«Санкт-Петербургский государственный технологический институт  
(технический университет)»  
СПбГТИ (ТУ)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| УГНС | | 09.00.00 | Информатика и вычислительная техника | | |
| Направление подготовки | | 09.03.01 | Вычислительные системы, сети и телекоммуникации | | |
| Направленность (профиль) | |  | Системы автоматизированного проектирования | | |
| Форма обучения | |  | очная | | |
|  | |  |  | | |
| Факультет | |  | Информационных технологий и управления | | |
| Кафедра | |  | Систем автоматизированного проектирования и управления | | |
| Учебная дисциплина | |  | **ИНТЕРНЕТ-ТЕХНОЛОГИИ** | | |
| Курс | III | | | Группа | 413 |

КУРСОВОЙ ПРОЕКТ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Тема: | Разработка сайта для анализа вакансий дизайнера, полученных с сайта HeadHunter.ru | | | | |
| Студент | |  |  |  | Баймухаметова А.Г. |
| Руководитель,  доцент, к.т.н. | |  |  |  | Разыграев А.С. |

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 3](#_Toc151581157)

[**1** **АНАЛИТИЧЕСКИЙ ОБЗОР** 5](#_Toc151581158)

[**1.1** **Обзор клиент-серверных приложений** 5](#_Toc151581159)

[**1.2** **Обзор протоколов транспортного уровня модели TCP/IP** 8](#_Toc151581162)

[**1.3** **Обзор и обоснование выбора инструментальных средств разработки программного комплекса** 9](#_Toc151581163)

[**2** **ОСНОВНАЯ ЧАСТЬ** 14](#_Toc151581164)

[**2.1** **Определение структурной (иерархической) схемы решения задачи и требования к проектируемой программе** 14](#_Toc151581165)

[**2.2**  **Разработка структуры данных, определение диапазона изменений** 14](#_Toc151581166)

[**2.3**  **Анализ ограничений и исключительных ситуаций для алгоритмов** 16](#_Toc151581167)

[**2.4**  **Разработка алгоритмов Банкира** 16](#_Toc151581168)

[**2.5**  **Разработка архитектуры программы** 20](#_Toc151581169)

[**2.6**  **Разработка дисплейных фрагментов** 21](#_Toc151581170)

[**2.7**  **Написание и отладка модулей программы** 22](#_Toc151581171)

[**2.8**  **Тестирование разработанного программного продукта** 24](#_Toc151581172)

[ВЫВОДЫ ПО РАБОТЕ 24](#_Toc151581173)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 25](#_Toc151581174)

[ПРИЛОЖЕНИЕ А 26](#_Toc151581175)

# **ВВЕДЕНИЕ**

Интернет-технологии — это совокупность различных инструментов и методов, которые позволяют создавать, хранить, обрабатывать и передавать информацию по сети Интернет. Эти технологии включают в себя различные протоколы передачи данных, веб-серверы, базы данных, программное обеспечение для работы с сетью, а также различные способы взаимодействия пользователей с информацией.

Интернет-технологии играют ключевую роль в современном обществе, так как они позволяют обмениваться информацией, связывать пользователей со всего мира, создавать новые сервисы и продукты. Благодаря интернет-технологиям возможно проведение онлайн-конференций, обучения, покупки, банковские операции, а также развлечения и многое другое. Эти технологии постоянно развиваются, что способствует увеличению их эффективности, удобства использования и возможностей [1].

В современном мире профессия дизайнера становится все более востребованной, и соответственно появляется множество вакансий на рынке труда. Поиск подходящей работы для дизайнера может быть длительным и трудоемким процессом. В связи с этим, разработка сайта для анализа вакансий дизайнера может значительно упростить поиск работы и помочь специалистам найти подходящую вакансию.

Данный курсовой проект направлен на создание удобного и эффективного сайта, который будет автоматически собирать и анализировать вакансии для дизайнеров с популярного сайта HeadHunter.ru. Сайт будет предоставлять пользователям возможность удобного поиска вакансий, сортировки по различным параметрам и анализа предложений от работодателей. Таким образом, дизайнеры смогут экономить время и ресурсы, необходимые для поиска подходящей работы.

Для реализации проекта будет использовано программное обеспечение для веб-разработки, базы данных и аналитические инструменты. Проект также будет ориентирован на улучшение пользовательского опыта, уделяя внимание дизайну и удобству использования сайта. В результате, разработанный сайт поможет дизайнерам найти идеальную вакансию, способствуя успешному трудоустройству и развитию профессиональной сферы.

# **АНАЛИТИЧЕСКИЙ ОБЗОР**

## 1.1 Описание автоматического парсера вакансий. Формирование функциональных требований к информационной системе

Автоматический парсер вакансий — это программа или инструмент, который используется для автоматического сбора информации о доступных вакансиях с различных интернет-ресурсов, таких как сайты компаний, специализированные порталы или профессиональные сети. Парсер анализирует текст вакансий, извлекает ключевую информацию, такую как название вакансии, требуемые навыки, обязанности, местоположение и другие важные данные, и сохраняет их в удобном для дальнейшего использования формате, например, в базе данных или таблице [1].

Такой инструмент часто используется рекрутерами, HR-специалистами и кадровыми агентствами для автоматического сбора и обработки информации о вакансиях, что позволяет сэкономить время и усилия при поиске подходящих кандидатов. Также автоматический парсер может быть полезен соискателям, позволяя быстро находить вакансии и отправлять свои резюме.

Некоторые парсеры вакансий также могут предоставлять аналитику и отчеты, а также возможность настройки критериев поиска и фильтров, чтобы получать наиболее релевантные результаты.

Цель проекта: анализ опубликованных вакансий с сайта HeadHunter.ru для поиска работы студенту дизайнеру.

Требования к разрабатываемому веб-сайту:

1. Таблица со списком вакансий;
2. Поиск по столбцам таблицы;
3. Диаграмма с городами, где наибольшее количество вакансий;
4. Диаграмма по диапазонам зарплат.

## 1.2 Обзор и обоснование выбора технологий вёрстки макетов

EJS

Движок шаблонов EJS или EJS - популярный движок шаблонов для веб-разработки. он используется в Nodejs. Он позволяет генерировать динамический контент путем встраивания JavaScript в HTML-контент. EJS служит языком шаблонов, использующим преобразование javascript в HTML. Он позволяет вводить данные в шаблон на стороне клиента для генерации окончательного HTML-вывода.

EJS обладает рядом преимуществ, которые сделали его популярный выбор:

* Простота принятия: EJS легко интегрируется с приложениями Node.js, что делает его консолью для пользователей, уже использующих JavaScript на стороне сервера.
* Знакомый синтаксис: Те, кто хорошо разбирается в JavaScript, делают EJS простым для понимания, поскольку он позволяет использовать JavaScript непосредственно в своих HTML-шаблонах.
* Надежность: EJS предоставляет мощные функции, такие как циклы, условные выражения и частичные выражения, позволяющие легко реализовать сложные структуры шаблонов.

EJS работает путем анализа HTML-файлов со встроенным кодом JavaScript и выполнения этого кода для создания динамического контента. Когда клиент запрашивает веб-страницу, сервер обрабатывает шаблоны EJS, заменяет встроенный JavaScript фактическими данными и отправляет полученную HTML-разметку в браузер клиента для рендеринга.

Таблицы

Создание каркаса с помощью таблиц. В прошлом самый основной и популярный способ. Сегодня устаревает с каждым годом, хотя остаются сайты, которые сделаны так. Весь метод завязан на теге table и его дочерних элементах, таких как строки и ячейки. Вся структура сайта размечается как большая таблица [2].

Достоинство:

* Такой подход позволяет достаточно легко создать много колонок и столбцов;
* Макеты со сложной структурой достаточно просто верстаются с помощью таблицы;
* Тег table и все связанные с ним элементы поддерживаются абсолютно одинаково во всех браузерах.

Недостатки:

* Очень громоздкий код – главный минус табличной верстки;
* Нужно отметить, что в одну таблицу можно вставить другую, то есть вложенную. Это приводит к большой вложенности и куче тегов. И никуда от этого не деться, потому что все эти теги нужны для работоспособности нашей таблицы;
* Табличные данные очень сложно адаптировать для просмотра на различных устройствах. Намного проще сделать это с блоками.

Альтернатива. Не так давно в CSS появились специальные значения свойства display, которые позволяют имитировать таблицу, используя на самом деле блоки. Например, display: table, display: table-row, display: table-cell. Элементы, к которым применяться эти свойства, начнут вести себя как таблица, ряд таблицы и ячейка соответственно. Нужно скачать, что и этот метод несовершенен, потому что не совсем одинаково поддерживается во всех браузерах. Если вы используете эти свойства, нужно тщательно тестировать на кроссбраузерность.

Пример использования Таблицы на рисунках 1 и 2:

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 1 – HTML-код для демонстрации таблицы

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

Рисунок 2 – Результат отображения HTML кода с использованием таблицы

Блоки

Данная технология имеет ряд преимуществ и в целом признана самой удобной. Каждый блок – это независимый элемент, в который можно вкладывать неограниченное количество элементов. Блок можно позиционировать, менять ему размеры, стилизовать. Все это делается с помощью CSS [2].

Достоинства:

* Исключительно компактный код, который отлично читается;
* К блокам легче привязывать различные стили, чем к таблицам. Такой сайт быстрее загружается и лучше индексируется.
* Блочные элементы можно накладывать друг на друга, как слои в фотошопе;
* Блоки легко адаптировать, они легко скрываются и изменяются;
* Контейнеры div и span правильно отображаются даже в старых браузерах.

Флексбоксы (flex)

Относительно новый метод верстки. Нормальную поддержку в браузерах впервые получил с 2014 года. Теперь многие разработчики придерживаются именно этого метода. Суть его в том, что структурным элементам прописывается display: flex. Это тоже своего рода блоки, только более гибкие и функциональные [3].

К достоинствам флексбоксов можно отнести новые css-свойства, которые позволяют выстраивать разнообразные сетки и колонки без особых усилий. Также поддерживается вертикальная центровка, которую не поддерживает обычный блок.

Пример использования Flexbox на рисунках 3 и 4.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описаниеИзображение выглядит как текст, Шрифт, снимок экрана, типография

Автоматически созданное описание

Рисунок 3 – Разметка методом flexbox

Изображение выглядит как текст, снимок экрана, Шрифт, диаграмма

Автоматически созданное описание

Рисунок 4 – Отображение в браузере разметки методом flexbox

## 

## 1.3 Обзор и обоснование выбора архитектуры информационной системы, способа клиент-серверного взаимодействия, используемой СУБД, инструментальных средств разработки

Наиболее известные подходы к архитектуре:

– Многослойная архитектура (Layered Architecture);

– Многоуровневая архитектура (Tiered Architecture);

– Сервис-ориентированная архитектура (Service Oriented Architecture — SOA);

– Микросервисная архитектура (Microservice Architecture) [1].

Многослойная архитектура работает по принципу разделения ответственностей. Программное обеспечение разделено на слои, лежащие друг на друге, и каждый из них выполняет определенную обязанность. Преимущества: более простая реализация по сравнению с другими подходами; предлагает абстракцию благодаря разделению ответственностей между уровнями; изолирование защищает одни слои от изменений других; повышает управляемость программного обеспечения за счет слабой связанности. Недостатки: не предлагает большой масштабируемости; программное обеспечение, созданное с таким подходом, будет иметь монолитную структуру, усложняющую внесение модификаций; данные должны проходить по каждому слою, даже если нет необходимости передавать их с определенных слоев.

Многоуровневая архитектура разделяет комплекс программного обеспечения на уровни по принципу взаимодействия “клиент-сервер”. Архитектура может иметь один, два и больше уровней, разделяющих ответственности между поставщиком данных и потребителем. Этот подход использует шаблон Request Response для связи между уровнями. В отличие от многослойной архитектуры, он предлагает масштабируемость, которая может быть как горизонтальной (масштабирование сети с помощью высокопроизводительных узлов), так и вертикальной (масштабирование каждого узла путем повышения его производительности).

Сервис-ориентированная архитектура состоит из компонентов и приложений, которые связываются друг с другом с помощью четко определенных сервисов. Клиент отправляет запрос с использованием стандартного протокола и формата данных по сети. Этот запрос обрабатывается ESB (enterprise service bus — сервисная шина предприятия), которая считается сердцем сервис-ориентированной архитектуры и отвечает за оркестровку и маршрутизацию. С помощью сервисного репозитория ESB направляет запрос в специальный сервис, который может взаимодействовать с другими сервисами и базами данных, чтобы составить полезную нагрузку (данные) ответа [4].

Микросервисная архитектура разрабатывается как набор небольших сервисов, каждый из которых работает в собственном процессе и связывается с легковесными механизмами, обычно API для HTTP-ресурса. Преимущества:

предлагает слабую связанность благодаря высокой степени изоляции; повышает модульность; cбой в одном сервисе не затронет всю систему, поскольку они изолированы; предлагает высокую гибкость и масштабируемость; простота модификации может ускорить итерации; позволяет реализовать улучшенную систему обработки ошибок; решает проблемы с потоками данных, которые бывают у многослойной архитектуры. Недостатки: повышенный риск сбоя при обмене данными между сервисами; большим количеством сервисов трудно управлять; требует решения таких проблем, как задержки в сети, балансировка нагрузки и прочих трудностей, свойственных распределенной архитектуре; нуждается в комплексном тестировании в распределенной среде; на реализацию потребуется гораздо больше времени.

Для реализации проекта выбрана трёхуровневая архитектура приложения, представленная на рисунке 1.

Изображение выглядит как электроника, дизайн

Автоматически созданное описание

Рисунок 5 – Схема трехуровневой архитектура

Хранение данных будет осуществляться на стороне севера в серверной системе управления базами данных – это снизит требования к клиентскому оборудованию (требуется меньший объем накопителя).

Обработка запросов на получение статических файлов и данных из системы управления базами данных будет производиться на сервере.

Клиент и сервер осуществляют запросы посредством протокола HTTP. В целях обеспечения большей безопасности рекомендуется использовать расширение протокола HTTPS для поддержки шифрования данных [4].

Далее, необходимо выбрать систему управления базами данных. Самыми популярными и проверенным базами данных являются SQL-подобные базы, однако в последнее время свою популярность набирают документоориентированные СУБД.

Рассмотрим два самых популярных представителя двух лагерей, результаты приведены в таблице 1.

Таблица 1 – Сравнение систем управления базами данных MySQL и MongoDB

| Критерий | MySQL | MongoDB |
| --- | --- | --- |
| Краткое описание | Широко используемая свободная [реляционная система управления базами данных](https://ru.bmstu.wiki/%D0%A0%D0%B5%D0%BB%D1%8F%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D0%A1%D0%A3%D0%91%D0%94) | Одно из наиболее популярных документных хранилищ |
| Основная модель хранения данных | Реляционная база данных | Документоориентированная база данных |
| Схема данных | Да | Свободная |
| Типизация | Да | Да |
| SQL | Да | Нет |
| API и другие методы доступа | Проприентарное нативное API, ADO.NET, JDBC, ODBC | Проприетарный протокол с использованием [JSON](https://ru.bmstu.wiki/JSON_(JavaScript_Object_Notation)) |
| Триггеры | Да | Нет |
| Методы разбиения | Горизонтальное разбиение, шардинг с MySQL Cluster или MySQL Fabric | Шардинг |
| Концепции согласования | Немедленное согласование | Согласованность в конечном счёте, Немедленное согласование |
| Внешние ключи | Да | Нет |
| Параллелизм | Да | Да |

Для реализации в проекте хранения данных о толщине материнских рулонов выбрана СУБД MongoDB обладающая весомыми преимуществами в рамках, а именно:

– MongoDB, предлагает динамическую структуру данных, которые хранятся документо-ориентированно, такая гибкость означает следующее: можно создавать документы, не задавая их структуру заранее, каждый документ может обладать собственной структурой, у каждой базы данных может быть собственный синтаксис, можно добавлять поля прямо во время работы с данными. Таким образом MongoDB позволяет гибко работать со схемой данных без необходимости изменять сами данные;

– База данных MongoDB горизонтально масштабируема что позволяет легко уменьшить нагрузку на сервера при больших объёмах данных. Это означает, что увеличивать трафик можно, распределяя его или добавляя больше серверов к СУБД. В этом случае система может стать куда больше и мощнее;

– СУБД не нуждается в отдельном администраторе базы данных. Благодаря достаточному удобству в использовании ей легко могут пользоваться как разработчики, так и системные администраторы;

– Высокая производительность при выполнении простых запросов [5].

Таблица 2 – Сравнительная характеристика языков программирования

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ЯП | Типизация | Серверная часть | Диаграммы | Разработки GUI |
| C# | Статическая | .Net Core | Charting | SpaceVIL, AvaloniaUI |
| JavaScript | Динамическая | Node.js | Chart.js | React, Vue |
| Python | Динамическая | Сокеты | Matplotlib | Qt, GTK, Kivy |

C# – это объектно-ориентированный язык программирования, разработанный компанией Microsoft. Он часто используется для создания приложений под платформу Windows, веб-приложений и игр с использованием технологии Unity.

JavaScript – это язык программирования, используемый для создания интерактивных веб-страниц. Он также часто используется для создания веб-приложений и игр.

Python – это высокоуровневый язык программирования, известный своей простотой и читаемостью. Он широко используется для разработки веб-приложений, анализа данных, научных вычислений и автоматизации задач.

В целом, C# и Python являются универсальными языками программирования, которые могут использоваться для различных типов приложений, в то время как JavaScript является специализированным языком для веб-разработки..

По результатам сравнения был выбран язык JavaScript для разработки как клиентской и серверной части (с использованием платформы Node.js).

Node.js – кросплатформенная среда выполнения с открытым исходным кодом, которая позволяет создавать серверные инструменты и приложения использованием языка JavaScript. Node.js предназначена для использования вне контекста браузера, поэтому она исключает API-интерфейсы JavaScript для браузера и добавляет поддержку более традиционных OS API-интерфейсов, включая библиотеки HTTP и файловых систем.

Express – это минималистичный и гибкий веб-фреймворк для приложений Node.js, предоставляющий обширный набор функций для мобильных и веб-приложений. Имея в своем распоряжении множество служебных методов HTTP и промежуточных обработчиков, создать надежный API можно быстро и легко. Express предоставляет тонкий слой фундаментальных функций веб-приложений, которые не мешают работать c функциями Node.js.

Для парсинга вакансий и загрузки в базу данных был выбран язык программирования Python.

Python — высокоуровневый язык программирования. Благодаря ясному синтаксису его код понятен и легко читается. Python часто называют «исполнимым псевдокодом»: на нём можно писать код быстрее, чем на других языках.

Python поддерживает несколько стилей программирования — можно выбрать наиболее подходящий в зависимости от задач. Например, если необходимо моделировать сложную систему, удобнее использовать объектно-ориентированное программирование. А если работаем с данными, то предпочтительным будет функциональное.

# **2** **ТЕХНОЛОГИЧЕСКАЯ ЧАСТЬ**

## 2.1 Разработка функциональной структуры информационной системы с учётом клиент-серверной архитектуры. Постановка задачи разработки

Функциональная структура сайта для анализа данных с толщиномеров представлена на рисунке 6. Функциональная структура представлена в виде модели клиент-сервер. Клиент может отправлять на сервер HTTP запросы Ruquest и получать от сервера HTTP ответы Response.

Изображение выглядит как текст, диаграмма, План, снимок экрана

Автоматически созданное описание

Рисунок 6 – Функциональная структура веб-приложения

Постановку задачи разработки:

Требуется разработать веб-приложение, позволяющее пользователю:

– просмотр таблиц с вакансиями;

– просмотр анализа вакансий в виде диаграмм.

## 2.2 Разработка UML-диаграммы вариантов использования программного комплекса для студента

На рисунке 7 представлена UML-диаграммы вариантов использования веб-приложения для пользователя.

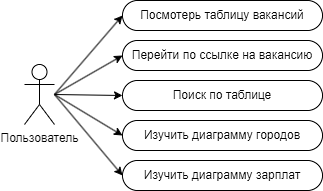


Рисунок 7 – UML-диаграммы вариантов использования веб-приложения

## 2.3 Разработка макета графического интерфейса информационной системы

Разрабатываемое веб-приложение будет состоять из одной страницы. На рисунке 8 представлен схематичный макет страниц веб-приложения пользовательского интерфейса.

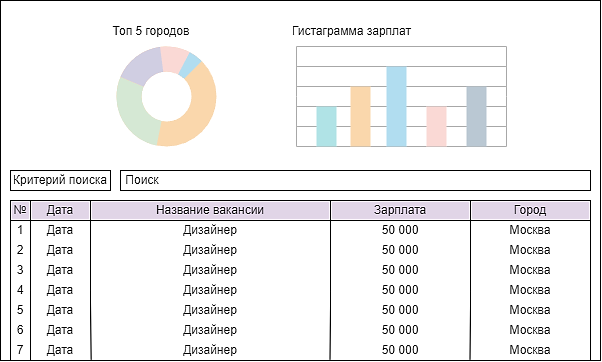


Рисунок 8 – Схематичный макет страницы

## 2.4 Разработка блок-схемы алгоритма парсинга

На рисунке 9 представлена блок-схема алгоритма парсинга вакансий.

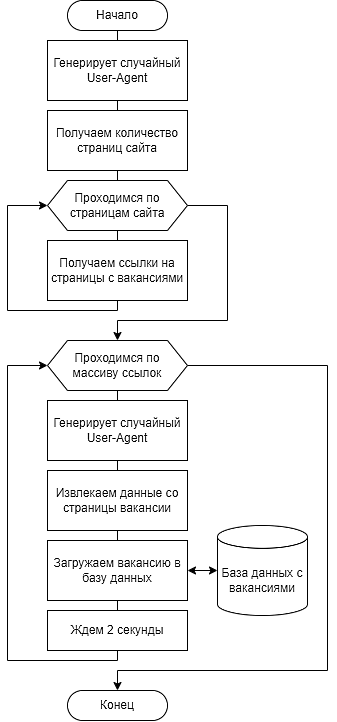


Рисунок 9 – Блок-схема алгоритма парсинга

## 2.5 Проектирование серверного интерфейса взаимодействия с клиентом и СУБД

В таблице 3 представлены описания API обращений клиента к серверу по основным запросам.

Таблица 3 – Описание API обращения клиента к серверу по основным запросам

|  |  |  |
| --- | --- | --- |
| Метод HTTP запроса | Путь запроса | Назначения запроса |
| GET | /main | Рендеринг шаблона "vacancy" с передачей данных о вакансиях, городах и зарплатах в виде объекта в качестве контекста шаблона. |
| GET | /data1 | Сервер получит данные о городах и отправит их в качестве ответа. |
| GET | /data2 | Сервер получит данные о зарплатах и отправит их в качестве ответа. |

Для взаимодействия с выбранной СУБД на серверной стороне используется драйвер mongoose.

const mongoose = require('mongoose');

mongoose.Promise = global.Promise;

// Подключаемся к базе данных MongoDB с помощью Mongoose

db = mongoose.connect('mongodb://localhost:27017/mydatabase');

mongoose.connection

.once('open', () => console.info('Подключено к базе данных MongoDB mydatabase'))

.on('error', err => console.error(err));

const {Schema, model}=require('mongoose')

const vacancySchema=new Schema({

date:{type:String},

link:{type:String},

name:{type:String},

salary:{type:String},

city:{type:String}

})

const app = express();

app.set('views', path.join(\_\_dirname, 'views'));

app.set('view engine', 'ejs');

app.use(express.static(\_\_dirname + '/public'));

app.use(bodyParser.json());

app.use(bodyParser.urlencoded({ extended: true }));

app.use(session({ secret: 'your-secret-key', resave: false, saveUninitialized: false, cookie: {maxAge: 1000\*60\*24\*7} }));

app.get("/main", async (req, res) => {

try {

const vacancy = await Vacancy.find({});

const data\_сity = await Cities;

const data\_salary = await Salaries;

res.render('vacancy', { vacancies: vacancy, data\_сity: data\_сity, data\_salary: data\_salary });

} catch (err) {

console.error(err);

res.status(500).send('Internal Server Error');

}

});

app.get('/data1',async function(req,res) {

const data\_сity = await Cities;

res.send(data\_сity)

})

app.get('/data2',async function(req,res) {

const data\_salary = await Salaries;

res.send(data\_salary)

})

// Запускаем экспресс-сервер

const port = process.env.PORT || 3000;

app.listen(port, () => {

console.log(`Server is running on http://localhost:${port}/main`);

});

## 2.6 Разработка базы данных информационной системы

Информационное обеспечение состоит из базы данных mydatabase. Внутри БД есть одна коллекция vacancy.

Таблица 4 – Представленные типы данных в БД

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Значение |
| \_id | Object\_ID | Уникальный идентификатор документа |
| date | String | Дата публикации вакансии |
| link | String | Ссылка на вакансию |
| name | String | Название вакансии |
| salary | String | Зарплата |
| city | String | Город |

## 2.7 Разработка программного обеспечения информационной системы

Структура программного обеспечения для балансировки химических реакций представлена на рисунке 10.

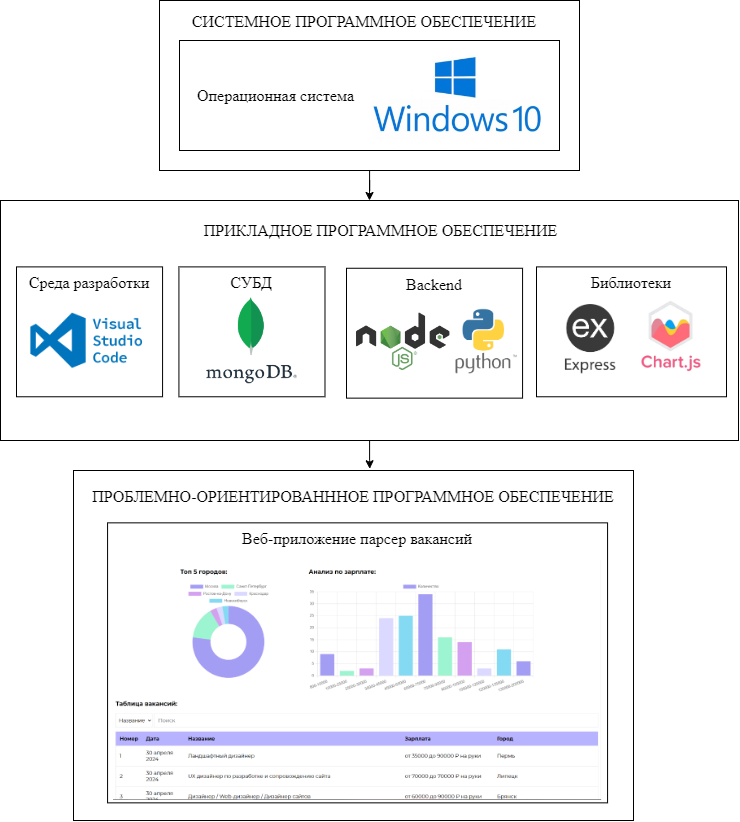


Рисунок 10 – Структура программного обеспечения.

Требования к ЭВМ, необходимой для нормального функционирования веб-приложения, представлены в таблице 5.

Таблица 5 – Минимальные системные требования

| **Показатель** | **Значение** |
| --- | --- |
| Тип ЭВМ | Персональный компьютер |
| Тактовая частота процессора, ГГц | 1 |
| Объем оперативной памяти, ГБ | 1 |
| Объем внешней памяти, ГБ | 16 |
| Состав и характеристика периферийных устройств ЭВМ | Клавиатура, мышь, монитор с разрешением 800 × 600 пикселей |
| Состав и характеристика сетевого оборудования | Сетевой адаптер |
| Операционная система | Windows 10 |
| Прикладное программное обеспечение, необходимое для функционирования программного комплекса | Visual Studio Code 1.43.0 |

Код реализации парсера вакансий на Python:

def get\_vacancy(link):

ua = fake\_useragent.UserAgent()

data = requests.get(

url=link,

headers={"user-agent":ua.random}

)

if data.status\_code != 200:

return

soup = BeautifulSoup(data.content, "lxml")

try:

name = soup.find(attrs={"data-qa":"vacancy-title","class":"bloko-header-section-1"}).text

except:

name = ""

try:

salary = soup.find("div", attrs={"data-qa":"vacancy-salary"}).find("span", attrs={"class":"bloko-header-section-2\_lite"}).text.replace("\xa0", "")

except:

salary = ""

try:

tags = [tag.text.replace("\xa0", " ") for tag in soup.find(attrs={"class":"bloko-tag-list"}).find\_all(attrs={"class":"bloko-tag\_\_section\_text", "data-qa":"bloko-tag\_\_text"})]

except:

tags = []

date = soup.find("p", attrs={"class":"vacancy-creation-time-redesigned"}).find("span").text.replace("\xa0", " ")

try:

city = soup.find(attrs={"data-qa":("vacancy-view-location", "vacancy-view-raw-address")}).text.split(",")[0]

except:

city = ""

vacancy = {

"link":link,

"name":name,

"salary":salary,

"date":date,

"city":city,

"tags":tags

}

return vacancy

Код реализации поиска наибольших городов на NodeJS:

async function findMostCommonCities() {

try {

const result = await Vacancy.aggregate([

{

$group: {

\_id: '$city', // Группировка по полю 'city'

count: { $sum: 1 } // Подсчет количества записей для каждого города

}

},

{

$sort: { count: -1 } // Сортировка по убыванию количества

},

{

$limit: 5 // Ограничение на 6 записей

}

]);

const cities = result.map(item => ({

name: item.\_id.toString(),

count: item.count.toString()

}));

return cities;

} catch (error) {

console.error('Ошибка при выполнении запроса:', error);

}

}

Код реализации кольцевой диаграммы на JavaScript:

fetch('/data1')

.then(response => response.json())

.then(data\_city => {

const ctx = document.getElementById('myChart\_city').getContext('2d');

const name\_city = data\_city.map(item => item.name);

const count\_city = data\_city.map(item => item.count);

new Chart(ctx, {

type: 'doughnut',

data: {

labels: name\_city,

datasets: [{

label: 'Количество',

data: count\_city,

borderWidth: 0,

backgroundColor: ['#a49df4', '#9df4d0',

'#D4A1F0', '#DBD9FF', '#83DAF2']

}]

},

});

});

## 2.8 Тестирование программного комплекса на примере вакансий сайта HeadHanter.ru полученных по ключевому слову «Дизайнер»

На рисунке 11 представлено отображение интерфейса страницы в браузере.

Изображение выглядит как текст, снимок экрана, Шрифт, диаграмма

Автоматически созданное описание

Рисунок 11 – Интерфейс веб-приложения

На рисунке 12 представлено отображение кольцевой диаграммы городов, в которых больше всего вакансий Дизайнера.

Изображение выглядит как текст, снимок экрана, круг, Шрифт

Автоматически созданное описание

Рисунок 12 – Диаграмма топ 5 городов

На рисунке 13 представлено отображение кольцевой диаграммы диапазон зарплат вакансий Дизайнера.

Изображение выглядит как текст, снимок экрана, диаграмма, График

Автоматически созданное описание

Рисунок 13 – Диаграмма зарплаты

На рисунке 14 представлено отображение таблицы вакансий Дизайнера.

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

Рисунок 14 – Таблица вакансий

На рисунке 15 представлено отображение результата поиска по названию в таблице вакансий Дизайнера.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 15 – Тестирование поиска по названию вакансии

# ВЫВОДЫ ПО РАБОТЕ

В ходе выполнения курсового проекта разработан сайт для анализа вакансий дизайнера, полученных с сайта HeadHunter.ru:

* на основании аналитического обзора разработана функциональная структура веб-приложения с учётом клиент-серверной архитектуры и сформулирована задача разработки веб-приложения;
* разработана UML-диаграмма вариантов использования веб-приложения для пользователя;
* разработаны макеты графического интерфейса веб-приложения;
* разработана блок-схема алгоритма парсера вакансий с HeadHunter.ru;
* спроектирован и описан серверный интерфейс взаимодействия с клиентской подсистемой и СУБД MongoDB;
* разработана база данных вакансий веб-приложения;
* разработано программное обеспечение для вывода анализа вакансий в виде диаграмм;
* проведено тестирование веб-приложения.

В дальнейшем, чтобы улучшить работоспособность и расширить функционал программы, необходимо реализовать возможность введения разных профессий.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Воронежский ГАУ: цифровой ресурс: сайт. – Воронеж. 2001 – . – URL: http://iomas.vsau.ru/wp-content/uploads/uch\_proz/ei/txt/internet/start.htm (дата обращения 30.10.2023). – Режим доступа : свободный доступ

2 TestMatick: цифворой ресурс: сайт. – США. 2001 – . – URL: https://testmatick.com/ru/ (дата обращения 30.10.2023). – Режим доступа : свободный доступ

3 Selectel.ru: цифровой ресурс: сайт. – США. 2000 – . – URL: https://selectel.ru/blog (дата обращения 30.10.2023). – Режим доступа : свободный доступ.

4 Таненбаум, Э. Компьютерные сети : учебное пособие / Э. Таненбаум, Н. Фимстер, Д. Уэзеролл – 6-е издание. – СПб. : "Питер", 2023. – 992 с. – ISBN 978-5-4461-1766-6

5 Таненбаум, Э. Современные операционные системы : учебное пособие / Э. Таненбаум. – 4-е издание. – СПб. : "Питер", 2015. – 1120 с. – ISBN 978-5-4461-1155-8

6 Руссинович, М. Внутреннее устройство Windows : учебное пособие / М. Руссинович, Д. Соломон, А. Ионеску. – 7-е изд. – СПб. : "Питер", 2022. – 944 с. – ISBN 978-5-4461-0663-9

7 Щупак, Ю.А. Win32 API. Эффективная разработка приложений: учебное пособие / Ю.А Щупак – СПб. : "Питер", 2011. – 592 с. – ISBN 978-5-388-00301-0

8 Троелсен, Д. Язык программирования C# 9 и платформа .NET 5: основные принципы и практики программирования: учебное пособие / Д. Троелсен. – 10-е издание – М. : Диалектика, 2022. – 1392 с. – ISBN 978-5-907458-67-3

9 Джозеф, А. C# 9.0. Карманный справочник / А. Джозеф, А. Бен. – СПб : Диалектика, 2021. – 256с. – ISBN 978-5-907365-36-0.