.1/openserver/phpmyadmin/index.php?route=/sql&db=WORK&table=right-resume&pos=0)», «[specialization](http://127.0.0.1/openserver/phpmyadmin/index.php?route=/sql&db=WORK&table=specialization&pos=0)», «[vacancy](http://127.0.0.1/openserver/phpmyadmin/index.php?route=/sql&db=WORK&table=vacancy&pos=0)», и «[work\_schedule](http://127.0.0.1/openserver/phpmyadmin/index.php?route=/sql&db=WORK&table=work_schedule&pos=0)». Каждая из таблиц была заполнена информацией и внедрена в сайт при помощи PHP vue.js. База данных может быть отредактирована на самом сайте по средствам PHP vue.js.

Также было произведено моделирование пяти диаграмм. Были смоделированы функциональная и концептуальная модели, и диаграмма вариантов использования, состояний, деятельности, последовательности. Все они представлены в виде информации, описывающей существенные для данного рассмотрения параметры и переменные величины объекта.

Разработан интерфейс приложения, который будет понятен каждому пользователю. В котором все интуитивно понятно и все цвета подобраны идеально и соответствуют стандартным нормам качества веб-сайта.

В приложении были реализованы админская и клиентская части. Между этими частями есть значительные отличия. У администратора есть права на редактирование, удаление и просмотр информации, которая недоступна обычному клиенту. Клиент незарегистрированный на сайте также имеет ряд действий, которые недоступны до тех пор, пока не произойдет авторизация. Например, просмотр личного кабинета, создание резюме.

Также приложение было протестировано при помощи метода «черного ящика». Тестирование не вывело каких-то явных ошибок, которые могут как-то повлиять на работу приложения.

В результате выполнения курсового проекта были выполнены все задачи.

# СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

1. Федорова,Г.Н. Разработка, внедрение и адаптация программного обеспечения отраслевой направленности [Электронный ресурс]: учеб. Пособие/ Г.Н. Федорова— М. :КУРС : ИНФРА-М, 2018. — 336 с. (Среднее Профессиональное Образование). -Режим доступа: <http://new.znanium.com/bookread2.php?book=898670>
2. Гагарина, Л. Г. Разработка и эксплуатация автоматизированных информационных систем [Электронный ресурс] : учебное пособие / Л. Г. Гагарина. — Москва : ИД «ФОРУМ» : ИНФРА-М, 2019. — 384 с. — (Среднее профессиональное образование). - Режим доступа: <https://new.znanium.com/read?id=333679>
3. Исаев, Г. Н. Управление качеством информационных систем [Электронный ресурс] : учебное пособие / Исаев Г. Н. - Москва :НИЦ ИНФРА-М, 2016. - 248 с.: 60x90 1/16. - (Высшее образование: Бакалавриат) (Переплёт 7БЦ) ISBN 978-5-16-011794-2 - Режим доступа: <https://new.znanium.com/read?id=36233>
4. Коваленко, В. В. Проектирование информационных систем [Электронный ресурс] : учебное пособие / В. В. Коваленко. — Москва : ФОРУМ : ИНФРА-М, 2018. — 320 с. — Режим доступа: <https://new.znanium.com/read?id=329691>
5. Золотухина, Е. Б. Управление жизненным циклом информационных систем (продвинутый курс) [Электронный ресурс]: конспект лекций / Е. Б. Золотухина, С. А. Красникова, А. С. Вишня. - Москва :КУРС, НИЦ ИНФРА-М, 2017. - 119 с.: ISBN 978-5-906818-36-2 - Режим доступа: <https://new.znanium.com/read?id=145707>
6. Сысоева, Л. А. Управление проектами информационных систем [Электронный ресурс] : учебное пособие / Л. А. Сысоева, А. Е. Сатунина. — Москва : ИНФРА-М, 2019. — 345 с. — (Высшее образование: Бакалавриат). — Режим доступа: <https://new.znanium.com/read?id=342011>
7. Портал по php, MySQL и другим веб-технологиям [электронный ресурс] : учебное пособие / [Дж. Уайттеккер, Дж. Воас](http://ict.edu.ru/catalog/index.php?a=nav&c=getForm&r=navDesc&id_res=930&d=light) // Режим доступа : <http://www.php.su>
8. Ресурсы по программированию на php: [Электронный ресурс] // URL: <http://www.php.net>
9. Основы языка гипертекстовой разметки HTML и CSS [электронный ресурс] : учебное пособие /Сост. С. М. Наместников. – Ульяновск: УлГТУ, 2014. // Режим доступа: <http://sernam.ru/book_html.php>
10. Ресурсы по программированию: [Электронный ресурс] // URL: <http://www.realcoding.net/>

ПРИЛОЖЕНИЕ А

Листинг 1 – Вывод вакансий из базы данных.

<?php

// Устанавливаем заголовки для разрешения CORS (Cross-Origin Resource Sharing),

// позволяющие запросам из других доменов получать доступ к этому скрипту

header('Access-Control-Allow-Origin: \*'); // Здесь \* означает, что доступ разрешен с любого домена

header('Content-Type: application/json'); // Устанавливаем тип контента как JSON

require\_once 'conect.db.php'; // Подключаем файл с настройками подключения к базе данных

$query = "SELECT \* FROM vacancy"; // Формируем SQL-запрос для выборки всех записей из таблицы vacancy

$result = mysqli\_query($conn, $query) or die(mysqli\_error($conn)); // Выполняем запрос к базе данных

// Формируем массив с данными вакансий

$data = array(); // Создаем пустой массив, в который будут добавляться данные вакансий

while ($row = mysqli\_fetch\_assoc($result)) {

$data[] = $row; // Добавляем каждую строку результата выборки в массив $data

}

// Отправляем данные в виде JSON

echo json\_encode($data); // Преобразуем массив данных в формат JSON и отправляем клиенту

?>

<script>

import axios from 'axios';

export default {

data() {

return {

vacancies: [], // Массив для хранения данных о вакансиях

}

},

mounted() {

// Получение данных из сервера

axios.get('http://localhost/this\_is\_the\_base/vacancy.php') // Отправка GET-запроса к серверу

.then(response => {

this.vacancies = response.data; // Присваивание полученных данных о вакансиях свойству vacancies

})

.catch(error => {

console.log(error); // Обработка ошибок в случае неудачного запроса

});

},

};

</script>  
<template>  
<div v-for="vacancy in vacancies" :key="vacancy.id" class="block">

<!-- Итерация по каждой вакансии в массиве vacancies -->

<h2>{{ vacancy.title }}</h2> <!-- Вывод заголовка вакансии -->

<p>{{ vacancy.salary }} р</p> <!-- Вывод зарплаты вакансии с символом "р" -->

<p>{{ vacancy.work\_experience }}</p> <!-- Вывод опыта работы для данной вакансии -->

</div>

</template>