|  |  |
| --- | --- |
| 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:***

***«-»***

|  |  |  |
| --- | --- | --- |
| Студент РК6-74Б | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | **Гассиев В.Г.** |
|  | (Подпись, дата) | И.О. Фамилия |
| Руководитель | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | **Витюков Ф.А.** |
|  | (Подпись, дата) | И.О. Фамилия |

*2023 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

«\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2022 г.

**ЗАДАНИЕ**

**на выполнение научно-исследовательской работы**

по теме: Методы разработки интерактивных элементов пользовательского интерфейса в трехмерном движке Unreal Engine 4\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Студент группы \_\_РК6-71Б\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_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:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

Расчетно-пояснительная записка на 14 листах формата А4.

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

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

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

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

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

Дата выдачи задания «8» февраля 2023 г.

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

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 4](#_Toc152940969)

[1. Обзор средств генерации кода из макета в Figma 5](#_Toc152940970)

[2. Разработка проекта на Svelte и Tailwind 7](#_Toc152940971)

[3. Оптимизация карусели 8](#_Toc152940972)

[4. Демонстрация реализованной карусели 9](#_Toc152940973)

[5. Создание адаптивного интерфейса 13](#_Toc152940974)

[ЗАКЛЮЧЕНИЕ 14](#_Toc152940978)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 16](#_Toc152940979)

ВВЕДЕНИЕ

Современная веб-разработка сталкивается с постоянными вызовами, когда речь идет о балансе между дизайном и технической реализацией. В этом контексте использование интегрированных инструментов дизайна, таких как Figma, предоставляет удивительные возможности для визуализации и концептуализации веб-приложений. Однако, перевод этих концепций в эффективный и оптимизированный код остается вызовом. С ростом популярности Tailwind CSS и фреймворка Svelte, вопрос о взаимодействии этих технологий с инструментами дизайна становится ключевым моментом в современной веб-разработке.

Эффективное взаимодействие с инструментами дизайна становится неотложной задачей, так как оно прямо влияет на конечный результат работы - создание веб-приложений с высокой производительностью и легкостью поддержки. Оптимизированный процесс генерации кода из макета в Figma с использованием Tailwind становится ключевым фактором для сохранения дизайнерских концепций в разработанном приложении. Поднятие этого вопроса на новый уровень необходимо для создания более эффективных рабочих процессов и улучшения конечного качества продукта.

Целью настоящего исследования является оценка средств генерации кода из макета в Figma с использованием Tailwind, а также анализ оптимизации карусели в проекте, построенном на фреймворке Svelte. Для достижения этой цели были выделены следующие задачи:

1. Оценить средства генерации кода из макета в Figma с использованием Tailwind.
2. Изучить существующие инструменты для генерации кода из Figma.
3. Проанализировать преимущества и ограничения применения Tailwind.
4. Исследовать вопросы оптимизации карусели в проекте на Svelte.
5. Проанализировать текущую реализацию карусели и выявить проблемы производительности, предложить и реализовать оптимизации для улучшения работы карусели.
6. Обзор средств генерации кода из макета в Figma

Существующие инструменты для генерации кода из макетов, особенно из таких популярных инструментов, как Figma, играют важную роль в современном процессе веб-разработки. Интеграция дизайн-концепций в кодовую базу проекта стала ключевым этапом в создании современных и эстетически приятных веб-приложений. В данном обзоре мы рассмотрим несколько существующих инструментов и подходов к генерации кода из макетов, с акцентом на применение технологии Tailwind CSS.

**Существующие инструменты**

На сегодняшний день рынок предлагает разнообразные инструменты для генерации кода из макетов в Figma, каждый из которых имеет свои особенности и преимущества. Среди них выделяются такие решения, как Zeplin, Framer, Avocode и др. Каждый из них предоставляет возможности для автоматизации процесса создания кода, позволяя разработчикам более эффективно переносить дизайн-решения в код.

**Оценка преимуществ и ограничений**

**Преимущества*:***

**Ускорение разработки:** Использование инструментов генерации кода позволяет значительно ускорить процесс разработки, минимизируя временные затраты на перенос дизайн-концепций.

**Сохранение структуры дизайна:** Интеграция с Figma сохраняет структуру дизайн-макета, обеспечивая более точное соответствие между дизайном и кодом.

**Снижение вероятности ошибок:** Автоматизированный процесс сокращает ручной ввод и, как следствие, уменьшает вероятность возникновения ошибок в переводе дизайн-концепций в код.

**Недостатки:**

**Ограниченная поддержка некоторых технологий:** Некоторые инструменты могут оказаться менее гибкими в поддержке некоторых технологий, включая Tailwind, что может потребовать дополнительных настроек и компромиссов.

**Необходимость дополнительной оптимизации:** Сгенерированный код может требовать дополнительной оптимизации, особенно при использовании специфичных библиотек и фреймворков, таких как Svelte и Tailwind.

1. Разработка проекта на Svelte и Tailwind

Данный раздел посвящен подробному рассмотрению проекта, основанного на технологиях Svelte и Tailwind CSS. Проект представляет собой интересное сочетание компонентов, изображений и текстов, а также демонстрирует активное взаимодействие пользователя. Давайте подробнее рассмотрим использованные технологии и особенности их интеграции.

**Используемые технологии**

**Технология Svelte**:

Проект активно использует преимущества фреймворка Svelte, который предлагает удобное управление компонентами, обработку событий и эффективное хранение состояний. Применение директив Svelte, таких как each и if, обеспечивает динамичное и эффективное построение пользовательского интерфейса.

**Технология Tailwind CSS**:

Интеграция Tailwind CSS видна в использовании классов для стилизации элементов. Этот инструмент предоставляет широкий спектр классов, упрощая процесс стилизации и обеспечивая гибкость в создании дизайна.

**Особенности интеграции и преимущества выбранных инструментов**

**Интеграция Svelte и Tailwind:**

Проект успешно комбинирует функциональности Svelte и удобства Tailwind CSS. Использование директив Svelte и классов Tailwind позволяет легко управлять компонентами и стилями.

**Преимущества использования Svelte:**

**Компактность кода:** Svelte позволяет создавать чистый и лаконичный код благодаря своей концепции компиляции компонентов.

**Эффективность:** Благодаря компиляции на этапе сборки, Svelte обеспечивает высокую производительность и быстрый отклик интерфейса.

**Преимущества использования Tailwind CSS:**

**Простота использования:** Tailwind предоставляет простой и интуитивно понятный способ добавления стилей с использованием классов.

**Гибкость стилизации:** Возможность комбинировать классы Tailwind обеспечивает гибкость в создании разнообразных стилей.

**Особенности проекта:**

Проект демонстрирует высокий уровень интерактивности, предоставляя пользователям возможность переключаться между изображениями, анимированными переходами и отображением доступных платформ для каждой игры.

Структура кода проекта является логичной и понятной, что способствует легкости его понимания и поддержки. Функции активно используются для обработки событий и управления состояниями, что содействует модульности и переиспользованию кода.

Стилизация проекта соблюдает принципы Tailwind, что придает интерфейсу современный и чистый внешний вид. Использование классов Tailwind делает стилизацию простой и эффективной.

1. Оптимизация карусели

Карусель, как ключевой элемент веб-приложения, может сталкиваться с рядом проблем, влияющих на производительность. Некоторые из вызовов, специфичных для Svelte, могут включать:

**Перерисовка компонентов:** Обновление состояния карусели может привести к перерисовке всех компонентов, даже если они не видны пользователю. Это может привести к излишней нагрузке на ресурсы.

**Управление памятью:** Неконтролируемое увеличение числа элементов в карусели может привести к проблемам с управлением памятью, особенно при работе с большими изображениями.

**Отрисовка большого количества элементов:** Карусель может содержать множество элементов, что может снизить производительность из-за необходимости постоянного пересчета и перерисовки.

**Предложение и реализация оптимизаций для улучшения производительности:**

**Ленивая загрузка изображений**: вместо загрузки всех изображений сразу, внедрите механизм ленивой загрузки, который будет подгружать изображения только при их фактическом появлении в области видимости. Это снизит нагрузку на сеть и ускорит начальную загрузку страницы.

**Кэширование компонентов:** внедрение механизма кэширования компонентов карусели, позволяет избежать лишних перерисовок в случаях, когда состояние компонента не изменяется.

**После внедрения предложенных оптимизаций ожидается следующее:**

**Улучшенная производительность**: Загрузка страницы и взаимодействие с каруселью станут более отзывчивыми благодаря ленивой загрузке.

**Экономия ресурсов**: Кеширование данных и компонентов поможет снизить объем передаваемых данных и использование памяти, что особенно важно при работе с большим количеством элементов.

**Повышенная стабильность**: Оптимизация кеширования данных повысит стабильность работы карусели, снижая возможные ошибки и зависания.

**Более эффективное использование ресурсов клиента**: Пользователи с более слабыми устройствами или медленным интернет-соединением будут иметь лучший опыт использования вашего веб-приложения.

1. Демонстрация реализованной карусели

**Интерактивность:**

Пользователи могут переключаться между изображениями, используя кнопки вправо и влево, а также путем выбора конкретного изображения в навигационной полосе внизу(Рисунок 1).



Рисунок 1

**Информация о каждой игре:**

Для каждой игры предоставлена дополнительная информация, включая описание, логотип и информацию о платформах, на которых она доступна (Рисунок 2).



Рисунок 2 – Демонстрация раскрывающегося меню дополнительных платформ

**Пример Реализации второй карусели:**

Приведенный ниже код представляет собой реализацию второй карусели с использованием фреймворка Svelte. Эта карусель предназначена для просмотра изображений, предоставляя пользователю возможность выбора конкретного изображения из миниатюр. Давайте рассмотрим основные элементы этой реализации:

**<script>**

*// Состояние компонента*

**let** currentIndex = 0;

**let** images = [

'src/img/Escapists 2 0.png',

'src/img/Escapists 2 1.png',

'src/img/Escapists 2 2.png',

'src/img/Escapists 2 3.png',

'src/img/Escapists 2 4.png',

'src/img/Escapists 2 5.png',

'src/img/Escapists 2 6.png',

'src/img/Escapists 2 7.png',

];

*// Обработчики событий*

**function** updateIndex(index) {

currentIndex = index;

}

**function** handleThumbnailClick(i) {

currentIndex = i;

}

**function** handleKeyPress(event, i) {

**if** (event.key === 'Enter' || event.key === ' ') {

currentIndex = i;

}

}

**</script>**

**<main** class="p-0"**>**

*<!-- Внешний контейнер -->*

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

**<div** class="bg-353C55 h-12"**></div>** *<!-- Хедер -->*

*<!-- Основное изображение -->*

**<img** src={images[currentIndex]} alt={`Image ${currentIndex + 1}`} class="w-full h-full" **/>**

*<!-- Миниатюры и навигация -->*

**<div** class="flex justify-center absolute bottom-0 w-full"**>**

{#each images as image, i (image)}

*<!-- Кнопки миниатюр -->*

**<button**

class={` xsm:w-[38px] xsm:h-[24px] sm:w-[58px] sm:h-[40px] md:w-[68px] md:h-[40px] lg:w-[90px] lg:h-[53px] xl:w-[130px] xl:h-[86px] mb-3 mx-1 xsm:border-[3px] sm:border-[3px] md:border-[3px] lg:border-[5px] xl:border-[11px]${

i === currentIndex ? ' border-blue-500' : ' border-gray-700'

}`}

on:click={() =**>** handleThumbnailClick(i)}

on:keypress={(event) => handleKeyPress(event, i)}

>

*<!-- Контейнер миниатюры -->*

**<div** class="w-full h-full bg-cover bg-center relative"**>**

**<img** src={image} alt={`Thumbnail ${i + 1}`} class="w-full h-full" **/>**

**</div>**

**</button>**

{/each}

**</div>**

**</div>**

**</main>**

Листинг 1. Реализация второй карусели

**Основные особенности:**

* Миниатюры представлены в виде кнопок, каждая из которых представляет собой изображение.
* Активная миниатюра выделена синей рамкой.
* Переключение изображений происходит при клике на соответствующую миниатюру.



Рисунок 3 – Демонстрация второй карусели с миниатюрами

1. Создание адаптивного интерфейса

Для обеспечения оптимального пользовательского опыта на различных устройствах – мобильных устройствах, планшетах и компьютерах – мы использовали подход адаптивного дизайна, поддерживаемый фреймворком Tailwind CSS с его встроенными функциями адаптации.

Используя классы `sm`, `md`, `lg`, `xl`, `2xl` в Tailwind CSS, мы смогли точно настроить стили для разных размеров экранов:

1. Мобильные устройства (sm):

Для мобильных устройств, где пространство ограничено, мы применяли стили для обеспечения оптимальной читаемости и удобства использования на маленьких экранах.

1. Планшеты (md, lg):

Для планшетов, которые имеют более крупные экраны по сравнению с мобильными устройствами, мы использовали стили `md` и `lg`, чтобы адаптировать интерфейс и расположение элементов для более комфортного отображения.

1. Компьютеры (xl, 2xl):

Для экранов компьютеров с большим разрешением мы применяли стили `xl` и `2xl`, которые позволяли эффективно использовать доступное пространство и обеспечивали оптимальный дизайн для широкоформатных мониторов.

<div class="bg-blue-500 text-white p-4 sm:p-6 md:p-8 lg:p-10 xl:p-12">

<!-- Контент -->

</div>

Листинг 2. Пример использования Tailwind CSS классов для адаптации интерфейса

Такой подход позволил нам эффективно оптимизировать дизайн для разных устройств, учитывая размеры экранов и особенности их использования. Использование функций адаптации в Tailwind CSS значительно упростило процесс создания адаптивного интерфейса, позволяя быстро и точно настроить стили для разных устройств.

ЗАКЛЮЧЕНИЕ

Исследование эффективности генерации кода из макета в Figma с использованием Tailwind CSS

В процессе анализа эффективности генерации кода из макетов в Figma при использовании Tailwind CSS были выявлены ключевые преимущества данной интеграции. Оценка существующих инструментов позволила глубже понять удобство и оптимизированный подход Tailwind CSS в процессе создания веб-приложений. Технология Tailwind CSS предоставляет разработчикам широкий набор готовых стилевых классов, что значительно ускоряет процесс верстки и обеспечивает единообразие в дизайне.

Оптимизация карусели на платформе Svelte для решения проблем производительности

Оптимизация карусели на основе фреймворка Svelte была направлена на эффективное решение проблем, связанных с производительностью. Эксперименты и тщательный анализ привели к выявлению слабых мест в реализации, после чего были предложены и внедрены оптимизации. Результаты экспериментов ясно подтвердили значительное улучшение производительности карусели, что является ключевым показателем успешности внесенных изменений.

Демонстрация уникального функционала в двух реализациях каруселей

Две реализации каруселей не только предоставляют средства для эффективной визуализации контента, но и демонстрируют уникальные функциональные особенности. Интерактивность, плавные анимации, ленивая загрузка изображений и поддержка различных платформ – все эти элементы подчеркивают гибкость и эффективность веб-разработки на базе Svelte и Tailwind CSS. Такие функциональные возможности придает веб-приложениям высокую степень привлекательности и современности.

В целом, результаты исследования не только подтверждают эффективность выбранных технологий – Tailwind CSS и Svelte, но и выделяют их важную роль в современной веб-разработке. Удобство работы с макетами в Figma, оптимизация производительности карусели и функциональные возможности двух реализаций подчеркивают значимость правильного выбора инструментов для создания производительных, привлекательных и современных веб-приложений.

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]