|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

|  |  |
| --- | --- |
| ФАКУЛЬТЕТ | Робототехника и комплексная автоматизация (РК) |
| КАФЕДРА | Системы автоматизированного проектирования (РК6) |

**РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

***К НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ***

***НА ТЕМУ:***

***«Разработка интерактивных web-компонентов на фреймворке Svelte»***

|  |  |  |
| --- | --- | --- |
| Студент РК6-81Б | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | **Гассиев В.Г.** |
|  | (Подпись, дата) | И.О. Фамилия |
| Руководитель | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | **Витюков Ф.А.** |
|  | (Подпись, дата) | И.О. Фамилия |

*2024 г.*

**Министерство науки и высшего образования Российской Федерации**

**Федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«Московский государственный технический университет имени Н.Э. Баумана**

**(национальный исследовательский университет)»**

**(МГТУ им. Н.Э. Баумана)**

УТВЕРЖДАЮ

Заведующий кафедрой РК6

А.П. Карпенко

«\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2024 г.

**ЗАДАНИЕ**

**на выполнение научно-исследовательской работы**

по теме: Разработка интерактивных web-компонентов на фреймворке Svelte\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Студент группы \_\_РК6-81Б\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Гассиев Валерий Германович\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Фамилия, имя, отчество)

Направленность НИР (учебная, исследовательская, практическая, производственная, др.) \_учебная\_\_\_

Источник тематики (кафедра, предприятие, НИР) \_кафедра \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

График выполнения НИР: 25% к 5 нед., 50% к 11 нед., 75% к 14 нед., 100% к 16 нед.

Техническое задание: Разработать интерактивные web компоненты используя фреймворки Svelte и Tailwind. Разработать две интерактивные карусели и оптимизировать их работу. Реализовать компонентный подход в разработке.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

***Оформление научно-исследовательской работы:***

Расчетно-пояснительная записка на листах формата А4.

Перечень графического (иллюстративного) материала (чертежи, плакаты, слайды и т.п.):

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата выдачи задания «20» апреля 2024 г.

|  |  |  |
| --- | --- | --- |
| **Руководитель НИР** | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | **Витюков Ф.А.** |
|  | (Подпись, дата) | И.О. Фамилия |
| **Студент** | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | **Гассиев В.Г.** |
|  | (Подпись, дата) | И.О. Фамилия |

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 4](#_Toc165467897)

[1. Компонентная разработка веб-приложений 5](#_Toc165467898)

[2. Основные понятия и принципы компонентной разработки в Svelte 6](#_Toc165467899)

[3. Проектирование и создание компонентов 9](#_Toc165467900)

[4. Удобство и чистота кода с помощью компонентного подхода 15](#_Toc165467901)

[5. Проблемы во время разработки и их решения 16](#_Toc165467902)

[6. Развитие компонентной архитектуры в будущем 18](#_Toc165467903)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 20](#_Toc165467904)

ВВЕДЕНИЕ

В современном мире разработки программного обеспечения, компонентная архитектура играет ключевую роль в создании гибких, масштабируемых и эффективных приложений. Она представляет собой подход к разработке, основанный на декомпозиции больших систем на более мелкие и управляемые компоненты, которые могут быть легко созданы, поддержаны и переиспользованы.

Компонентная архитектура привносит в разработку множество преимуществ, включая повышение производительности, улучшение поддерживаемости кода, сокращение времени разработки и облегчение совместной работы в больших командах. Она позволяет разработчикам сосредоточиться на отдельных частях приложения, разрабатывая их независимо и интегрируя в общую систему.

В данном работе рассмотрим различные аспекты компонентной архитектуры, начиная с основных принципов и преимуществ, и заканчивая перспективами её развития в будущем. Также проанализируем реальный пример использования компонентов в проекте, и рассмотрим способы оптимизации производительности и решения проблем, возникающих в процессе разработки. Погружаясь в эту тему, сможем лучше понять роль компонентной архитектуры в современной разработке и её влияние на создание инновационных и устойчивых программных продуктов.

1. Компонентная разработка веб-приложений

Компонентная архитектура представляет собой подход к разработке программного обеспечения, в котором приложение разбивается на небольшие и независимые компоненты. Каждый компонент представляет собой модуль, который обладает собственным набором функций и может быть многократно использован в различных частях приложения. Этот подход позволяет повысить читаемость, поддерживаемость и масштабируемость кода.

**Основные принципы компонентной архитектуры:**

1. Разделение на компоненты: Приложение разбивается на небольшие компоненты, каждый из которых отвечает за конкретную функциональность или отображение данных.
2. Независимость компонентов: Компоненты должны быть максимально независимыми друг от друга. Это позволяет легко вносить изменения в один компонент без влияния на другие.
3. Повторное использование: Компоненты должны быть разработаны таким образом, чтобы их можно было многократно использовать в различных частях приложения или даже в других проектах.
4. Единственная ответственность (Single Responsibility Principle): Каждый компонент должен отвечать только за одну функциональность или часть интерфейса.

**Преимущества компонентной разработки:**

1. Модульность: Легко понять и изменить отдельные компоненты без влияния на другие части приложения.
2. Повторное использование кода: Компоненты могут быть многократно использованы в различных частях приложения, что уменьшает объем дублирующегося кода.
3. Улучшенная поддержка: Благодаря независимости компонентов, поддержка и разработка новых функций становится более эффективной и быстрой.
4. Скорость разработки: Компоненты позволяют разрабатывать приложения модульно, что ускоряет процесс разработки и облегчает масштабирование проекта.
5. Тестирование: Изоляция компонентов облегчает их тестирование, поскольку каждый компонент может быть протестирован независимо от других.
6. Основные понятия и принципы компонентной разработки в Svelte

**Структура компонента**

В Svelte компоненты представляют собой файлы с расширением .svelte, которые объединяют в себе HTML, CSS и JavaScript в одном файле. Например, простой компонент кнопки в Svelte может выглядеть следующим образом:

*<!-- Button.svelte -->*

**<script>**

**let** buttonText = "Click me";

**function** handleClick() {

alert("Button clicked!");

}

**</script>**

**<button** on:click={handleClick}**>**{buttonText}**</button>**

**<style>**

**button** {

**background-color**: #007bff;

**color**: #fff;

**border**: **none**;

**padding**: 10px 20px;

**border**-radius: 5px;

**cursor**: **pointer**;

}

**</style>**

Листинг 1. Компонент кнопки в Svelte

**Связывание данных (Data Binding)**

Svelte предоставляет удобные механизмы для связывания данных между JavaScript и HTML в компонентах. Вы можете объявлять переменные в блоке <script> и использовать их в разметке:

**<script>**

**let** name = 'World';

**</script>**

**<h1>**Hello, {name}!**</h1>**

Листинг 2. Связывания данных между JavaScript и HTML в компонентах

При изменении переменной name автоматически обновится и содержимое элемента <h1>.

**Директивы**

Директивы - это специальные атрибуты, которые добавляют дополнительное поведение элементам DOM в Svelte. Например, директива on:click используется для добавления обработчика событий клика к элементу:

**<button** on:click={handleClick}**>**Click me**</button>**

Листинг 3. Директива on:click

**Слоты (Slots)**

Слоты в Svelte позволяют передавать контент из родительского компонента в дочерний компонент. Это удобно для создания компонентов с динамическим содержимым. Например, компонент Button может иметь слот для текста:

*<!-- Button.svelte -->*

**<button>**{#slot}**</button>**

*<!-- ParentComponent.svelte -->*

**<Button>**

Click me

**</Button>**

Листинг 4. Слот для текста

**Локальное состояние компонентов**

Каждый компонент в Svelte может иметь свое локальное состояние, которое не доступно извне. Для объявления локального состояния используется ключевое слово let:

**<script>**

**let** count = 0;

**function** increment() {

count += 1;

}

**</script>**

**<button** on:click={increment}**>**

Count: {count}

**</button>**

Листинг 5. Объявления локального состояния

**Жизненный цикл компонентов**

В Svelte есть специальные функции, которые могут быть вызваны на различных этапах жизненного цикла компонента. Например, функция onMount вызывается после того, как компонент был добавлен в DOM:

**<script>**

**import** { onMount } from 'svelte';

onMount(() => {

console.log('Component mounted');

});

**</script>**

Листинг 6. Функция onMount

**Условные операторы и циклы**

Svelte поддерживает условные операторы и циклы прямо в разметке компонентов. Например, вы можете использовать директиву if для условного рендеринга:

{#if loggedIn}

**<p>**Welcome, {username}!**</p>**

{:else}

**<p>**Please log in.**</p>**

{/if}

Листинг 7. Условные операторы и циклы

1. Проектирование и создание компонентов

В данном разделе представлено проектирование и разработка компонентов, необходимых для реализации интерактивной веб-страницы с каруселью игровых проектов. Для обеспечения функциональности и эстетичного визуального представления проекта были созданы следующие компоненты:

1. **Шапка (Header):**

* Компонент отображает верхнюю часть страницы с заголовком и дополнительным контентом.
* Обеспечивает единый стиль и навигацию по странице.

**<header** title="Esc" class="text-white"**>**

**<nav** class="flex items-center justify-between px-4"**>**

**<a** href="/escapists" class="my-5 8k:my-[70px] 8k:ml-[70px] bg-blue-500 hover:bg-blue-600 font-bold rounded-full inline-block text-sm py-2 px-4 sm:text-base sm:py-3 sm:px-6 md:text-lg md:py-2 md:px-3 lg:text-[12px] lg:py-1 lg:px-2 2k:text-xl 2k:py-3 2k:px-5 4k:text-3xl 4k:py-6 4k:px-12 8k:text-6xl 8k:py-[50px] 8k:px-[110px]"**>**

Часть 2

**</a>**

**</nav>**

**<slot></slot>**

**</header>**

Листинг 8. header.svelte

1. **Изображения карусели (imageSection):**

* Этот компонент отвечает за отображение изображений и видео в карусели.
* Предоставляет пользователю визуальный контент и информацию о каждом проекте.

**<script>**

**export** **let** mainImageIndex;

**export** **let** carouselItems;

**function** getYouTubeEmbedUrl(videoId) {

**return** `https:*//www.youtube.com/embed/${videoId}?autoplay=1&enablejsapi=1&controls=0&showinfo=0&autohide=1&mute=1&modestbranding=1&loop=1&rel=0&key=${apiKey}`;*

}

**function** handleVideoTimeUpdate(event) {

currentTime = event.detail.currentTime;

}

**function** startTimer() {

*// Устанавливаем таймер на 1 минуту и 55 секунд*

timer = setTimeout(nextSlide, 115 \* 1000);

}

**const** apiKey = 'AIzaSyCf72M2WmMeWnrJ0ADzfZv86Ap7EI3ASNU';

**</script>**

**<style>**

**@import** '../components/css/ImageCarousel.styles.css';

**</style>**

**<div** class="w-full h-auto"**>**

{#if carouselItems && carouselItems.length > 0}

{#if mainImageIndex === carouselItems.length - 1 && carouselItems[mainImageIndex].video}

**<div** class="video-container w-full" style="position: relative;"**>**

**<iframe**

title="video"

class="w-full h-full"

style="video-container iframe"

src={getYouTubeEmbedUrl(carouselItems[mainImageIndex].video)}

allow="autoplay; encrypted-media"

allowfullscreen

on:timeupdate={handleVideoTimeUpdate}

on:playing={startTimer}

**></iframe>**

**</div>**

{:else}

*<!-- Display image for other slides -->*

**<img** class="main-image w-full object-fill" alt="" src={carouselItems[mainImageIndex].image} **/>**

{/if}

{/if}

**</div>**

Листинг 9. imageSection.svelte

1. **Текст и логотипы (TextAndLogoSection):**

* Компонент отображает текстовое содержание и логотипы, соответствующие выбранному слайду карусели.
* Предоставляет пользователю дополнительную информацию о каждом проекте.

**<script>**

**export** **let** mainImageIndex;

**export** **let** carouselItems;

**function** shouldCenterText(index) {

**return** index === 1 || index === 3;

}

**</script>**

**<div** class="absolute top-0 left-0 w-full h-full"**>**

{#each carouselItems as { text, position, logo, platforms }, index (text)}

{#if index === mainImageIndex}

**<div** style={`top: ${position.top}; left: ${position.left};`} class=" text-white 8k:text-[250px] 4k:text-[140px] 2k:text-[64px] xl:text-5xl xl:text-[43px] lg:text-[33.68px] md:text-[26px] sm:text-xl xsm:text-[9px] font-extrabold 8k:leading-[270px] 4k:leading-[147px] 2k:leading-[77px] lg:leading-[39px] leading-tight absolute ${position.classPos}"**>**

{#if shouldCenterText(index)}

**<div** class="text-center"**>**

{@html text.split('**<br/>**').join('**<br** **/>**')}

**</div>**

{:else}

{@html text.split('**<br/>**').join('**<br** **/>**')}

{/if}

**</div>**

{#if logo}

**<div** style={`top: ${logo.position.top}; left: ${logo.position.left};`} class="absolute"**>**

**<img** src={logo.image} alt="Logo" class={logo.sizeClasses} **/>**

**</div>**

{/if}

{/if}

{/each}

**</div>**

Листинг 10. textAndLogo.svelte

1. **Кнопки навигации (NavigationButtons):**

* Этот компонент обеспечивает навигацию по слайдам карусели.
* Позволяет пользователям перемещаться между проектами в карусели.

**<script>**

**export** **let** goToSlide;

**export** **let** carouselItems;

**export** **let** mainImageIndex;

**export** **let** hoveredIndex;

**function** handleMouseEnter(index) {

hoveredIndex = index;

}

**function** handleMouseLeave() {

hoveredIndex = **null**;

}

**</script>**

**<div** class="absolute bottom-0 left-0 w-full h-10 flex justify-center items-center 4k:mb-[20px]"**>**

{#each carouselItems as { image }, index (image)}

**<div**

role="button"

tabindex="0"

class="{ 'w-[50px] h-[9px] 8k:w-[350px] 8k:h-[50px] 8k:mx-[20px] 4k:w-[140px] 4k:h-[25px] 4k:mx-[10px] 2k:w-[60px] 2k:h-[10px] xl:w-[50px] xl:h-[9px] sm:w-[40px] sm:h-[7px] xsm:w-[30px] xsm:h-[8px] 8k:mb-20 mx-1 sm:mb-[-3%] lg:mb-1 transition duration-300 ease-in-out cursor-pointer rounded-sm ' + (index === mainImageIndex ? 'bg-white' : (hoveredIndex === index ? 'bg-white' : 'bg-zinc-400')) }"

on:click={() =**>** goToSlide(index)}

on:keydown={e => e.key === 'Enter' && goToSlide(index)}

on:mouseenter={() => handleMouseEnter(index)}

on:mouseleave={() => handleMouseLeave(index)}

>**</div>**

{/each}

**</div>**

Листинг 11. navigationButtons.svelte

1. **Платформы (Platforms):**

* Компонент отображает доступные игровые платформы для выбранного проекта.
* Предоставляет пользователю информацию о поддерживаемых платформах для каждого проекта.

**<script>**

**export** **let** mainImageIndex;

**export** **let** carouselItems;

**export** **let** goToSlide;

**export** **let** resetVisiblePlatforms;

**export** **let** visiblePlatforms;

**export** **let** showAdditionalPlatforms;

**</script>**

**<div** class="absolute bottom-0 left-0 w-full h-[67px] p-2.5 opacity-95 rounded-[20px] flex items-center justify-center whitespace-nowrap 4k:pr-[200px] 4k:mb-[65px] 2k:mb-[40px] xl:mb-[30px] lg:mb-[18px] md:mb-[20px] sm:mb-[14px]"**>**

{#if mainImageIndex !== carouselItems.length - 1 || !carouselItems[mainImageIndex].video}

**<div** class="text-white sm:mb-1 md:mb-1 lg:mb-5 2k:mb-2 sm:mr-[2%] md:mr-[1%] lg:mr-[0%] 2k:mr-[-0.2%] 4k:mr-[-1.2%] 8k:mr-[-2.1%] 8k:mb-[530px] 4k:mb-[150px] xl:mb-[20px] text-[28px] 8k:text-[160px] 4k:text-[75px] 2k:text-[32px] xl:text-[24px] lg:text-xl md:text-xl sm:text-[17px] font-normal font-['Inter'] leading-[23px] whitespace-nowrap 4k:pl-[100px] xl:mt-[10px] xsm:hidden sm:block md:block lg:block xl:block 2k:block 4k:block 8k:block "**>**

Buy it for

**</div>**

{#if visiblePlatforms.length > 0}

{#each visiblePlatforms as platform (platform)}

*<!-- secondary icons -->*

**<div** class="w-11 h-[47px] flex-col justify-center items-center gap-2.5 inline-flex group ml-3 8k:ml-[290px] 8k:mb-[490px] 4k:ml-[110px] 4k:mb-[130px] sm:mb-[-3.2%] group sm:ml-[-1%] sm:mr-[-1%] md:ml-[1%] md:mr-[-1.3%] xl:ml-[1%] xl:mr-[-0.7%] 2k:ml-[1%] 2k:mr-[-0.2%] 4k:mr-[-0.2%] 8k:mr-[-0.6%] lg:mb-0 2k:mb-[-0.5%] "**>**

**<div** class="w-[47px] h-[47px] 8k:w-[230px] 8k:h-[230px] 4k:w-[110px] 4k:h-[110px] 2k:w-[47px] 2k:h-[47px] xl:w-[40px] xl:h-[40px] md:w-[30px] md:h-[30px] sm:w-[25px] sm:h-[25px] xl:mb-[10px] md:mb-[20px] sm:mb-[25px] relative group flex flex-col items-center"**>**

**<img** class="w-full h-full object-fill" src={`src/img/Platforms/${platform}.svg`} alt={platform} **/>**

**<div** class="text-center text-white text-opacity-0 8k:text-[75px] 8k:mt-[0] 4k:text-[32px] 4k:mt-0 2k:text-[14px] xl:text-[12px] md:text-[12px] sm:text-[12px] font-light font-['Inter'] leading-[23px] group-hover:text-opacity-100 transition-opacity duration-300 ease-in-out"**>**

{platform}

**</div>**

**</div>**

**</div>**

{/each}

**<div**

role="button"

tabindex="0"

class="w-11 h-[47px] flex-col justify-start items-center gap-2.5 inline-flex group ml-3 8k:ml-[290px] 4k:ml-[110px] sm:mb-[-3.3%] sm:ml-[-1.3%] md:ml-[0.2%] lg:mb-[-0.2%] lg:ml-[0.5%] xl:mb-[-0.2] xl:ml-[0.5%] 2k:mb-[-0.2%] 4k:mb-[5.5%] 8k:mb-[9%]"

on:click={resetVisiblePlatforms}

on:keydown={e =**>** e.key === 'Enter' && goToSlide()}

>

**<div** class="w-[47px] h-[47px] 8k:w-[230px] 8k:h-[230px] 4k:w-[110px] 4k:h-[110px] 2k:w-[47px] 2k:h-[47px] xl:w-[40px] xl:h-[40px] md:w-[30px] md:h-[30px] sm:w-[25px] sm:h-[25px] relative group "**>**

**<img** class="w-full h-full" src={`src/img/Arrows/Back.svg`} alt="Arrow Back" **/>**

**<div** class="text-center text-white text-opacity-0 text-xs 8k:text-[75px] 8k:mt-[50px] 4k:text-[32px] 4k:mt-5 2k:text-[14px] xl:text-[12px] md:text-[12px] font-light font-['Inter'] leading-[23px] group-hover:text-opacity-100 transition-opacity duration-300 ease-in-out"**>**Back**</div>**

**</div>**

**</div>**

{:else}

{#each carouselItems[mainImageIndex].platforms as platform (platform)}

*<!-- main icons -->*

**<div**

role="button"

tabindex="0"

class="w-11 h-[47px] text-center flex-col justify-start items-center inline-flex sm:mb-1 lg:mb-5 group sm:ml-[-1%] sm:mr-[-1%] md:ml-[1%] md:mr-[-1.3%] xl:ml-[1%] xl:mr-[-0.7%] 2k:ml-[1%] 2k:mr-[-0.2%] 2k:mb-4 4k:mr-[-0.2%] 8k:mr-[-0.6%] 8k:ml-[290px] 8k:mb-[720px] 4k:ml-[100px] 4k:mb-[220px] sm:mt-[18px] xsm:hidden sm:block md:block lg:block xl:block 2k:block 4k:block 8k:block "

on:click={() =**>** showAdditionalPlatforms(mainImageIndex, platform.main)}

on:keydown={e => e.key === 'Enter' && goToSlide()}

>

**<div** class="w-[47px] h-[47px] 8k:w-[230px] 8k:h-[230px] 4k:w-[110px] 4k:h-[110px] 2k:w-[47px] 2k:h-[47px] xl:w-[40px] xl:h-[40px] md:w-[30px] md:h-[30px] sm:w-[25px] sm:h-[25px] relative group flex flex-col items-center"**>**

**<img** class="w-full h-full " src={`src/img/Platforms/${platform.main}.svg`} alt={platform.main} **/>**

**<div** class="text-center text-white text-opacity-0 text-xs 8k:text-[75px] 8k:mt-[50px] 4k:text-[32px] 4k:mt-5 2k:text-[14px] xl:text-[12px] md:text-[12px] font-light font-['Inter'] leading-[23px] group-hover:text-opacity-100 transition-opacity duration-300 ease-in-out "**>**

{platform.main}

**</div>**

**</div>**

**</div>**

{/each}

{/if}

{/if}

**</div>**

Листинг 12. platforms.svelte

1. Удобство и чистота кода с помощью компонентного подхода

Использование вышеупомянутых компонентов в главном файле значительно улучшает читаемость, структурированность и удобство кода проекта. Вместо того чтобы плодить длинные и запутанные блоки кода, лучшее решение было разделить функционал на отдельные компоненты, каждый из которых отвечает за определенную часть интерфейса. Вот как это преимущество проявляется в коде:

1. **Модульность и структурированность:**

* Каждый компонент отвечает только за свою специфическую задачу, что позволяет легко понять его назначение и внести изменения при необходимости.
* Разбиение интерфейса на компоненты упрощает работу с кодом и облегчает его поддержку.

1. **Повторное использование:**

* Компоненты созданы с учетом возможности многократного использования в различных частях проекта.
* Это позволяет избежать дублирования кода и сделать проект более модульным и гибким.

Таким образом, благодаря разделению интерфейса на небольшие и самодостаточные компоненты, мы создаем чистый, понятный и удобный код, который легко поддерживать и масштабировать. Кроме того, возможность многократного использования компонентов делает наш проект более эффективным и экономит время разработки.

**<main** class="p-0 relative flex flex-col items-center"**>**

**<CarouselImage** mainImageIndex={mainImageIndex} carouselItems={carouselItems}/**>**

**<TextAndLogoSection** mainImageIndex={mainImageIndex} carouselItems={carouselItems} **/>**

**<CarouselControls** prevImage={prevImage} nextImage={nextImage} **/>**

**<Platforms**

mainImageIndex={mainImageIndex}

carouselItems={carouselItems}

goToSlide={goToSlide}

resetVisiblePlatforms={resetVisiblePlatforms}

visiblePlatforms={visiblePlatforms}

showAdditionalPlatforms={showAdditionalPlatforms}

**/>**

**<NavigationButtons**

goToSlide={goToSlide}

prevImage={prevImage}

nextImage={nextImage}

carouselItems={carouselItems}

mainImageIndex={mainImageIndex}

**/>**

**</main>**

Листинг 13. Пример главного файла проекта, где используются все компоненты

1. Проблемы во время разработки и их решения

В ходе разработки проекта столкнулись с рядом проблем, которые затрудняли процесс и требовали тщательного анализа и поиска решений. Вот некоторые из них:

1. **Отображение изображений и иконок:**

* Одной из проблем, с которой столкнулись, было неправильное отображение изображений и иконок на странице.
* Изображения не загружались или отображались с задержкой, что приводило к негативному пользовательскому опыту.

1. **Загрузка ресурсов:**

* Еще одной проблемой была медленная загрузка ресурсов, таких как изображения, скрипты и стили.
* Это приводило к длительному времени загрузки страницы и ухудшению производительности.

1. **Оптимизация производительности:**

* Важной проблемой была необходимость оптимизации производительности приложения для достижения быстрой загрузки и отзывчивости интерфейса.
* Без оптимальной производительности пользователи могли столкнуться с задержками и зависаниями при использовании приложения.

1. **Решение проблем:**

* Для решения проблем с отображением изображений и иконок использовали предварительную загрузку ресурсов с помощью тега

<link rel="prefetch">.

* Это позволило предварительно загрузить изображения и иконки, ускоряя их загрузку и обеспечивая их доступность на странице без задержек.
* Кроме того, был проведен анализ и оптимизация загрузки других ресурсов, таких как скрипты и стили, чтобы сократить время загрузки страницы и повысить производительность приложения.

1. **Уроки и выводы:**

* Эти проблемы и их решения подчеркивают важность тщательного анализа процесса разработки и постоянного поиска способов улучшения производительности и пользовательского опыта.
* Решения, принятые для устранения проблем, помогли не только решить текущие проблемы, но и повысить качество и эффективность нашего проекта в целом.

1. Развитие компонентной архитектуры в будущем

Компонентная архитектура уже играет ключевую роль в современной веб-разработке, обеспечивая модульность, повторное использование кода и упрощение поддержки приложений. Однако в будущем мы ожидаем ее дальнейшего развития и усовершенствования, влияющих на процесс создания веб-приложений. Некоторые из направлений развития компонентной архитектуры включают в себя:

**Рост экосистемы компонентов**: Ожидается, что экосистема компонентов будет продолжать расширяться, предоставляя разработчикам все больше готовых компонентов для использования в своих проектах. Это позволит сократить время разработки и повысить эффективность создания интерфейсов.

Улучшение инструментов разработки: В будущем ожидается появление новых инструментов и фреймворков, специально разработанных для создания и управления компонентами. Эти инструменты будут обладать расширенными возможностями автоматизации, проверки и тестирования компонентов.

Дальнейшая оптимизация производительности: Развитие компонентной архитектуры будет сопровождаться улучшениями в области оптимизации производительности компонентов. Это включает в себя разработку более эффективных алгоритмов рендеринга, улучшение управления состоянием компонентов и оптимизацию передачи данных между компонентами.

Рост в области дизайн-систем: Ожидается, что компонентная архитектура будет тесно интегрироваться с развитием дизайн-систем, предоставляя стандартизированные компоненты и решения для создания согласованных и качественных пользовательских интерфейсов.

Развитие Web Components: Стандарт Web Components будет продолжать развиваться и расширяться, обеспечивая более широкую поддержку браузерами и стандартизированный подход к созданию компонентов на веб-платформе.

Рост применения в интернете вещей: c расширением интернета вещей (IoT), компонентная архитектура будет использоваться для создания компонентов, взаимодействующих с умными устройствами и сенсорами. Это позволит разработчикам создавать более умные и автоматизированные приложения для дома, здоровья, транспорта и других областей.

Развитие облачных технологий: облачные технологии будут играть все более важную роль в разработке и развертывании компонентов. Благодаря облачным платформам и сервисам, разработчики смогут создавать, хранить и масштабировать компоненты более эффективно и гибко.

Стремление к единому языку дизайна: будущее компонентной архитектуры также будет связано с развитием единого языка дизайна, который позволит создавать консистентные и интуитивно понятные пользовательские интерфейсы на основе стандартных компонентов и дизайн-систем.

Обеспечение безопасности и приватности: с увеличением угроз в области кибербезопасности и регулированием защиты данных, будущее компонентной архитектуры будет также ориентировано на обеспечение высокого уровня безопасности и защиты приватности пользовательских данных в компонентах и приложениях.

В целом, развитие компонентной архитектуры будет непрерывным процессом, включающим в себя инновации в области технологий, дизайна и методологий разработки. Эти тенденции отражают стремление к созданию более гибких, умных и безопасных веб-приложений, способных эффективно решать разнообразные задачи и ожидания пользователей.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Tailwind CSS Documentation // Tailwind CSS Documentation URL: <https://tailwindcss.com/docs>. Дата обращения: [15.10.2023].;
2. Svelte Documentation // Svelte Documentation URL: <https://svelte.dev/docs>. Дата обращения: [16.10.2023];
3. HTML Living Standard // HTML Living Standard URL: <https://html.spec.whatwg.org/multipage/>. Дата обращения: [16.07.2023].
4. Stack Overflow // Stack Overflow URL: <https://stackoverflow.com/>. Дата обращения: [17.10.2023]