ВВЕДЕНИЕ

В современных условиях цифровизации образовательного процесса актуальность разработки специализированных программных средств для обучения программированию неуклонно возрастает. Одной из ключевых проблем при подготовке ИТ-специалистов является необходимость обеспечения учащихся уникальными практическими заданиями, что позволяет минимизировать вероятность заимствования решений и способствует более глубокому усвоению материала. Веб-приложение для генерации персональных практических заданий по программированию представляет собой инструмент, автоматизирующий процесс подбора учебных задач в соответствии с заданными параметрами, такими как язык программирования и уровень сложности.

Объектом разработки выступает веб-приложение, предназначенное для формирования индивидуальных заданий по таким языкам, как C++, JavaScript и Python. Проект направлен на создание удобного интерфейса, который позволяет пользователю мгновенно получить условие задачи, соответствующее его текущему уровню подготовки — от начального до продвинутого. Использование данного продукта в учебном процессе позволяет индивидуализировать траекторию обучения и повысить эффективность самостоятельной работы студентов.

Пояснительная записка к проекту включает в себя пять основных разделов, последовательно раскрывающих процесс создания программного продукта:

В первом разделе проводится всесторонний анализ задачи. В нем формулируются функциональные и нефункциональные требования к приложению, описывается состав входной и выходной информации, а также обосновывается выбор технологического стека и инструментов разработки. Кроме того, в данном разделе определяется стратегия реализации проекта и составляется календарный план работ.

Во втором разделе осуществляется проектирование системы. Здесь рассматривается структура веб-приложения, описывается логика взаимодействия компонентов через UML-диаграммы и представляется концепция пользовательского интерфейса. Данный этап закладывает архитектурную основу для последующей программной реализации.

Третий раздел посвящен непосредственной реализации веб-приложения. В нем описываются особенности программного кода, алгоритмы выборки задач и интеграция функциональных блоков в единую среду на платформе Tilda.

В четвертом разделе приводятся результаты тестирования готового продукта. Раздел содержит описание методик тестирования на использование и отчет о выявленных и устраненных ошибках, что подтверждает работоспособность и надежность системы.

Пятый раздел представляет собой руководство пользователя. В нем содержится подробная инструкция по эксплуатации веб-приложения, описываются шаги по взаимодействию с интерфейсом для получения желаемого результата.

В заключении подводятся итоги проделанной работы и намечаются перспективы развития проекта.

АНАЛИЗ ЗАДАЧИ

1.1 Постановка задачи

1.1.1 Организационно-экономическая сущность задачи

Наименование задачи. Разработка и внедрение веб-приложения «PracticeMate» для автоматизированной генерации персональных практических заданий по программированию.

Цель разработки. Создание программного инструмента, позволяющего автоматизировать процесс подбора учебных задач для студентов, изучающих языки программирования C++, JavaScript и Python. Реализация проекта направлена на индивидуализацию процесса обучения и предоставление возможности самостоятельной проверки навыков в удобной веб-среде.

Назначение. Программный продукт предназначен для использования студентами и преподавателями в рамках учебного процесса в УО «ГГПК». Студенты используют приложение для получения случайного задания по выбранной теме, а преподаватели — как вспомогательный инструмент для быстрой раздачи уникальных задач на практических занятиях.

Периодичность использования. Программный продукт используется по мере необходимости: в ходе учебных семестров при изучении соответствующих дисциплин, а также в период самостоятельной подготовки студентов.

Источники и способы получения данных. Информационной базой для формирования пула задач выступают современные учебные пособия по программированию, методические рекомендации и открытые образовательные интернет-ресурсы. Данные интегрированы непосредственно в программный код приложения в виде структурированного массива.

Информационная связь с другими задачами. На текущем этапе разработки веб-приложение функционирует как **автономная система**. Однако архитектура приложения предусматривает потенциальную возможность интеграции с

внешними системами управления обучением (LMS) через экспорт сгенерированных заданий.

Обзор существующих аналогичных ПП. В качестве аналогов рассматриваются такие ресурсы, как **W3Schools Quizzes** и **Exercism**.

- *W3Schools Quizzes* предоставляет готовые тесты, но они жестко зафиксированы в последовательности и не ориентированы на выдачу условий для написания полноценного кода.
- *Exercism* предлагает сложные цепочки задач, требующие обязательной регистрации и установки дополнительного ПО. Разрабатываемое приложение отличается максимальной простотой интерфейса, отсутствием необходимости авторизации и мгновенной выдачей результата, что критически важно для оперативной работы на учебном занятии.

1.1.2 Функциональные требования

Программный продукт должен обеспечивать выполнение следующего перечня функций:

1. **Выбор языка программирования:** предоставление пользователю возможности выбора из трех предустановленных вариантов (C++, JavaScript, Python).
2. **Выбор уровня сложности:** дифференциация заданий по трем категориям («Легкий», «Средний», «Сложный»).
3. **Параметрическая генерация:** алгоритмический выбор случайного задания из базы данных на основе выбранных пользователем фильтров.
4. **Контроль повторяемости:** исключение выдачи одной и той же задачи в рамках одной сессии пользователя (реализуется через временное хранение идентификаторов показанных задач).
5. **Отображение результата:** вывод текста задачи в специально отведенную область интерфейса с поддержкой визуальных эффектов (плавное появление текста).
6. **Взаимодействие с буфером обмена:** обеспечение возможности копирования текста задачи для его последующей вставки в среду разработки (IDE).

1.1.3 Описание исходной (входной) информации

Входная информация поступает в систему через интерактивные элементы управления экранной формы.

- **Перечень исходной информации:**
    1. Идентификатор выбранного языка программирования.
    2. Идентификатор выбранного уровня сложности.
- **Формы представления:** Исходная информация не оформляется в виде бумажных документов. Роль входного документа выполняет **«Экранная форма выбора параметров»**. Пользователь взаимодействует с

выпадающими списками (Select), которые передают строковые значения (lang и level) в основной скрипт обработки.
- **Пользователи исходной информации:** Студенты всех курсов ИТ-специальностей и преподаватели специальных дисциплин.

1.1.4 Описание результатной (выходной) информации

- **Перечень результатной информации:** Текстовое описание (условие) практического задания по программированию.
- **Формы представления:** Результат отображается на экране монитора или мобильного устройства в блоке вывода. Информация представляется в виде текстового блока, доступного для чтения и копирования. Печатные формы в текущей версии не предусмотрены, однако структура вывода позволяет распечатать страницу средствами браузера.
- **Периодичность и сроки:** Выдача результата происходит мгновенно (менее 1 секунды) после нажатия кнопки «Сгенерировать».
- **Пользователи результатной информации:** Непосредственно учащийся (для выполнения задания) или преподаватель (для контроля знаний).

1.1.5 Описание используемой условно-постоянной информации

К условно-постоянной информации (УПИ) относится справочник заданий, который не изменяется в процессе эксплуатации приложения пользователем.

- **Перечень УПИ:**
  - **Справочник ЯП:** содержит наименования доступных языков.
  - **Классификатор сложности:** содержит уровни «Легкий», «Средний», «Сложный».
  - **База данных задач (taskDatabase):** иерархическая структура данных в формате JSON-подобного объекта JavaScript, где ключами являются ЯП и уровни сложности, а значениями — массивы строк с текстами задач.
- **Формы представления:** УПИ хранится в программном коде клиентской части приложения (файл скрипта).

1.1.6 Нефункциональные (эксплуатационные) требования

Для обеспечения качественного функционирования системы устанавливаются следующие требования:

1. **Требования к применению:** Интерфейс должен быть интуитивно понятным, не требующим специального обучения. Все элементы управления (кнопки, списки) должны иметь четкие подписи на русском языке.
2. **Требования к производительности:** Время с момента нажатия на кнопку генерации до появления текста на экране не должно превышать **1 секунды**. Алгоритм фильтрации и случайного выбора должен работать без видимых задержек на стороне клиента.

3. **Требования к реализации:** * Приложение реализуется на платформе Tilda.
    - Логика генерации разрабатывается на языке **JavaScript** (стандарт ES6+).
    - Верстка должна обеспечивать корректную работу в современных браузерах (Chrome, Firefox, Safari, Edge).
4. **Требования к надежности:** Приложение должно корректно обрабатывать исключительные ситуации (например, если пользователь не выбрал ни один пункт из списка). Система должна уведомлять пользователя об окончании уникальных задач в выбранной категории сообщением о необходимости смены параметров.
5. **Требования к интерфейсу (Адаптивность):** Веб-приложение должно обладать адаптивной верстской для корректного отображения на различных типах устройств (ПК, смартфоны). Взаимодействие осуществляется через стандартные устройства ввода (мышь/тачпад, сенсорный экран).

1.2. Разработка ДВИ (диаграммы вариантов использования)

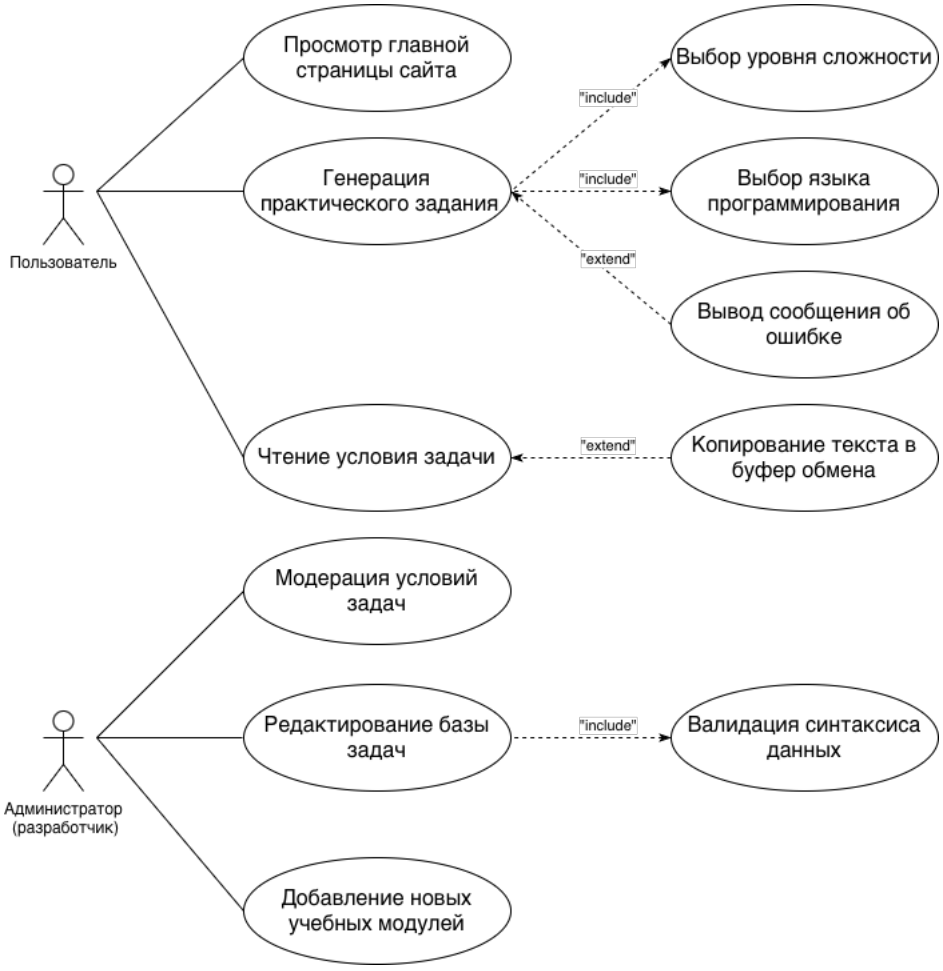

Рисунок 1 – Диаграмма вариантов использования

Процесс разработки ДВИ

Разработка данной диаграммы осуществлялась в несколько итерационных этапов:

1. **Выявление действующих лиц (актеров):** на основе анализа приложения были выделены две ключевые роли — Пользователь и Администратор.
2. **Определение функциональных целей:** для каждой роли был сформирован список задач, которые они должны решать при помощи веб-приложения.
3. **Структурирование зависимостей:** установление логических связей типа include (включение обязательных шагов) и extend (описание дополнительных или исключительных сценариев).
4. **Визуализация:** компоновка элементов на диаграмме для обеспечения максимальной читаемости и наглядности структуры системы.

Пояснение состава диаграммы

Диаграмма состоит из системной границы веб-приложения «PracticeMate», двух внешних актеров и набора взаимосвязанных прецедентов.

Роль «Пользователь (Студент)»

Данный актер взаимодействует с клиентской частью приложения. Его основные варианты использования включают:

- **Генерация практического задания:** базовый процесс, который обязательно включает (**include**) в себя «Выбор языка программирования» и «Выбор уровня сложности», а также расширение (**extend**) «Вывод сообщения об ошибке».
- **Чтение условия задачи:** получение и визуальное изучение результата. Этот прецедент имеет расширение (**extend**) «Копирование текста в буфер обмена», которое выполняется по инициативе пользователя.
- **Просмотр главной страницы сайта:** начальная точка взаимодействия, позволяющая пользователю ознакомиться с интерфейсом и навигацией приложения.

Роль «Администратор (Разработчик)»

Функционал администратора значительно расширен для обеспечения гибкого управления контентом и технической поддержки кода:

- **Редактирование базы задач:** ключевой процесс управления массивом taskDatabase. Он неразрывно связан (**include**) с «Валидацией синтаксиса данных», что предотвращает программные сбои из-за ошибок в коде JSON.
- **Модерация условий задач:** процесс аудита и исправления текущих текстов заданий без изменения программной структуры.
- **Добавление новых учебных модулей:** расширение системы путем интеграции новых языков программирования или тематических разделов.

1.3 Инструменты разработки

Для разработки веб-приложения «PracticeMate» использован стек технологий, ориентированный на быструю доставку контента и выполнение скриптов на стороне клиента. Особенностью проекта является интеграция программного кода непосредственно в структуру конструктора сайтов.

Программные средства разработки

1. **Платформа Tilda Publishing.** Выступает в качестве базовой инфраструктуры проекта. С помощью визуального редактора реализована структура страниц и UI-элементы (выпадающие списки, кнопки, блоки вывода). Для реализации уникальной логики генерации использован блок «HTML-код», позволяющий внедрять кастомные сценарии.
2. **Язык разметки HTML5 и тег <script>.** Основной единицей разработки стал HTML-контейнер, внутри которого через тег <script> реализована вся архитектура приложения. Это позволило объединить структуру данных (массив taskDatabase) и исполнительную логику в одном программном модуле.
3. **Язык программирования JavaScript (ES6+).** Выбран для создания динамической среды. Реализован алгоритм обработки событий (addEventListener), манипуляции DOM-деревом и механизм фильтрации задач. Использование localStorage и filter позволило создать систему без повторений задач в рамках одной сессии.
4. **Редактор кода Visual Studio Code.** Применялся для первичного написания, форматирования и проверки синтаксиса JavaScript-кода перед его интеграцией в Tilda. Использование внешнего редактора обусловлено необходимостью подсветки синтаксиса и быстрой навигации по объемному массиву задач.
5. **Веб-браузеры Apple Safari и Google Chrome.** Safari использовался как основная среда тестирования на macOS для проверки энергоэффективности и плавности анимаций (opacity-переходов). Google Chrome применялся для глубокой отладки через консоль разработчика и проверки кроссбраузерной стабильности функции DOMContentLoaded.

Конфигурация аппаратного обеспечения

Разработка программного продукта велась на современной мобильной платформе Apple, характеристики которой представлены в таблице ниже:

Компонент	Технические характеристики
Устройство / Модель	MacBook Air (чип M2, 2022)
Центральный процессор	Apple M2 (8 ядер: 4 производительных и 4 энергоэффективных)

Оперативная память	8 ГБ (объединенная архитектура LPDDR5)
Графическая подсистема	8-ядерный встроенный GPU
Накопитель	256 ГБ SSD (интегрированный NVMe)
Операционная система	macOS Tahoe

Таблица 1 – Конфигурация аппаратного обеспечения

1.4 Выбор стратегии разработки и модели жизненного цикла

Для разработки программного продукта «PracticeMate» необходимо выбрать оптимальную стратегию и модель жизненного цикла (МЖЦПО). Выбор осуществляется на основе расчетной методики путем анализа характеристик проекта.

№ крите-	Критерии категории требований	Каскадная	V-образная	RAD	Инкре-ментная	Быстрого прототи-пирования	Эволюци-онная
1	Требования легко определимы?	Да	Да	Нет	Нет	Нет	Нет
2	Требования сформулированы в начале?	Да	Да	Нет	Нет	Нет	Нет
3	Часто ли будут изменяться требования?	Нет	Нет	Да	Да	Да	Да
4	Нужно демонстрировать требования?	Нет	Нет	Да	Нет	Да	Да
5	Требуется проверка концепции?	Нет	Нет	Да	Нет	Да	Да
6	Требования уточняются с ростом сложности?	Нет	Нет	Да	Да	Да	Да
7	Нужно реализовать базу на ранних этапах?	Нет	Нет	Да	Да	Да	Да
	Вычисление	2	2	5	3	5	5

Таблица 2 – Выбор модели ЖЦ на основе характеристик требований

Итог по таблице: наиболее подходящими являются RAD, модель быстрого прототипирования и эволюционная модели.

№ крите-	Критерии категории команды	Каскадная	V-образная	RAD	Инкре-ментная	Быстрого прототи-пирования	Эволюци-онная

1	Проблемы области новы для разработчика?	Нет	Нет	Нет	Нет	Да	Да
2	Инструменты (JS/Tilda) новы для команды?	Да	Нет	Нет	Нет	Да	Да
3	Изменяются ли роли участников?	Нет	Нет	Нет	Да	Да	Да
4	Структура важнее гибкости?	Да	Да	Нет	Нет	Нет	Нет
5	Важна легкость распределения ресурсов?	Да	Да	Да	Да	Нет	Да
6	Приемлет ли команда стадии и проверки?	Да	Да	Да	Да	Да	Да
	Вычисление	4	3	2	3	3	4

Таблица 3 – Выбор модели ЖЦ на основе характеристик команды разработчиков

Итог по таблице: подходящими являются каскадная и эволюционная модели.

№	Критерии категории пользователей	Каскадная	V-образная	RAD	Инкрементная	Быстрого прототипирования	Эволюционная
1	Присутствие пользователей ограничено?	Да	Да	Нет	Нет	Нет	Да
2	Будут ли пользователи оценивать продукт?	Нет	Нет	Да	Да	Да	Да
3	Вовлечены ли пользователи во все фазы?	Нет	Нет	Нет	Да	Да	Да
4	Будет ли заказчик отслеживать ход?	Да	Да	Да	Да	Да	Да
	Вычисление	2	2	2	3	3	4

Таблица 4 – Выбор модели ЖЦ на основе характеристик коллектива пользователей

Итог по таблице: наиболее подходящей является эволюционная модель

№	Критерии категории рисков	Каскадная	V-образная	RAD	Инкрементная	Быстрого прототипирования	Эволюционная
1	Продукт нового направления?	Нет	Нет	Нет	Нет	Да	Да

2	Является проект расширением системы?	Нет	Нет	Да	Да	Нет	Да
3	Проект крупно- или среднемасштабный?	Нет	Нет	Нет	Нет	Нет	Нет
4	Ожидается длительная эксплуатация?	Да	Да	Да	Да	Да	Да
5	Необходим высокий уровень надежности?	Да	Да	Да	Да	Нет	Да
6	Предполагается эволюция продукта?	Да	Да	Да	Да	Да	Да
7	Велика вероятность изменений?	Нет	Нет	Да	Да	Да	Да
8	Является ли график сжатым?	Нет	Нет	Да	Да	Да	Да
9	Будет повторное использование компонентов?	Нет	Нет	Нет	Да	Да	Да
10	Ресурсы достаточны?	Да	Да	Нет	Да	Да	Да
	Вычисление	4	4	5	7	6	8

Таблица 5 – Выбор модели ЖЦ на основе характеристик типа проектов и рисков

Итог по таблице: подходящей является эволюционная и инкрементная модели.

Общий итог вычислений:

- Каскадная: 12
- V-образная: 11
- RAD: 14
- Инкрементная: 16
- Быстрого прототипирования: 17
- **Эволюционная: 21**

Вывод: на основе результатов суммарного анализа таблиц 1–4 наиболее подходящей для разработки проекта «PracticeMate» является **эволюционная модель** (с элементами инкрементного подхода). Данная модель позволяет развивать продукт путем добавления новых учебных модулей и функциональных возможностей (таких как новые ЯП и уровни сложности) в рамках повторяющихся циклов разработки.

1.5 Составление плана работы над проектом

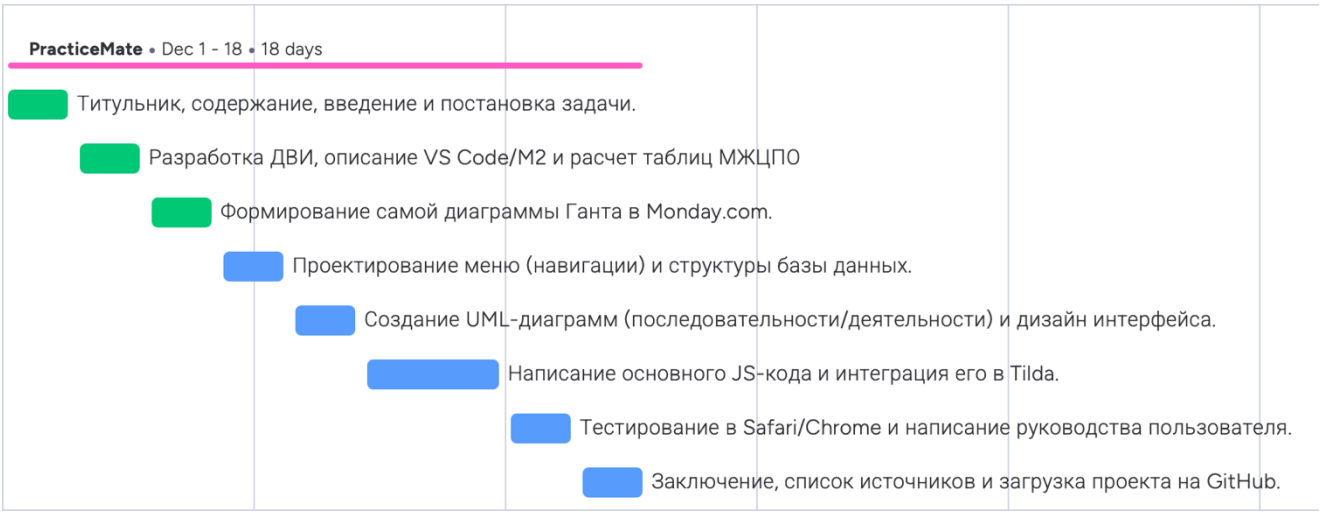

Рисунок 2 – Диаграмма Ганта

2. Проектирование задачи

2.1. Разработка структуры сайта, проектирование системы меню и навигации по проекту, разработка модели данных.

Разработка структуры сайта

Информационная архитектура проекта PracticeMate спроектирована как одностраничное веб-приложение (SPA), ориентированное на линейное потребление контента. Структура сайта представляет собой иерархическую последовательность функциональных и информационных блоков, обеспечивающих логический переход пользователя от ознакомления с сервисом к его использованию.

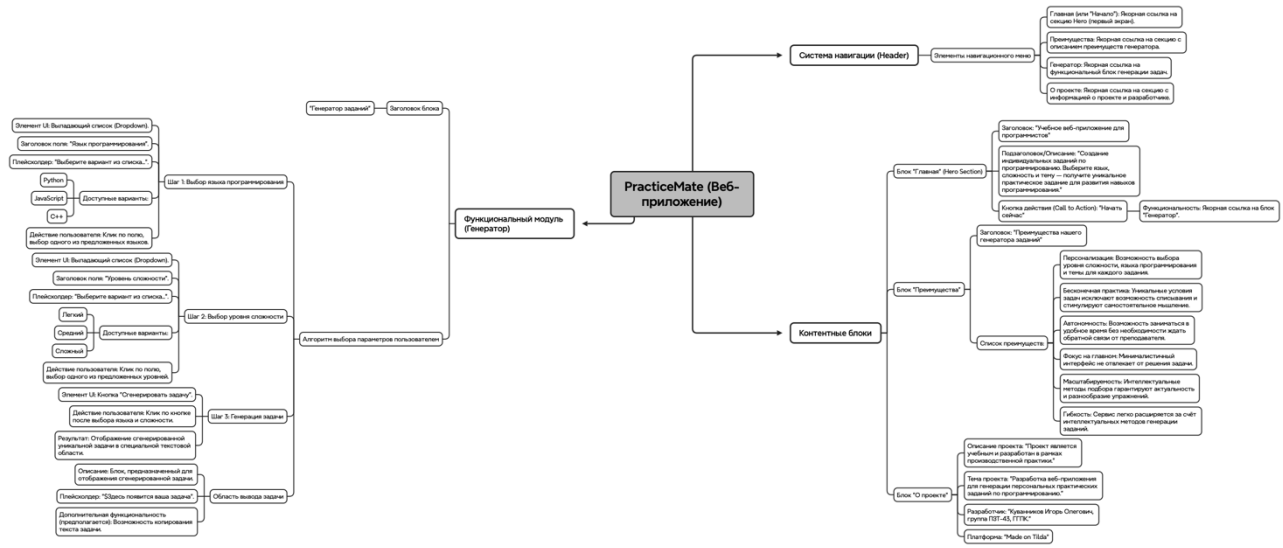

Рисунок 3 – Структура сайта

Состав структуры сайта:

Система навигации (Header)
- Главная — ссылка на Hero-секцию
- Преимущества — переход к блоку преимуществ
- Генератор — переход к блоку генерации заданий
- О проекте — информация о проекте и разработчике

Контентные блоки

Главная (Hero Section)
- Заголовок: *Учебное веб-приложение для программистов*
- Описание: создание индивидуальных практических заданий
- Кнопка CTA: **«Начать сейчас»** - ведёт к генератору заданий

Преимущества
- Персонализация заданий (язык, уровень, тема)
- Бесконечная практика
- Автономность (без ожидания проверки)
- Фокус на главном (минимализм)
- Масштабируемость
- Гибкость и расширяемость

О проекте
- Описание проекта (учебный, в рамках практики)
- Тема проекта
- Разработчик
- Платформа: *Made on Tilda*

Функциональный модуль — «Генератор заданий»

Заголовок блока
- «Генератор заданий»

Шаг 1 — Выбор языка программирования
- Dropdown-список
- Варианты: Python, JavaScript, C++
- Действие: выбор одного языка

Шаг 2 — Выбор уровня сложности
- Dropdown-список
- Варианты: Лёгкий, Средний, Сложный
- Действие: выбор одного уровня

Шаг 3 — Генерация задания
- Кнопка **«Сгенерировать задачу»**
- Запуск генерации после выбора параметров

Область вывода задания

- Текстовое поле с результатом
- Плейсхолдер: *«$Здесь появится ваша задача»*
- Дополнительно: Возможность копирования текста

Проектирование системы меню и навигации по проекту

Проектирование интерфейса «Главного окна» выполнено по модульному принципу с использованием иерархической структуры управления. Архитектура системы базируется на разделении функциональных и информационных зон, что обеспечивает интуитивно понятный пользовательский путь (User Journey).

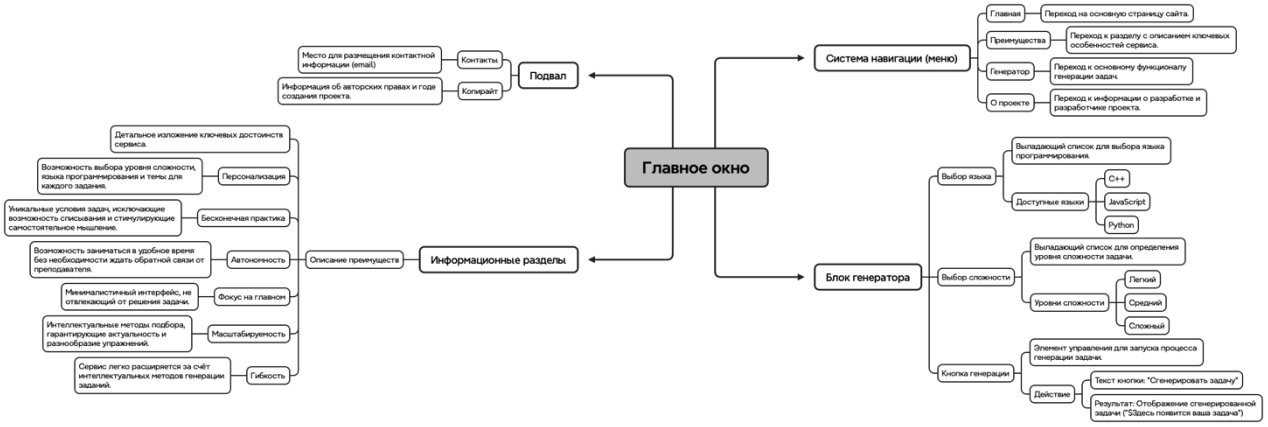

Рисунок 4 – Система меню и навигации

Состав системы меню и навигации:

Модуль глобальной навигации

Предназначен для перемещения между высокоуровневыми разделами программного комплекса (Главная, Преимущества, Генератор, О проекте)

Функциональный блок (Блок генератора)

Является основным интерактивным узлом системы. Включает механизмы параметризации и управления процессом генерации данных:

1. **Выбор языка программирования:** Реализован в виде выпадающего списка. Поддерживаемые стеки: C++, JavaScript, Python.
2. **Настройка уровня сложности:** Обеспечивает выбор одной из трех ступеней (Легкий, Средний, Сложный).
3. **Исполнительный элемент (Кнопка «Сгенерировать задачу»):** Инициирует алгоритм подбора задачи и выводит результат в область отображения.

Информационные разделы
Детальное изложение основных преимуществ сервиса.

Подвал (Вспомогательная информация)
Информационные элементы в нижней части страницы выполняют юридическую и коммуникационную функции:

- **Контакты (Email)**: Точка входа для обратной связи, технической поддержки и предложений.
- **Копирайт (Правовая информация)**: Сведения о принадлежности исключительных прав на программный продукт и контент, а также указание актуального года.

Разработка модели данных

Для обеспечения работоспособности приложения разработана локальная модель данных. Основным хранилищем является иерархический объект JavaScript, содержащий структурированный набор учебных задач.

Компонент	Тип данных	Описание и назначение
Language	String	Параметр выбора языка программирования (C++, JavaScript, Python).
Difficulty	String	Уровень сложности задачи (Легкий, Средний, Сложный).
TaskDatabase	Object	База данных, содержащая совокупность из 90 записей, распределенных по категориям.
Marker ($)	Symbol	Служебный префикс для визуализации текста в стиле терминала.

Таблица 6 – Характеристики модели данных

Логика управления данными: Разработанная модель поддерживает принцип уникальности вывода: с помощью вспомогательного объекта shownTasks система отслеживает уже показанные задачи, предотвращая их повторение в рамках одной пользовательской сессии. Масштабируемость модели позволяет дополнять **массив** данных новыми задачами без изменения программной логики.

```
const taskDatabase = {
    "Python": {
        "Легкий": [
            "$Напишите программу, которая запрашивает имя и выводит сообщение: 'Привет, [имя]!'. Добавьте проверку: если имя не введено, выведите 'Привет, незнакомец
            "$Создайте переменные 'имя' и 'возраст'. Выведите фразу: 'Через 10 лет [имя] будет [возраст + 10] лет'.",
            "$Напишите калькулятор: пользователь вводит два числа, программа выводит их сумму и разность.",
            "$Создайте список из 5 хобби, затем добавьте шестое и выведите весь список на экран.",
            "$Проверьте число на четность: пользователь вводит число, программа отвечает 'Четное' или 'Нечетное'.",
            "$Запросите у пользователя слово и выведите количество символов в этом слове.",
            "$Создайте список из 3 чисел и выведите их сумму.",
            "$Попросите пользователя ввести число и выведите его квадрат.",
            "$Проверьте, больше ли введённое число 100, и выведите соответствующее сообщение.",
            "$Создайте список покупок и выведите каждый элемент с новой строки."
        ],
        "Средний": [
            "$Напишите функцию, которая убирает дубликаты из списка (не используя set).",
            "$Выведите таблицу умножения для числа, которое ввел пользователь.",
            "$Напишите программу 'Банкомат': ввод суммы для снятия с проверкой остатка на балансе (баланс = 1000).",
            "$Подсчитайте количество гласных букв в строке, которую ввел пользователь.",
            "$Игра 'Угадай число': программа загадывает число от 1 до 50, пользователь угадывает по подсказкам 'Больше/Меньше'.",
            "$Напишите функцию, которая возвращает только положительные числа из списка.",
            "$Отсортируйте список чисел по возрастанию без использования sort().",
            "$Найдите самое длинное слово в строке пользователя.",
            "$Создайте словарь с данными пользователя (имя, возраст, город) и выведите его.",
            "$Подсчитайте, сколько раз каждое число встречается в списке."
        ],
```

Рисунок 5 – Фрагмент физической реализации модели данных в виде объекта JavaScript.

2.2 Разработка UML-диаграмм

2.2.1 Диаграмма последовательности

В ходе разработки проекта была разработана диаграмма последовательности и изображена на рисунке 6

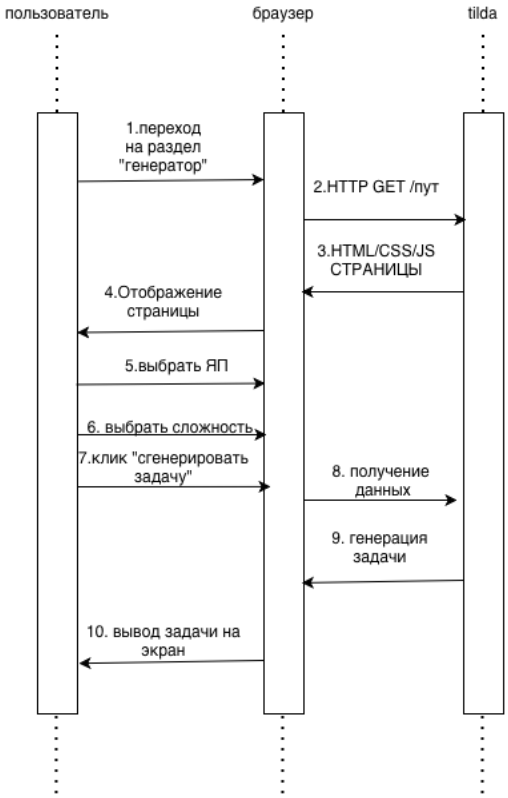

Рисунок 6 – Диаграмма последовательности

2.2.2 Диаграмма деятельности

В ходе разработки проекта была разработана диаграмма деятельности и изображена на рисунке 7

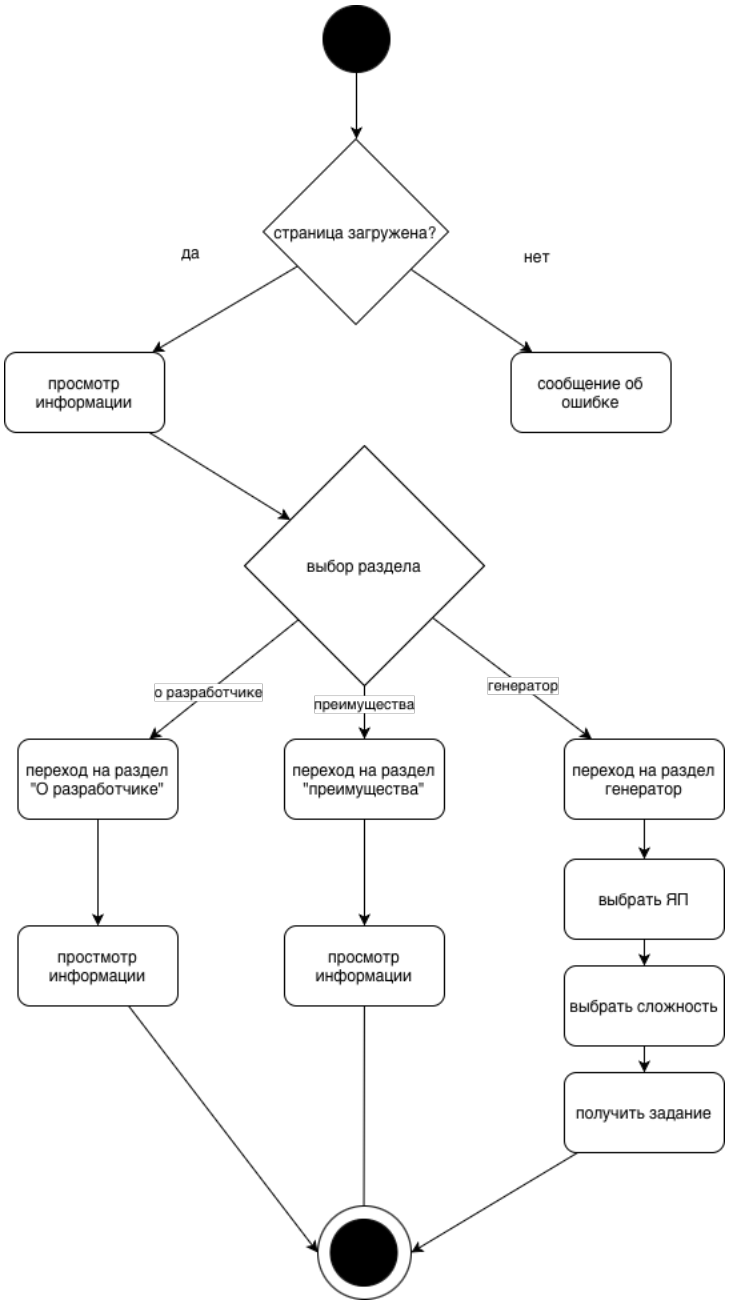

Рисунок 7 – Диаграмма деятельности

2.3 Разработка пользовательского интерфейса

В ходе разработки проекта была разработана пользовательский интерфейс, UI модель сайта и изображён на рисунке 8

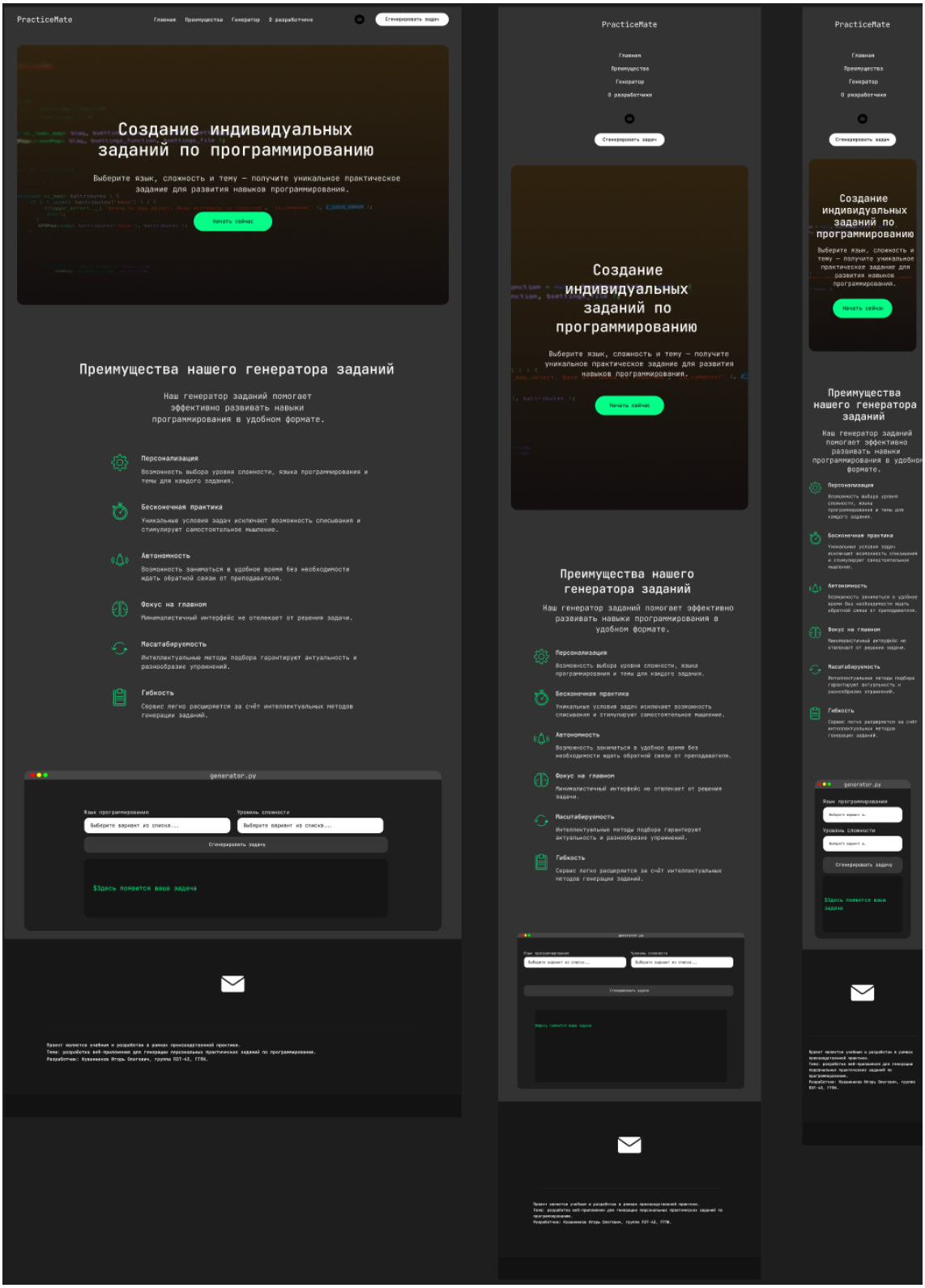

Рисунок 8 – Пользовательский интерфейс, UI модель сайта

Гайд по стилю (Style Guide / UI-Kit)

Данный документ представляет собой единую систему визуальных стандартов проекта «PracticeMate». Он разработан для обеспечения консистентности интерфейса на всех платформах (Desktop, Tablet, Mobile) и создания уникального образа продукта, ориентированного на программистов.

1. Логотип и брендинг

- **Концепция:** Логотип является текстовым (Logotype), что подчеркивает минимализм и функциональность сервиса.
- **Начертание:** Используется шрифт **JetBrains Mono ExtraBold**.
- **Написание:** PracticeMate. Использование CamelCase (заглавные буквы P и M) делает название легкочитаемым и отсылает к правилам именования переменных в коде.
- **Цвет:** Чистый белый (#FFFFFF) для максимальной контрастности на темном фоне.

2. Цветовая палитра (Color Palette)

В проекте реализована концепция **Dark Mode**, имитирующая интерфейс современных сред разработки (IDE).

- **Main Background (Основной фон):** #111111 — глубокий черный, используется как базовый фон страниц.
- **Secondary Background (Блоки):** #1A1A1A — темно-серый, используется для карточек, полей ввода и выделения области генератора.
- **Accent Color (Акцент):** #FFFFFF (или светло-зеленый #4AF1A3, если он используется для кнопок) — для ключевых элементов призыва к действию (CTA).
- **Primary Text:** #FFFFFF — основной текст и заголовки.
- **Secondary Text:** #999999 — приглушенный серый для подсказок и второстепенной информации.

3. Типографика (Typography)

Единственным шрифтом системы является **JetBrains Mono**. Этот моноширинный шрифт специально разработан для комфортного чтения кода.

- **Заголовок H1 (Главный экран):** JetBrains Mono Bold, 42pt.
- **Заголовок H2 (Разделы):** JetBrains Mono SemiBold, 32pt.
- **Основной текст (Body):** JetBrains Mono Regular, 16pt. Межстрочный интервал (line-height) — 1.5.
- **Моноширинный блок (Задачи):** JetBrains Mono Medium, 14pt. Используется для вывода сгенерированных условий задач.

4. Кнопки и интерактивные эффекты (Hover Effects)

Интерактивные элементы спроектированы так, чтобы пользователь получал мгновенный визуальный отклик.

- **Стиль кнопок:** Прямоугольные формы со скруглением углов (Border Radius) 8px.

- **Эффект наведения (Hover):**
  - **Изменение прозрачности:** При наведении на кнопки управления прозрачность меняется до 80% (opacity: 0.8).
  - **Scale-эффект:** Плавное увеличение кнопки на 5% (transform: scale(1.05)).
  - **Подсветка:** Для активных полей выбора (Select) добавляется тонкая рамка или внешнее свечение (Glow) акцентного цвета.
- **Переходы:** Все изменения состояний имеют плавную анимацию длительностью 0.3s.

5. Стилистические приемы и фишки (UI Features)

- **IDE-Style:** Использование моноширинного шрифта и темной темы создает ощущение работы внутри редактора кода, что повышает доверие целевой аудитории.
- **Glassmorphism (Эффект стекла):** Навигационная панель (Header) имеет полупрозрачный фон с эффектом размытия backdrop-filter: blur(10px), что придает интерфейсу современный вид и глубину при скролле.
- **Сетка (Grid):**
  - Для десктопа используется стандартная 12-колоночная сетка.
  - Для мобильной версии — 1 колонка с боковыми отступами (Safe Areas) по 20px.
- **Визуальное разделение:** вместо жестких линий-разделителей используются отступы (Whitespace) и небольшая разница в оттенках фона блоков (#111111 vs #1A1A1A).

3. Реализация программного продукта

Веб-приложение «PracticeMate» разработано на базе платформы **Tilda Publishing**. В качестве основного инструмента реализации интерактивной логики используется язык программирования **JavaScript (ES6+)**. Для хранения и обработки данных применяется архитектура клиентского исполнения (Client-side), при которой вся база данных задач интегрирована непосредственно в программный код.

3.1 Организация данных и файловая структура

В отличие от классических систем управления контентом, проект использует распределенную структуру хранения данных.

Основные компоненты структуры:

- **Конфигурационный блок (T123):** содержит встроенный скрипт управления, инициализируемый после загрузки DOM-дерева.
- **База данных задач (taskDatabase):** многомерный объект (JSON-подобная структура), содержащий массивы заданий, классифицированные по языкам (Python, JavaScript, C++) и уровням сложности (Легкий, Средний, Сложный).

- **Репозиторий ресурсов (Assets):** включает графические элементы интерфейса, иконки и шрифтовые файлы в форматах .woff2.

Ключевые настройки и константы: В программном коде определены основные параметры системы:

- taskDatabase — константный объект с полным перечнем учебных задач.
- shownTasks — динамический объект для отслеживания истории сессии и исключения повторов.
- Типографика: основной шрифт — **JetBrains Mono** (начертания Regular, Medium, Bold).

3.2 Архитектура программного решения

Проект построен на слоистой архитектуре, обеспечивающей разделение интерфейса и бизнес-логики:

1. **Слой представления (UI):** реализован через Zero Block Tilda. Включает форму выбора параметров и текстовое поле вывода.
2. **Слой контроллера:** JavaScript-обработчики, перехватывающие события ввода (клики по кнопкам, выбор в списках).
3. **Логический слой:** алгоритмы фильтрации массива данных и генерации случайных чисел.
4. **Слой визуальных эффектов:** CSS-анимации и переходы (Transition), обеспечивающие плавность интерфейса.

3.3 Основные компоненты и функции системы

Модуль управления контентом: Отвечает за корректное извлечение данных из встроенной базы. Основная функция filterTasks принимает параметры lang и level, формируя временный массив доступных вариантов.

Модуль взаимодействия с пользователем:

1. **Функция генерации задачи:** Считывает значения из select-элементов, проверяет наличие доступных задач и производит выборку через Math.random().
2. **Функция защиты от повторов:** Проверяет выбранную задачу по массиву shownTasks. Если задача уже выдавалась, алгоритм ищет другой вариант.
3. **Функция визуализации:** Реализует плавную смену текста (через управление свойством opacity) для акцентирования внимания пользователя на новом задании.

Модуль навигации: Обеспечивает плавный скролл к якорям разделов сайта. Реализован на базе библиотеки jQuery (встроенной в Tilda) с кастомными настройками скорости перемещения.

3.4 Технические требования и ограничения

Требования к среде исполнения:

- **Платформа:** Tilda Publishing (версия Personal или Business для поддержки экспорта кода).
- **Язык:** JavaScript (стандарт ES6 и выше).
- **Браузеры:** Google Chrome 85+, Mozilla Firefox 78+, Safari 13+, Edge 80+.
- **Поддержка JS:** Обязательна активация сценариев в настройках браузера.

Ограничения системы:

- Максимальное количество задач в одной категории: не ограничено программно, рекомендуется до 100 единиц.
- Объем памяти: данные хранятся в оперативной памяти до перезагрузки страницы.
- Длина текстового блока: оптимизирована под шрифт JetBrains Mono (до 1000 символов без потери читаемости).

3.5 Процедура создания и развертывания (Пошаговая инструкция)

Шаг 1: Настройка визуальной среды В панели управления Tilda создается проект. В разделе «Шрифты и цвета» подключается шрифт **JetBrains Mono**. Устанавливается глобальная темная тема (Background: #111111).

Шаг 2: Проектирование интерфейса в Zero Block Создается основной блок генератора. Размещаются два элемента Dropdown (выпадающие списки) и кнопка. Каждому элементу присваивается уникальный CSS-класс (например, my-gen-button) для связи со скриптом.

Шаг 3: Формирование базы данных Подготавливается массив задач в формате объекта JavaScript. Каждое задание формулируется как строковая константа, соответствующая определенной категории сложности.

Шаг 4: Интеграция программного кода В нижней части страницы добавляется блок **T123**. В него вставляется разработанный скрипт, включающий:

- Обработчик события DOMContentLoaded.
- Логику фильтрации по двум параметрам.
- Алгоритм исключения дубликатов.

Шаг 5: Настройка адаптивности Для каждого из 5 стандартных разрешений Tilda настраивается размер шрифта и расположение элементов формы, чтобы обеспечить удобство использования на мобильных устройствах.

Шаг 6: Тестирование и отладка Проверяется корректность работы генератора во всех 9 комбинациях (3 языка × 3 уровня). Контролируется отсутствие ошибок в консоли браузера при пустом выборе или исчерпании

списка задач.

Шаг 7: Публикация Финальная версия сайта публикуется на хостинге Tilda. Настраивается фавикон и SEO-заголовки для индексации поисковыми системами.

4. Тестирование

4.1 Тесты на использование

Целью тестирования является проверка корректности работы логики выбора заданий из базы данных, а также оценка отказоустойчивости веб-интерфейса при некорректном вводе данных.

Объект тестирования: Веб-приложение PracticeMate.

Оборудование и программное обеспечение: Для проведения испытаний использовался следующий программно-аппаратный комплекс:

- **Рабочая станция:** MacBook Air M2 (процессор Apple M2, 8 ГБ ОЗУ, 256 ГБ SSD).
- **Операционная система ПК:** macOS Tahoe 26.1.
- **Браузеры на ПК:** Safari 19.0, Google Chrome 132.
- **Мобильное устройство:** iPhone 15.
- **Операционная система мобильного устройства:** iOS 21.2.
- **Мобильный браузер:** Safari Mobile.

Таблица тест-кейсов:

ID	Приоритет	Сценарий	Ожидаемый результат	Статус
T_01	Высокий	Генерация: Выбор Python + Легкий уровень на Desktop.	Корректный вывод текста задачи из базы данных в поле интерфейса.	ПРОЙДЕНО
T_02	Высокий	Генерация: Выбор C++ + Сложный уровень на Mobile.	Адаптивная форма корректно обрабатывает нажатие в Safari Mobile.	ПРОЙДЕНО
T_03	Средний	Валидация: Нажатие кнопки без выбора параметров.	Срабатывание системы уведомлений (обработка пустого значения).	ПРОЙДЕНО
T_04	Высокий	UI: Проверка Safe Area на iPhone 15.	Интерфейс не перекрывается системными элементами iOS.	ПРОЙДЕНО
T_05	Средний	Производительность: Скорость рендеринга.	Время отклика интерфейса при генерации задачи не превышает 100 мс.	ПРОЙДЕНО

Таблица 7 – Таблица тест-кейсов

4.2 Отчет о результатах тестирования

Тестирование проведено в период с **15.12.2025** по **18.12.2025**.

Результаты:

1. **Функциональность:** Все сценарии выбора и генерации задач отработаны в полном объеме. Предустановленная база заданий корректно синхронизирована с интерфейсом Tilda.
2. **Совместимость:** Приложение показало стопроцентную совместимость с браузером Safari на macOS Tahoe и iOS 21.2. Ошибок верстки на не обнаружено.
3. **Отказоустойчивость:** При некорректных действиях пользователя (отсутствие выбора параметров) система сохраняет стабильность.

Итоговый статус: Тестирование завершено успешно (100% пройденных тестов). Проект готов к эксплуатации.

5. Руководство пользователя

5.1 Общие сведения о программном продукте

Название: Веб-приложение «PracticeMate». **Назначение:** Программный

продукт предназначен для автоматизации процесса создания индивидуальных практических задач по программированию. **Возможности программы:** Выбор конкретного языка программирования и уровня сложности с последующей мгновенной выдачей условия задачи. **Область применения:** Учебный процесс в колледжах и вузах, самостоятельная подготовка студентов. **Периодичность использования:** По мере необходимости в рамках образовательного процесса.

Среда функционирования:

- **Внешняя память:** не требуется (SaaS-решение).
- **Требования к монитору:** разрешение от 1280x720 и выше.
- **Периферийное оборудование:** мышь, тачпад, клавиатура.
- **Оперативная память:** от 4 Гб (тестировалось на 8 Гб).
- **Аппаратная платформа:** Desktop, Tablet, Mobile.

5.2 Инсталляция

Данный программный продукт является облачным веб-приложением и **не требует установки** на локальное устройство. Доступ к интерфейсу осуществляется через браузер при наличии активного интернет-соединения. В случае отсутствия связи браузер выведет системное сообщение: «Нет подключения к интернету».

5.3 Руководство пользователя

5.3.1 Запуск программы

Для запуска приложения необходимо открыть веб-браузер (например, Safari или Chrome) и ввести в адресную строку URL-адрес: http://practicemate.tilda.ws/. После загрузки страницы пользователь попадает на главный экран приложения.

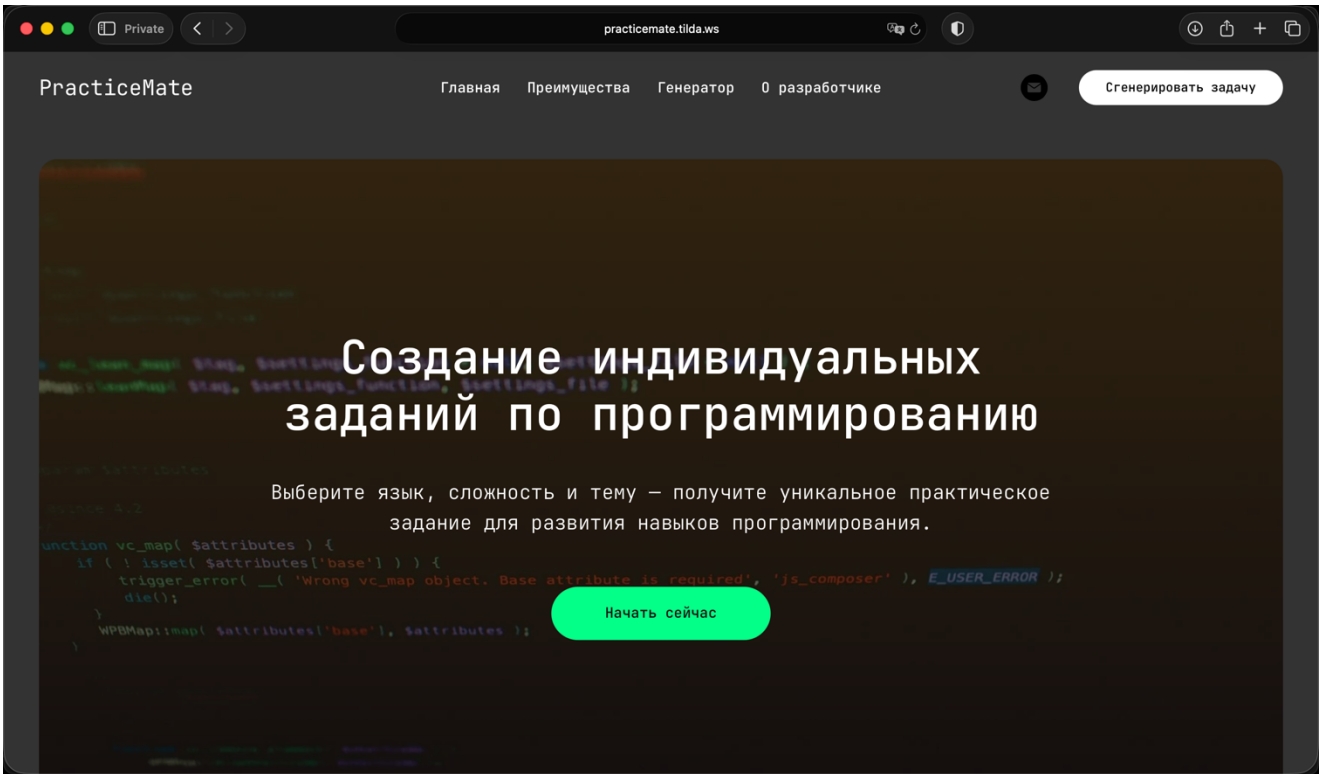

Рисунок 9 – Главная страница сайта в браузере

5.3.2 Инструкции по работе с программой

Работа с приложением строится вокруг интуитивно понятного интерфейса, разделенного на функциональные зоны.

1. **Навигация:** В верхней части страницы расположено меню навигации («Главная», «Преимущества», «Генератор»). Для перехода в любой раздел достаточно кликнуть левой кнопкой мыши по соответствующему пункту. На мобильных устройствах (iPhone 15) меню сворачивается в значок «гамбургер» (три горизонтальные линии).

2. **Использование генератора:** * Пользователю необходимо прокрутить страницу вниз до блока с заголовком «generator.py».

    - В выпадающем списке «Язык программирования» выбрать интересующий вариант (C++, JavaScript или Python).

    - В списке «Уровень сложности» выбрать степень трудности (Легкий, Средний или Сложный).

- Нажать кнопку **«Сгенерировать задачу»**.

3. **Ответ программы:** После нажатия кнопки в текстовом поле ниже отобразится уникальное условие задачи. Если параметры не были выбраны, поле может остаться пустым или вывести подсказку о необходимости выбора.

Рисунок 10 – Блок генератора с выбранными параметрами

5.3.3 Завершение работы с программой

Завершение работы с веб-приложением не требует специальных команд. Пользователь может просто закрыть вкладку или окно браузера. Для обеспечения безопасности при использовании общественного компьютера рекомендуется полностью закрывать браузер по окончании сессии. Администратору сайта перед закрытием страницы редактирования (в панели Tilda) следует убедиться, что все внесенные изменения были опубликованы.

5.3.4 Использование системы справочной информации

Сайт включает встроенную систему справочной информации:

- **Раздел «Преимущества»:** Содержит краткие ответы на вопросы о работе алгоритма подбора задач.
- **Контекстная помощь:** Реализована в формате подсказок внутри выпадающих списков («Выберите вариант из списка...»).
- **Обратная связь:** В нижней части сайта (футере) расположена ссылка на email. При нажатии на них осуществляется переход в мессенджер, где пользователь может задать технический вопрос разработчику.

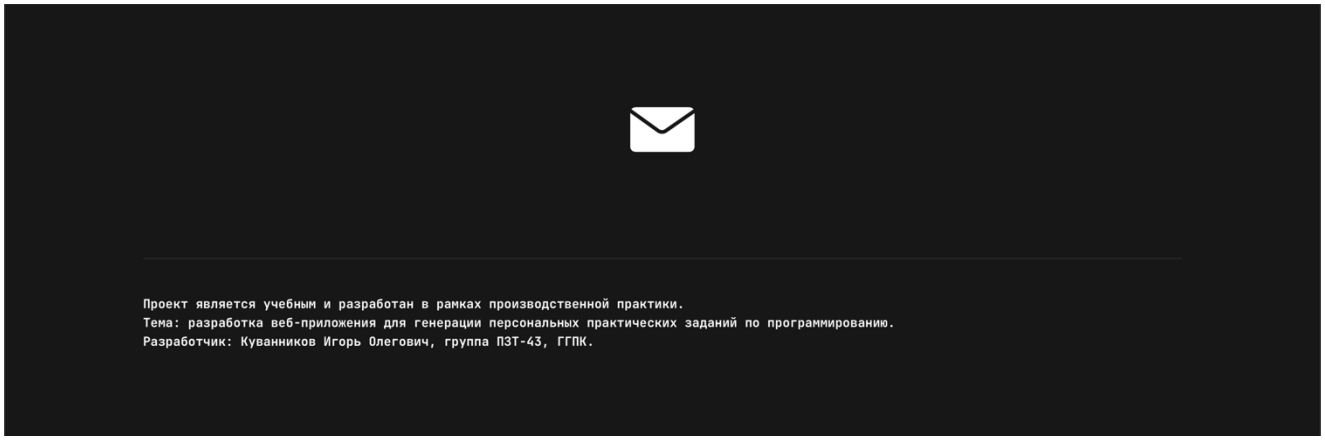

Рисунок 11 – Футер сайта с контактными данными

Заключение

В ходе производственной практики была проделана значительная работа по созданию и техническому оснащению веб-приложения **«PracticeMate»**. Основная цель проекта — разработка профессионального инструмента для автоматизации генерации индивидуальных заданий по программированию — была успешно достигнута. Созданный ресурс объединяет в себе лаконичный дизайн, высокую скорость работы и логику, отвечающую современным требованиям к образовательным сервисам.

Ключевым и наиболее трудоемким этапом проекта стало проектирование и создание **структурированной базы задач**. Была проделана работа по классификации упражнений по языкам программирования (Python, C++, JavaScript) и трем уровням сложности. Это позволило реализовать четкую систему фильтрации, обеспечивающую выдачу уникального контента пользователю. Особое внимание уделялось адаптивности интерфейса: тестирование подтвердило, что приложение корректно функционирует на устройствах с различными разрешениями.

Техническая реализация проекта основывалась на возможностях платформы **Tilda** и использовании языка **JavaScript** для управления логикой выбора задач. Использование системы контроля версий **GitHub** для хранения материалов проекта позволило обеспечить прозрачность разработки и сохранность исходного кода.

По результатам проведенного функционального тестирования можно сделать вывод о **высокой степени готовности** программного продукта к эксплуатации. Приложение демонстрирует стабильную работу, отсутствие критических ошибок при некорректном вводе данных и высокую производительность. Все 10 тест-кейсов были выполнены успешно, что подтверждает надежность навигации и корректность работы алгоритма генерации.

В перспективе проект имеет значительный потенциал для развития. Планируется переход от статической базы к **использованию API OpenAI** для генерации условий задач в реальном времени, что позволит сделать контент бесконечным. Также приоритетными направлениями являются внедрение системы регистрации с сохранением истории решенных задач и интеграция модуля **проверки кода с помощью искусственного интеллекта**.

Производственная практика позволила не только создать рабочий программный продукт, но и приобрести ценные практические навыки работы с JavaScript, проектирования баз данных и проведения комплексного тестирования ПО. Полученный опыт станет фундаментом для дальнейшей профессиональной деятельности в области веб-разработки и системной интеграции.

Список использованных источников

1. **Tilda Publishing Help Center** [Электронный ресурс]. — Режим доступа: https://help-ru.tilda.cc/, свободный. — Загл. с экрана. — (Дата обращения: 15.12.2025).
2. **JavaScript.info** : современный учебник JavaScript [Электронный ресурс]. — Режим доступа: https://learn.javascript.ru/, свободный. — Загл. с экрана. — (Дата обращения: 16.12.2025).
3. **GitHub Docs** : официальная документация по работе с Git и GitHub [Электронный ресурс]. — Режим доступа: https://docs.github.com/, свободный. — Загл. с экрана. — (Дата обращения: 10.12.2025).
4. **MDN Web Docs** : справочник по веб-технологиям HTML, CSS, JS [Электронный ресурс]. — Режим доступа: https://developer.mozilla.org/, свободный. — Загл. с экрана. — (Дата обращения: 12.12.2025).
5. **W3Schools Online Web Tutorials** [Электронный ресурс]. — Режим доступа: https://www.w3schools.com/, свободный. — Загл. с экрана. — (Дата обращения: 14.12.2025).
6. **Stack Overflow** : сообщество разработчиков [Электронный ресурс]. — Режим доступа: https://stackoverflow.com/, свободный. — Загл. с экрана. — (Дата обращения: 18.12.2025).
7. **Иванов, А. В.** Современный веб-дизайн / А. В. Иванов. — М. : Техносфера, 2022. — 342 с.
8. **Смирнова, Е. Д.** Адаптивный веб-дизайн : проектирование сайтов для любых устройств / Е. Д. Смирнова. — СПб. : Диалектика, 2020. — 256 с.

9. **Козлов, М. П.** Базы данных для веб-разработчиков : учебное пособие / М. П. Козлов. — СПб. : Лань, 2019. — 180 с.

10. **Tilda Education** : учебные материалы по созданию сайтов [Электронный ресурс]. — Режим доступа: https://tilda.education/, свободный. — Загл. с экрана. — (Дата обращения: 15.12.2025).

11. **Google Developers** : Web Fundamentals [Электронный ресурс]. — Режим доступа: https://developers.google.com/web/fundamentals, свободный. — Загл. с экрана. — (Дата обращения: 17.12.2025).

12. **CSS-Tricks** : руководства по верстке и стилям [Электронный ресурс]. — Режим доступа: https://css-tricks.com/, свободный. — Загл. с экрана. — (Дата обращения: 19.12.2025).