АО «[КАЗАХСКИЙ УНИВЕРСИТЕТ ТЕХНОЛОГИИ и БИЗНЕСА имени К.КУЛАЖАНОВА](https://kaztbu.edu.kz/)»

КАФЕДРА «ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ»

**Отчет по научно-исследовательской практике**

Магистранта 2 курса (период обучения 2 года)

Технологического факультета

по ОП 7М06136 – Информационные системы

Тулегенов Алишер Канатович

|  |
| --- |
| Место практики: ТОО «NTS Design» |
| Начало практики: 29.01.2024г. |
| Окончание практики: 26.04.2024г. |
| Руководитель практики: Акишев Каршыга Максутович, к.т.н., асс. профессор |
| от предприятия: Хапез Акжол Зиядаулы |
| отчет защищен с оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| «\_\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_ 2024г. |

**Астана, 2024**

**СОДЕРЖАНИЕ**

|  |  |
| --- | --- |
| 1 Цель и задачи исследовательской практики | 3 |
| 1.1 Характеристика базы практики (краткое описание выполненных работ и сроки их осуществления) | 3 |
| 1.2. Общая информация о компании | 4 |
| 2 Основная часть | 4 |
| 2.1. Исследование и проделанная работа | 4 |
| Заключение | 9 |
| Список использованной литературы | 10 |

1. **Цель и задачи научно-исследовательской практики**

Научно-исследовательская практика магистратов является обязательной составной частью образовательной программы подготовки магистров.

**Цель** **научно-исследовательской** практики состоит в формировании у магистрантов навыков и умений, необходимых для организации и проведения научных исследований, связанных с подготовкой магистерской диссертации, а также занятия научными исследованиями на стадии поствузовского образования и работы в научно-исследовательских, образовательных и культурно-просветительских учреждениях и организациях.

**Задачи исследовательской практики**. В ходе прохождения практики магистрант должен:

* Ознакомиться с формами и приемами организации научно-библиографического поиска (в том числе по электронным каталогам и через интернет)
* Освоить методику работы с историческими источниками, необходимыми для написания магистерской диссертации
* Освоить правила и требования к оформлению текста научного исследования, научно-справочного аппарата.

**1.1 Характеристика базы практики (краткое описание выполненных работ и сроки их осуществления)**

Место и период  прохождения практики

Я, Тулегенов Алишер Канатович, магистрант 2 курса Технологического факультета АО «Казахский университет технологии и бизнеса» проходил исследовательскую практику с 29 января по 26 апрель 2024г. ТОО «NTS Design» города Астана.

Место нахождения организации: г. Астана, улица Иманова, дом 9

**1.2 Общая информация о компании**

«NTS Design» — современная научно-производственная компания, занимающаяся разработкой и производством специальных технических устройств в сфере безопасности. Предприятие имеет собственную опытно-конструкторскую лабораторию, автоматизированную производственную линию, а также отдел технического обслуживания и ремонта. Компания имеет лицензию на разработку, производство и ремонт специальных технических средств для аварийных работ.

Особенности компании:

1) Штат частных инженеров и опытно-конструкторской лаборатории;

2) Разработка устройств по индивидуальным заказам заказчика;

3) Расположение производства на территории Республики Казахстан.

**2.1 Основная часть научного исследования**

С самого начала моей практики я приступил к выполнению задач, поставленных моим руководителем. В процессе работы я активно использовал программы Visual Studio и Photoshop. В Visual Studio занимался разработкой программного обеспечения и созданием интерфейсов, а в Photoshop занимался созданием графических элементов и макетов. Этот опыт позволил мне освоить различные аспекты разработки и дизайна, а также эффективно использовать инструменты и ресурсы для достижения поставленных целей.



В рамках первой задачи, поставленной моим руководителем, я познакомился с процессом создания и манипулирования различными файлами с использованием Visual Studio и языка программирования C#. Особое внимание было уделено файлам формата WAV. В ходе работы я изучил методы работы с этими файлами, включая чтение, запись и обработку аудиоданных. Также был разработан интерактивный интерфейс, обеспечивающий удобное взаимодействие с файлами и выполнение различных операций над ними.

Звуковые файлы формата WAV являются одними из наиболее распространенных и широко используемых форматов для хранения аудиоданных. WAV (Waveform Audio File Format) представляет собой контейнерный формат, который может содержать аудиозаписи в цифровой форме без сжатия данных. Этот формат обеспечивает высокое качество звука и широкую совместимость с различными аудиоустройствами и программным обеспечением.

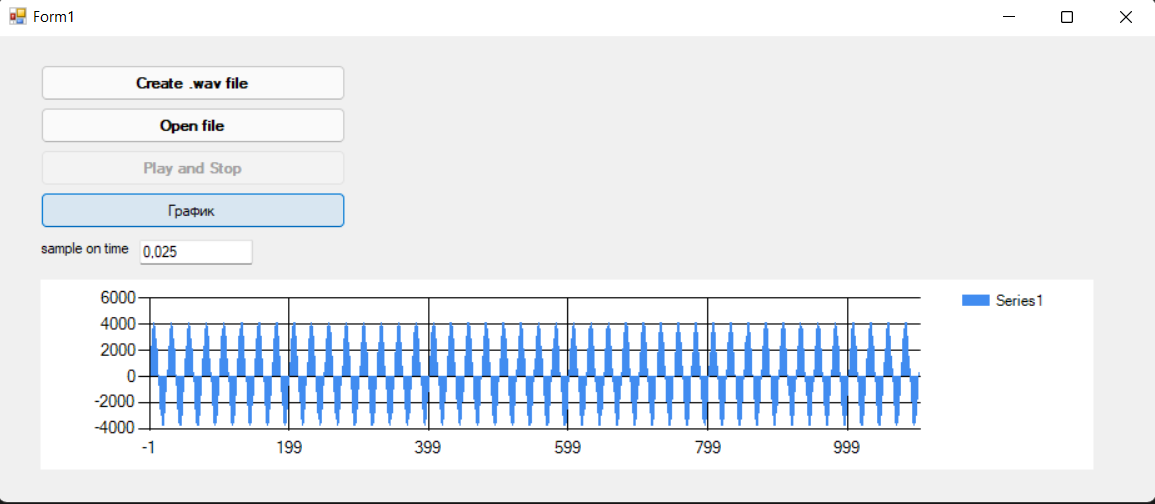
Одной из особенностей файлов WAV является их способность сохранять аудиоданные без потерь качества, поскольку они не используют сжатие. Это делает их идеальным выбором для профессионального звукозаписывающего оборудования, студий звукозаписи и аудиоинженеров, которым важна максимальная точность и сохранность звукового материала.

Еще одним преимуществом файлов WAV является их поддержка высококачественных аудиофайлов, таких как записи стереозвука с высоким разрешением (High-Resolution Audio), что делает их предпочтительным форматом для профессиональных музыкантов и звукорежиссеров.

Важно отметить, что файлы WAV могут занимать большой объем памяти из-за отсутствия сжатия, поэтому иногда для сокращения размера файлов используются альтернативные форматы сжатия, такие как MP3 или FLAC. Однако, при необходимости сохранения высокого качества звука, WAV остается предпочтительным выбором.

В общем, файлы WAV играют важную роль в области аудиозаписи и звукового дизайна благодаря своей надежности, высокому качеству звука и широкой совместимости, что делает их неотъемлемым элементом для многих профессиональных и любительских проектов, связанных с аудио.

Я выполнил данную задачу с использованием приложения для программирования интерфейса Windows Forms. На изображении изображена амплитуда аудиофайла в формате WAV.



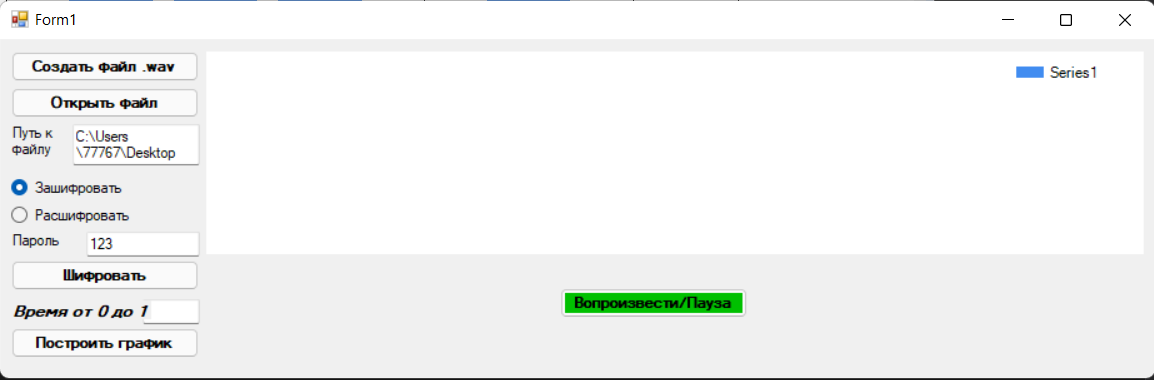
Дополнительная задача для этого проекта - шифрование и декодирование файлов.

Шифрование и декодирование файлов WAV — это процессы обеспечения безопасности и конфиденциальности аудиоданных путем преобразования их в нечитаемый формат для неавторизованных пользователей и последующего восстановления в исходное состояние.

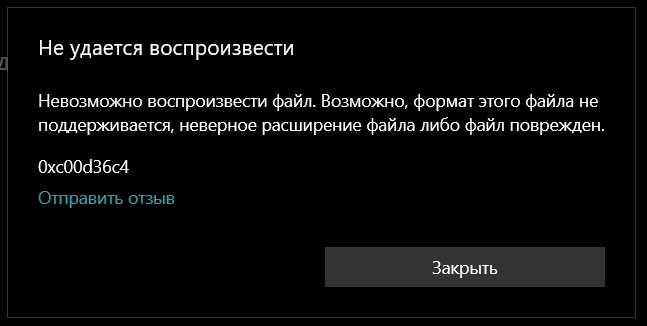
Одним из методов шифрования файлов WAV является использование алгоритмов криптографии, таких как AES (Advanced Encryption Standard) или RSA (Rivest-Shamir-Adleman). При шифровании файлы WAV преобразуются с помощью специального ключа, который делает их непонятными для посторонних лиц без знания этого ключа. Это обеспечивает защиту данных от несанкционированного доступа и перехвата.

Для декодирования зашифрованных файлов WAV необходимо обратное преобразование, используя соответствующий ключ. Только авторизованные пользователи, обладающие правильным ключом, могут успешно расшифровать файлы и получить доступ к оригинальным аудиоданным. Процесс декодирования обеспечивает восстановление файлов в их исходное состояние, пригодное для прослушивания или обработки.

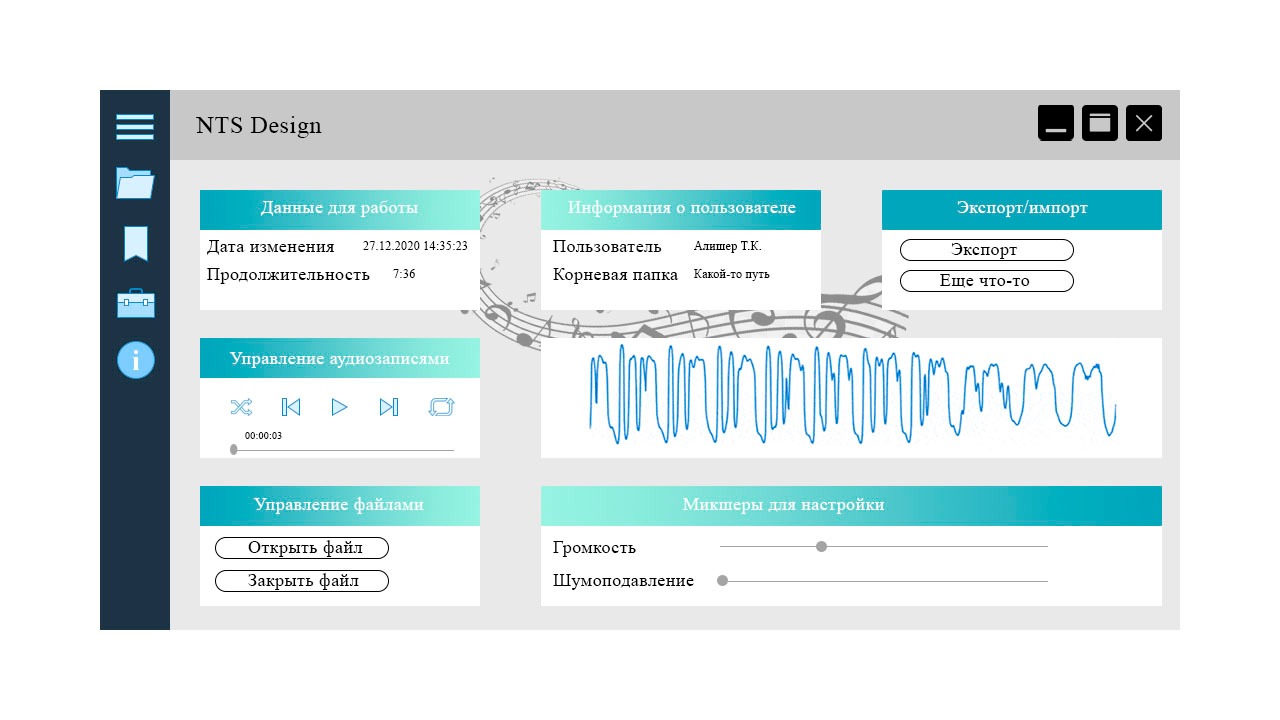
Шифрование и декодирование файлов WAV играют важную роль в защите конфиденциальной информации, такой как коммерческие записи, личные аудиозаписи или конфиденциальные беседы. Они обеспечивают безопасность данных в случае передачи по сети или хранения на устройствах хранения данных, таких как жесткие диски или облачные сервисы.



Результат шифрования



Следующим этапом было создание образцов макетов интерфейса с использованием Photoshop. За время работы я освоил работу с этой программой и разработал несколько макетов. Пример одного из первых макетов:



В рамках следующего этапа моей работы была поставлена задача реализовать последнюю версию макета с помощью программы Visual Studio. Для этого я воспользовался инструментом WPF в Visual Studio, используя .NET Framework.

В рамках моей исследовательской практики я изучал использование WPF (Windows Presentation Foundation) в Visual Studio с использованием .NET Framework. WPF представляет собой мощный инструмент для разработки графических пользовательских интерфейсов (GUI) для приложений под платформу Windows.

Visual Studio, в свою очередь, обеспечивает интегрированную среду разработки (IDE) для создания приложений с использованием различных технологий, включая WPF. С его помощью разработчики могут создавать, отлаживать и развертывать приложения с использованием различных инструментов и ресурсов, предоставляемых .NET Framework.

.NET Framework, являясь платформой для разработки и выполнения приложений на языке программирования C#, Visual Basic и других языках, обеспечивает широкий набор библиотек и инструментов для создания разнообразных приложений, включая те, которые используют WPF для построения пользовательского интерфейса.

Изучение данной комбинации технологий позволило мне лучше понять принципы разработки современных приложений под Windows, а также научиться эффективно использовать инструменты Visual Studio и возможности WPF для создания удобных и функциональных пользовательских интерфейсов. Эти знания и навыки окажутся весьма полезными для моей дальнейшей карьеры в области разработки программного обеспечения.

Я думаю, что получившаяся программа была положительной. Однако, из-за запроса менеджера, я не могу предоставить здесь конечный результат.

Изучение HTML, CSS и JavaScript

Начало моего погружения в веб-разработку было положено изучением основных технологий этой области: HTML, CSS и JavaScript. Перед началом изучения каждого из этих языков я столкнулся с вопросом, какие конкретно задачи могли быть мне поставлены для достижения целей исследовательской практики.

**Изучение веб-разработки**

HTML (HyperText Markup Language) был первым языком, который я решил изучить. Мой подход начался с ознакомления с основными концепциями языка, такими как теги, элементы, атрибуты и структура документа. Целью было понять, как HTML используется для создания структуры веб-страницы и организации её содержимого.

В процессе изучения HTML мне предоставлялись задачи на создание базовых веб-страниц с использованием различных элементов и атрибутов. Это помогло мне понять, как правильно структурировать информацию на странице, использовать заголовки, параграфы, списки и другие элементы.

Теги и элементы: Основа HTML - это теги и элементы, которые определяют структуру и содержание веб-страницы. Изучение включало в себя знакомство с различными типами тегов, такими как <div>, <p>, <h1>-<h6>, <ul>, <ol>, <li>, <img> и другими.

Атрибуты: Каждый HTML-элемент может иметь атрибуты, которые задают дополнительные характеристики элемента. Изучение включало в себя понимание таких атрибутов, как id, class, src, href, alt, title и другие, и их использование для добавления функциональности и стилей к элементам.

Структура документа: HTML-документ имеет определённую структуру, которая включает в себя элементы <html>, <head> и <body>. Изучение включало в себя понимание роли каждого из этих элементов и их взаимодействия для создания корректного документа.

Ссылки и изображения: Изучение включало в себя работу с элементами <a> для создания ссылок и элементами <img> для вставки изображений. Это включало в себя такие аспекты, как добавление ссылок на другие веб-страницы, абсолютные и относительные пути, а также использование атрибута alt для описания изображений.

Списки: HTML позволяет создавать упорядоченные и неупорядоченные списки с помощью тегов <ul>, <ol> и <li>. Изучение включало в себя различные способы создания и оформления списков для организации содержимого веб-страницы.

После того как я освоил основы HTML, я перешёл к изучению CSS (Cascading Style Sheets) - языка, который используется для оформления и стилизации веб-страниц. Задачи, поставленные передо мной на этом этапе, включали в себя создание стилей для различных элементов веб-страницы: изменение цвета текста, фона, шрифтов, добавление отступов и рамок.

Изучение CSS помогло мне понять, как создавать эстетически приятный и удобочитаемый дизайн веб-страницы, что играет важную роль в привлечении пользователей и улучшении пользовательского опыта.

Основы стилей: CSS используется для стилизации элементов HTML и определения их внешнего вида на веб-странице. Изучение начиналось с основных концепций CSS, таких как селекторы, свойства и значения, которые определяют внешний вид элементов.

Типы селекторов: В CSS существует несколько типов селекторов, позволяющих выбирать элементы для применения стилей. Изучение включало в себя знакомство с основными селекторами, такими как элементные селекторы, классовые селекторы, идентификаторы и дочерние селекторы.

Оформление текста и шрифтов: CSS предоставляет возможность изменять внешний вид текста на веб-странице, включая его размер, цвет, шрифт и стиль. Изучение включало в себя работу с свойствами font-size, font-family, color, font-weight и другими для создания эстетически приятного дизайна текста.

Модель блока и внешний вид элементов: CSS определяет модель блока, которая описывает, как элементы размещаются и отображаются на веб-странице. Изучение включало в себя работу с свойствами display, position, margin, padding и border, которые определяют расположение и отступы элементов.

Фон и изображения: CSS предоставляет возможность добавления фоновых изображений и цветов к элементам HTML. Изучение включало в себя работу с свойствами background-color, background-image, background-repeat и background-position для создания эффектных фоновых элементов.

Плавающие элементы и позиционирование: CSS позволяет создавать сложные макеты веб-страниц с помощью плавающих элементов и абсолютного позиционирования. Изучение включало в себя работу с свойствами float, clear, position, top, bottom, left и right для создания разнообразных макетов.

Последним этапом моего изучения было знакомство с JavaScript, языком программирования, который добавляет интерактивность и динамические элементы на веб-страницу. Задачи, связанные с JavaScript, включали в себя создание интерактивных элементов, таких как кнопки, формы, анимации и валидация данных.

Изучение JavaScript позволило мне понять, как создавать динамические веб-страницы, которые реагируют на действия пользователя, и обогащать пользовательский опыт новыми функциональными возможностями.

Основы языка: JavaScript является языком программирования, который добавляет интерактивность и динамическое поведение на веб-страницы. Изучение начиналось с основных концепций языка, таких как переменные, типы данных, операторы и функции.

Переменные и типы данных: JavaScript поддерживает различные типы данных, такие как числа, строки, булевы значения, массивы, объекты и другие. Изучение включало в себя работу с объявлением переменных, присваиванием значений и операциями над ними.

Управляющие конструкции: JavaScript предоставляет управляющие конструкции, такие как условные операторы (if, else if, else), циклы (for, while, do...while), операторы switch и другие, которые позволяют выполнять различные действия в зависимости от условий.

Функции: Функции являются основным строительным блоком JavaScript и позволяют создавать многократно используемые блоки кода. Изучение включало в себя создание функций, передачу параметров, возврат значений и работу с областями видимости.

Работа с событиями: JavaScript позволяет создавать интерактивные веб-страницы, реагирующие на действия пользователя с помощью событий. Изучение включало в себя работу с различными типами событий, такими как клики мыши, нажатия клавиш, загрузка страницы и другие, а также привязку обработчиков событий к элементам HTML.

Манипуляция DOM: DOM (Document Object Model) представляет структуру HTML-документа в виде древовидной структуры объектов, с которой JavaScript может взаимодействовать. Изучение включало в себя работу с DOM для добавления, удаления и изменения элементов и их атрибутов на веб-странице.

AJAX и взаимодействие с сервером: JavaScript позволяет создавать динамические веб-приложения, которые взаимодействуют с сервером без перезагрузки страницы с помощью технологии AJAX (Asynchronous JavaScript and XML). Изучение включало в себя отправку асинхронных запросов на сервер, обработку ответов и обновление содержимого страницы без её перезагрузки.

**Адаптивная верстка**

Адаптивная верстка - это методология разработки веб-страниц, которая обеспечивает оптимальное отображение контента на различных устройствах и разрешениях экрана. Основная идея адаптивной верстки состоит в том, чтобы создавать веб-страницы, которые адаптируются к размеру экрана устройства, на котором они отображаются, обеспечивая при этом удобочитаемость и удобство использования для пользователей.

Основные принципы адаптивной верстки включают использование относительных единиц измерения, таких как проценты или em, вместо абсолютных пикселей для задания размеров и расположения элементов. Это позволяет элементам масштабироваться в зависимости от размера экрана устройства.

Другим важным аспектом адаптивной верстки является использование медиа-запросов CSS. Медиа-запросы позволяют изменять стили элементов в зависимости от различных характеристик устройства, таких как ширина экрана, плотность пикселей или ориентация экрана. Это позволяет создавать различные варианты отображения контента для различных устройств и разрешений экрана.

Также важным аспектом адаптивной верстки является использование гибкой сетки. Гибкая сетка позволяет автоматически адаптировать расположение элементов на странице в зависимости от размера экрана, обеспечивая оптимальное использование доступного пространства.

Кроме того, для обеспечения оптимального отображения контента на мобильных устройствах часто используются такие техники, как скрытие или изменение некоторых элементов на маленьких экранах, использование адаптивных изображений и оптимизация скорости загрузки страницы.

В целом, адаптивная верстка играет ключевую роль в создании современных веб-страниц, которые могут эффективно отображаться на различных устройствах и обеспечивать удобное пользовательское взаимодействие. Она позволяет создавать многофункциональные и гибкие веб-приложения, которые могут быть доступны пользователям независимо от их устройства и разрешения экрана.



**Создание архитектур веб-приложений**

Создание архитектур веб-приложений - это важный этап разработки, который определяет структуру, организацию и взаимодействие различных компонентов приложения. Основные принципы, которые руководят созданием архитектур веб-приложений, включают модульность, масштабируемость, гибкость и производительность.

Модульная структура является ключевым аспектом архитектуры веб-приложений. Она позволяет разбить приложение на небольшие, независимые компоненты или модули, что упрощает поддержку, тестирование и масштабирование приложения. Каждый модуль может быть разработан и поддерживаться отдельно, что способствует повышению производительности и гибкости разработки.

Еще одним важным аспектом архитектуры веб-приложений является выбор архитектурного шаблона. Один из наиболее распространенных шаблонов - это MVC (Model-View-Controller), который разделяет приложение на три основных компонента: модель (отвечает за данные и бизнес-логику), представление (отображает данные пользователю) и контроллер (управляет взаимодействием между моделью и представлением).

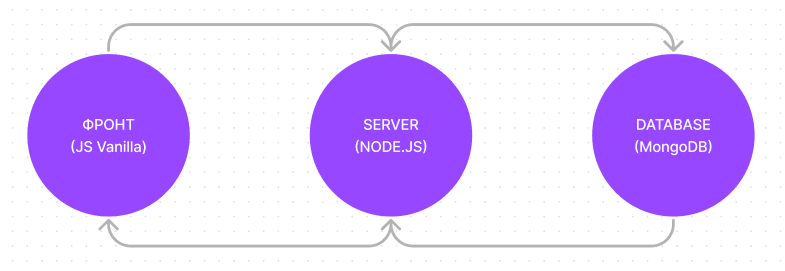
Другие распространенные архитектурные шаблоны включают MVP (Model-View-Presenter), MVVM (Model-View-ViewModel) и Clean Architecture. Каждый из них имеет свои особенности и преимущества, и выбор конкретного шаблона зависит от требований и характеристик конкретного проекта.

**Клиент-серверная архитектура**

Клиент-серверная архитектура - это распределенная архитектура, в которой клиентские и серверные компоненты взаимодействуют между собой через сеть. В этой архитектуре клиентские компоненты обычно отвечают за представление данных пользователю и взаимодействие с пользователем, в то время как серверные компоненты обрабатывают запросы от клиентов и управляют бизнес-логикой приложения.

Основные преимущества клиент-серверной архитектуры включают отделение интерфейса пользователя от бизнес-логики приложения, что упрощает разработку, тестирование и поддержку приложения. Кроме того, это позволяет легко масштабировать приложение путем добавления или изменения серверных компонентов без изменения клиентской части.

Разработка клиент-серверных веб-приложений требует тщательного планирования и проектирования архитектуры, чтобы обеспечить эффективное взаимодействие между клиентом и сервером и создать надежное и масштабируемое приложение.



**Серверная часть и Node.js**

Node.js - это среда выполнения JavaScript, позволяющая запускать JavaScript на сервере. Она основана на движке V8 от Google Chrome, что обеспечивает высокую производительность и эффективность исполнения кода. Основным преимуществом Node.js является его асинхронная модель программирования, которая позволяет обрабатывать множество запросов одновременно без блокировки потоков.

Серверная часть приложения, созданного с использованием Node.js, может быть написана на JavaScript. Это обеспечивает единый язык программирования для как клиентской, так и серверной частей приложения, что упрощает разработку и поддержку кода. Node.js также позволяет разработчикам создавать высокопроизводительные и масштабируемые серверные приложения с использованием JavaScript.

Для создания серверных приложений на Node.js используются различные фреймворки и библиотеки, такие как Express.js, Koa.js, Nest.js и другие. Они предоставляют удобные инструменты для разработки веб-серверов, API-серверов, приложений реального времени и других типов серверных приложений.

Node.js также поддерживает работу с различными типами баз данных, такими как MongoDB, MySQL, PostgreSQL и другие. Для взаимодействия с базами данных используются ORM (Object-Relational Mapping) или библиотеки для работы с базами данных.

С помощью Node.js можно создавать высоконагруженные и масштабируемые серверные приложения, что делает его популярным выбором для разработки современных веб-приложений и микросервисов.

Одним из ключевых преимуществ Node.js является его асинхронная и событийно-ориентированная модель программирования. В отличие от традиционных серверных языков, таких как PHP или Java, которые используют многопоточную модель, Node.js использует однопоточную модель событийного цикла. Это позволяет обрабатывать большое количество запросов одновременно без блокировки потоков, что делает его идеальным выбором для создания высоконагруженных и масштабируемых серверных приложений.

Для создания серверных приложений на Node.js часто используются фреймворки и библиотеки. Наиболее популярным из них является Express.js, который предоставляет минималистичный и гибкий инструментарий для создания веб-приложений и API. Кроме того, существуют такие фреймворки как Koa.js, который является более современной и легковесной альтернативой Express.js, и Nest.js, который предоставляет расширенные возможности для создания масштабируемых и модульных приложений.

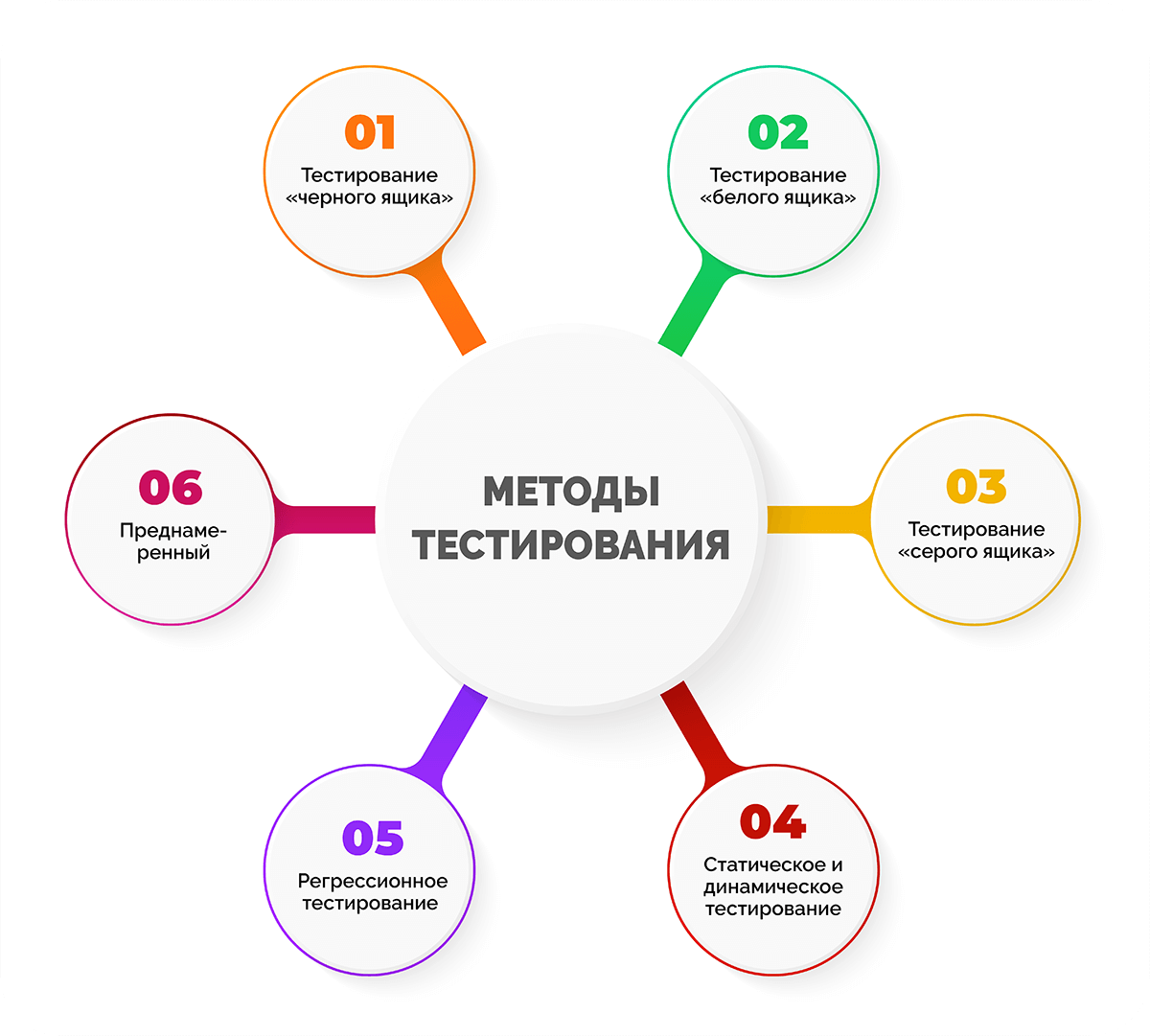
Node.js также предлагает множество модулей и библиотек для работы с базами данных, сетевыми протоколами, аутентификацией и многое другое. С его помощью можно легко создавать API-серверы, веб-приложения, приложения реального времени и другие типы серверных приложений.

В целом, Node.js открывает новые возможности для разработки серверных приложений, обеспечивая высокую производительность, гибкость и масштабируемость. Его активное сообщество и богатая экосистема инструментов делают его популярным выбором среди разработчиков по всему миру.

**Тестирование ПО**

Тестирование программного обеспечения (ПО) играет ключевую роль в разработке, обеспечивая качество и надежность конечного продукта. Его целью является обнаружение и исправление ошибок и дефектов в программном коде до его выпуска на рынок. В рамках тестирования проводятся различные виды проверок, включая функциональное тестирование, которое осуществляется для проверки соответствия программы заявленным требованиям, а также нагрузочное тестирование, направленное на проверку стабильности и производительности системы при высоких нагрузках. Юнит-тестирование используется для проверки отдельных компонентов программы, а интеграционное тестирование - для проверки взаимодействия между различными компонентами. Пользовательское тестирование выполняется конечными пользователями для проверки удобства использования и соответствия их потребностям.

Для автоматизации тестирования используются различные инструменты, такие как Selenium WebDriver, Jest, JUnit и другие. Они помогают ускорить процесс тестирования, повысить его эффективность и снизить вероятность человеческих ошибок.



**Внедрение ПО**

Внедрение программного обеспечения - это процесс развертывания и запуска программного продукта на рабочих системах конечных пользователей. Подготовка к внедрению включает в себя ряд этапов, начиная с планирования и подготовки инфраструктуры, такой как установка необходимых программ и настройка оборудования. Важной частью подготовки является обучение пользователей, чтобы они могли успешно использовать новое программное обеспечение.

Внедрение ПО может быть постепенным, начиная с ограниченного круга пользователей или функциональности, а затем постепенно расширяясь до полного масштаба. Важным шагом в процессе внедрения является тестирование программного обеспечения в реальных условиях перед его полным запуском. Это позволяет выявить проблемы и ошибки, которые могут возникнуть в процессе эксплуатации программы.

После внедрения программного обеспечения необходимо обеспечить его поддержку, включая регулярное обновление и исправление ошибок. Также важно осуществлять мониторинг работы программы и собирать обратную связь от пользователей для постоянного улучшения качества и функциональности продукта.

**Фреймворки**

Фреймворки в веб-разработке представляют собой наборы инструментов, библиотек и шаблонов, которые упрощают создание веб-приложений, обеспечивая структурирование кода, повторное использование компонентов, управление зависимостями и обработку рутинных задач. Они предоставляют разработчикам готовые решения для типовых задач, что позволяет сосредоточиться на более сложных аспектах разработки и повысить производительность.

Как работают фреймворки:

Структура проекта: Фреймворки предоставляют структуру для организации кода в веб-приложении. Они определяют конвенции и правила именования файлов и каталогов, что облегчает навигацию и поддержку проекта.

Компоненты и модули: Фреймворки предоставляют готовые компоненты и модули, которые можно использовать для построения веб-приложений. Это может включать в себя компоненты пользовательского интерфейса, роутинг, аутентификацию, обработку форм и другие функциональные блоки.

Управление зависимостями: Фреймворки предоставляют средства для управления зависимостями и внешними библиотеками, что упрощает установку, обновление и удаление компонентов в проекте.

Шаблоны и макеты: Фреймворки обычно предоставляют готовые шаблоны и макеты для создания внешнего вида веб-приложения. Это позволяет создавать профессионально выглядящие приложения с минимальными усилиями.

Расширяемость: Фреймворки обычно предоставляют API и средства для расширения и настройки функциональности по мере необходимости. Это позволяет разработчикам создавать свои собственные компоненты и модули и интегрировать их в проект.

Зачем нужны фреймворки:

Ускорение разработки: Фреймворки позволяют ускорить разработку веб-приложений, предоставляя готовые решения для типовых задач и упрощая повторное использование кода.

Структурирование проекта: Фреймворки помогают организовать код в проекте, определяя конвенции и стандарты разработки, что улучшает его читаемость и поддерживаемость.

Обеспечение качества: Фреймворки часто предоставляют инструменты для тестирования кода и обеспечения его качества, что позволяет выявлять и исправлять ошибки на ранних этапах разработки.

Упрощение масштабирования: Фреймворки обеспечивают гибкую архитектуру приложения, что упрощает его масштабирование по мере роста бизнеса и объема пользовательского трафика.

Сообщество и поддержка: Фреймворки часто имеют активное сообщество разработчиков, которое предоставляет поддержку, документацию, обучающие материалы и решения проблем, что помогает разработчикам быстро решать трудности и продвигать проект вперед.

Фреймворки являются неотъемлемой частью современной веб-разработки, предоставляя разработчикам инструменты и ресурсы для создания высококачественных и профессионально выглядящих веб-приложений.

**Express.js и EJS**

Express.js - это минималистичный и гибкий веб-фреймворк для Node.js, который позволяет создавать веб-приложения и API с использованием JavaScript. Express.js обеспечивает простой и интуитивно понятный интерфейс для обработки HTTP-запросов, маршрутизации, обработки ошибок, шаблонизации и многое другое.

Основные особенности Express.js:

Маршрутизация: Express.js предоставляет мощные средства для определения маршрутов HTTP-запросов и их обработки с помощью обработчиков маршрутов. Это позволяет легко определять различные пути URL и связывать их с соответствующими действиями.

Шаблонизация: Express.js поддерживает использование различных шаблонизаторов для генерации HTML-страниц на сервере. Один из наиболее популярных шаблонизаторов, который часто используется с Express.js, это EJS (Embedded JavaScript). EJS позволяет встраивать JavaScript код прямо в HTML-шаблоны, что облегчает динамическое формирование контента на сервере.

EJS (Embedded JavaScript) - это шаблонизатор для JavaScript, который позволяет создавать динамические HTML-страницы с помощью встраивания JavaScript кода непосредственно в HTML. Он интегрируется с Express.js и другими фреймворками для Node.js, обеспечивая удобный способ создания и отображения динамического контента на сервере.

Основные особенности EJS:

Простота использования: EJS предоставляет простой и интуитивно понятный синтаксис для встраивания JavaScript кода в HTML-шаблоны. Это позволяет создавать динамические страницы без необходимости изучения новых языков или синтаксисов.

Мощные возможности: EJS обладает мощными возможностями, такими как условные операторы, циклы, включение других шаблонов, передача параметров и многое другое. Это делает его удобным инструментом для создания разнообразного и динамичного контента на сервере.

Интеграция с Express.js: EJS легко интегрируется с Express.js, что делает его популярным выбором для создания веб-приложений на Node.js. Он обеспечивает удобный способ создания шаблонов и отображения данных на сервере, что упрощает разработку и поддержку веб-приложений.

Express.js и EJS представляют собой мощный инструментарий для создания веб-приложений на Node.js. Они обеспечивают простоту использования, гибкость и мощные возможности для разработки динамических и интерактивных веб-приложений.

**React.js**

React.js - это JavaScript библиотека для создания пользовательских интерфейсов. Разработанная компанией Facebook, React позволяет строить эффективные, масштабируемые и многофункциональные веб-приложения, обеспечивая простоту разработки и высокую производительность.

Основные принципы React:

Компонентный подход: React основан на компонентной архитектуре, где пользовательский интерфейс разбит на множество независимых и переиспользуемых компонентов. Каждый компонент представляет собой небольшую часть интерфейса, которая может быть легко создана, изменена и повторно использована в различных частях приложения.

Виртуальный DOM: React использует виртуальное дерево объектов (Virtual DOM), которое является легковесным представлением реального DOM. Это позволяет React эффективно обновлять только те части интерфейса, которые изменились, без необходимости перерисовывать всю страницу. Это повышает производительность приложения и улучшает пользовательский опыт.

Однонаправленный поток данных: В React используется однонаправленный поток данных, что означает, что данные передаются сверху вниз через иерархию компонентов. Это делает код более предсказуемым и упрощает отладку и тестирование приложения.

JSX (JavaScript XML): JSX - это расширение синтаксиса JavaScript, которое позволяет писать HTML-подобный код непосредственно в JavaScript файлы. Это упрощает создание пользовательского интерфейса, делая код более читаемым и понятным.

Преимущества использования React:

Высокая производительность: Благодаря виртуальному DOM и эффективной рендеризации компонентов, React обеспечивает высокую производительность даже при работе с большими объемами данных.

Масштабируемость: Компонентная архитектура React позволяет легко масштабировать приложение, добавляя новые компоненты или изменяя существующие без необходимости переписывания большого объема кода.

Экосистема: React имеет обширную экосистему инструментов и библиотек, таких как React Router, Redux, Material-UI и многие другие, которые расширяют его возможности и облегчают разработку.

Активное сообщество: React имеет большое и активное сообщество разработчиков, которые предоставляют поддержку, документацию, обучающие материалы и решения проблем, что упрощает разработку и поддержку приложений на React.

React является одним из самых популярных инструментов в веб-разработке благодаря своей эффективности, производительности и простоте использования. Он широко применяется для создания интерактивных пользовательских интерфейсов в современных веб-приложениях и сайтах.

**Git**

Git - это распределенная система управления версиями, разработанная Линусом Торвальдсом. Она обеспечивает возможность отслеживания изменений в исходном коде программного обеспечения, контроля версий файлов и совместной работы над проектами.

В основе работы Git лежит концепция хранилища данных, где каждое изменение исходного кода сохраняется как отдельный "снимок" или коммит. Эти коммиты образуют историю изменений проекта, позволяя разработчикам отслеживать историю изменений, возвращаться к предыдущим версиям кода и объединять изменения разных разработчиков.

Одной из ключевых особенностей Git является его распределенная природа. Каждый разработчик имеет полную копию репозитория проекта на своем компьютере, что позволяет им работать независимо друг от друга, не завися от доступности центрального сервера. Это обеспечивает гибкость и надежность при работе над проектами в распределенных командах или в удаленных условиях.

Git также обладает обширным набором инструментов для управления исходным кодом, включая возможности создания веток (branches) для параллельной работы над различными функциями или исправлениями, слияние изменений между ветками, и многое другое. Это делает Git не только мощным инструментом для контроля версий, но и центральным элементом в разработке программного обеспечения в современном мире.



**Заключение**

В ходе исследовательской практики мне удалось применить на практике все свои теоретические знания. Я полностью выполнил задачи, поставленные моим руководителем. Среди этих задач я работал в VisualStudio C#, Figma, Adobe Photoshop. Я представил свою новую идею этому учреждению. В частности, я предложил программы удаленной работы в связи с предстоящей пандемией. За время стажировки я достиг всех своих целей. Студенту во время исследовательской практики было дано:

1) Выполнение задач, поставленных руководителем.

2) Ведение дневника.

3) Представить свою новую идею заведению.

4) Приехать на место практики в назначенное время и приступить к ознакомлению с производством.

5) Развитие и закрепление теоретических знаний в практических ситуациях.

6) Проводить общественные работы во время практики.

7) Формирование должностной квалификации – свои обязанности я выполнил.

**Список используемых источников**

1. Официальный сайт ТОО «NTS Design» [Электронный ресурс]. Режим доступа: https://nts.kz/o-kompanii (дата запроса: 29.01.2024).
2. Оффициальный сайт Visual Studio [Электронный ресурс]. Режим доступа: https://visualstudio.microsoft.com/ru/ (дата запроса: 06.02.2024).
3. Руководство по классическим приложениям [Электронный ресурс]. Режим доступа: https://learn.microsoft.com/ru-ru/dotnet/fundamentals/ (дата запроса: 08.02.2024).
4. Оффициальный сайт Photoshop [Электронный ресурс]. Режим доступа: https://www.adobe.com/ru/products/photoshop.html (дата запроса: 18.02.2024).
5. Классические эффекты Photoshop [Электронный ресурс]. Режим доступа: https://nodejs.org/en (дата запроса: 20.02.2024).