

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего
образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

Направление подготовки: 09.03.01 «Информатика и вычислительная техника»

Профиль подготовки: «Вычислительные машины, системы, комплексы и
сети»

БАКАЛАВРСКАЯ РАБОТА

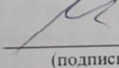
на тему

ПРИЛОЖЕНИЕ А/В ТЕСТИРОВАНИЯ
ПОЛЬЗОВАТЕЛЬСКИХ ИНТЕРФЕЙСОВ
САЙТОВ

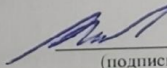
Студент

 27.05.2026 Лосяков Глеб Сергеевич
(подпись, дата) (ФИО полностью)

Руководитель

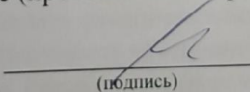
 27.05.2026 Митрохин М.А.
(подпись, дата) (фамилия, инициалы)

Нормоконтролёр

 27.05.2026 Бычков А.С.
(подпись, дата) (фамилия, инициалы)

Работа допущена к защите (протокол зас. кафедры от 27.05.2026 г. № 14)

Заведующий кафедрой

 Митрохин М.А.
(подпись) (фамилия, инициалы)

Работа защищена с отметкой хорошо (протокол заседания ГЭК от
08 июня 2026 г. № 3)

Секретарь ГЭК

 Гурин Е.И.
(подпись) (фамилия, инициалы)

Пенза 2026

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

«Утверждаю»
Заведующий кафедрой
М.А. Митрохин
«07» 04 2026 г.

**ЗАДАНИЕ
НА БАКАЛАВРСКУЮ РАБОТУ**

1. Студент Лосяков Глеб Сергеевич гр.22BBB1 факультета вычислительной техники.

2. Тема работы: Приложение А/В тестирования пользовательских интерфейсов сайтов

Тема утверждена приказом ПГУ № 347/о от "7" апреля 2026 г.

3. Руководитель бакалаврской работы д.т.н кафедры ВТ ПГУ Митрохин М.А.

4. Задание на работу (назначение разработки, исходные данные и т.п.)

Разработать веб-приложение А/В-тестирования пользовательских интерфейсов сайтов. Комплекс должен обеспечивать создание и администрирование тестовых кампаний, реализовывать механизмы динамического перенаправления и подмены трафика для посетителей целевых ресурсов, осуществлять сплошной сбор и обработку аналитических данных, выполнять расчет статистической значимости результатов и автоматически формировать аналитические заключения для пользователя системы.

5. Перечень подлежащих разработке вопросов

Произвести анализ предметной области веб-аналитики, исследовать существующие аналогичные решения для А/В-тестирования и обосновать актуальность разработки собственного веб-приложения. Определить функциональное назначение, технические требования и общую архитектуру системы. Выполнить обоснованный выбор программных инструментов и стека технологий разработки. Разработать программное обеспечение серверной части и клиентской части. Произвести опытную проверку работы веб-приложения и оценить эффективность применяемых алгоритмов распределения трафика.

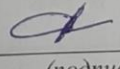
6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объём ра- боты	Срок вы- полнения	Подпись ру- ководителя
1	Изучение предметной области	30	07.04.2026 – 10.04.2026	
2	Анализ аналогов, обоснование актуальности, постановка задачи	24	10.04.2026 – 13.04.2026	
3	Выбор и настройка инструментов разработки	8	13.04.2026 – 14.04.2026	
4	Проектирование архитектуры приложения	40	14.04.2026 – 20.04.2026	
5	Проектирование классов и модуля сбора информации	60	20.04.2026 – 28.04.2026	
6	Разработка модуля расчета статистической значимости	45	28.04.2026 – 07.05.2026	
7	Проектирование API	40	07.05.2026 – 13.05.2026	
8	Разработка модуля извлечения распределения трафика	55	13.05.2026 – 15.05.2026	
9	Реализация пользовательского интерфейса	40	15.05.2026 – 16.05.2026	
10	Тестирование и отладка	20	16.05.2026 – 17.05.2026	
11	Оформление пояснительной записки	20	18.05.2026 – 21.05.2026	

Дата выдачи задания " 07 " апреля 2026г.

Руководитель бакалаврской работы  Митрохин М.А.
(подпись, дата)

Задание к исполнению принял студент  Лосяков Г.С.
(подпись, дата)

Работу к защите допустить
Декан факультета  Фионова Л.Р.
(подпись, дата)
08.06.2028

Реферат

Бакалаврская работа содержит 89 листов, 33 рисунка, 3 таблицы, 9 источников, 3 приложения.

ВЕБ-ПРИЛОЖЕНИЕ, А/В ТЕСТИРОВАНИЕ, АЛГОРИТМ ТОМПСОНА, FASTAPI, PYTHON, REACT, ОПТИМИЗАЦИЯ КОНВЕРСИИ, СБОР АНАЛИТИКИ, ДИНАМИЧЕСКОЕ РАСПРЕДЕЛЕНИЕ ТРАФИКА

Цель работы – разработать приложение для А/В тестирования пользовательских интерфейсов сайтов.

Объектом разработки является веб-приложение, которое должно выполнять функции создания А/В тестов, реализовывать механизм подмены трафика для пользователей сайта, собирать и обрабатывать данные тестов, выдавать пользователю заключение по полученным данным.

В результате проделанной работы разработано приложение, полностью соответствующее целям выпускной квалификационной работы.

					ПГУ 1.09.03.01.08.001 ПЗ			
Изм.	Лист	№ докум.	Под	Дата	Приложение А/В тестирования пользовательских интерфейсов сайтов	Лит.	Лист	Листов
Разраб.		Лосяков Г.С.		27.08.24			3	92
Провер.		Митрохин М.А.		27.08.24		ФВТ, каф. ВТ		
Н. Контр.		Бычков А.С.		27.08.24				
Утверд.		Митрохин М.А.		27.08.24		гр. 22ВВВ1		

Содержание

Перечень сокращений и обозначений	6
Введение	7
1 Приложение А/В тестирования пользовательских интерфейсов сайтов	8
1.1 Анализ предметной области	8
1.1.1 Анализ аналогов	13
1.1.2 Постановка задачи	15
1.2 Выбор инструментов разработки	18
1.2.1 Выбор IDE для написания кода	18
1.2.1.1 Visual Studio Code	18
1.2.1.2 PyCharm	19
1.2.1.3 Sublime Text	19
1.2.2 Выбор языка программирования	20
1.2.2.1 Java	20
1.2.2.2 Python	21
1.2.2.3 HTML5	22
1.2.2.4 CSS3	22
1.2.2.5 JavaScript	22
1.2.3 Выбор СУБД	24
1.2.3.1 SQLite	24
1.2.3.2 PostgreSQL	25
1.2.4 Выбор библиотек и фреймворков	25
1.2.4.1 FastAPI	25
1.2.4.2 SQLAlchemy	26
1.2.4.3 SciPy	26
1.2.4.4 Charts.js	27
1.3 Установка и настройка рабочих сред	27
1.3.1 Установка IDE Visual Studio Code	27
1.3.2 Установка СУБД PgAdmin	30
2 Реализация приложения	32

					ПГУ 1.09.03.01.08.001 ПЗ	Лист
						4
Изм.	Лист	№ докум.	Подпись	Дата		

2.1 Разработка алгоритма рапределения трафика	33
2.2 Проектирование классов	36
2.3 Разработка алгоритма аналитики данных	38
2.4 Проектирование эндпоинтов	41
2.5 Разработка скрипта-симулятора действий пользователей	45
2.6 Разработка встраиваемого сниппета	46
3 Тестирование приложения	46
3.1 Создание теста	47
3.2 Начало тестирования	50
3.3 Тестирование с помощью скрипта	53
4 Безопасность жизнедеятельности	58
4.1 Требования к организации рабочего места	58
4.2 Требование к освещению	59
4.3 Требования к режиму отдыха	60
Заключение	62
Список использованных источников	63
Приложение А – UML-диаграммы	64
Приложение Б – Листинг программы	69
Приложение В – Презентация	92

Перечень сокращений и обозначений

В настоящей работе применяют следующие сокращения и обозначения

БД – база данных

					ПГУ 1.09.03.01.08.001 ПЗ	Лист
						5
Изм.	Лист	№ докум.	Подпись	Дата		

ОС – операционная система

ПК – персональный компьютер

СУБД – система управления базами данных

API (Application Programming Interface) – программный интерфейс приложения

CORS (Cross-origin resource sharing) – технология, которая

позволяет предоставлять веб-страницам доступ к объектам сторонних

интернет-ресурсов

CRUD (Create, Read, Update, Delete) – аббревиатура, обозначающая четыре основные операции управления данными

IDE (Integrated Development Enviroment) – интегрированная среда разработки

ORM (Object-Relational Mapping) – технология программирования, которая облегчает взаимодействие между объектно-ориентированным программированием и реляционными базами данных

SQL (Structured Query Language) – язык структурированных запросов

UML (Unified Modeling Language) – унифицированный язык моделирования

Virtual DOM (виртуальное объектное дерево) – концепция программирования, при которой идеальное, или «виртуальное», представление пользовательского интерфейса хранится в памяти и синхронизируется с «реальным» объектным деревом

					ПГУ 1.09.03.01.08.001 ПЗ	Лист
						6
Изм.	Лист	№ докум.	Подпись	Дата		

Введение

В современном мире для решения любой проблемы существуют десятки приложений похожих друг на друга. Чаще всего для рядового пользователя самым важным в приложении является не его функционал, а удобство использования, которое заключается в простоте и привлекательности интерфейса, удобстве навигации и скорости работы приложения. И для того, чтобы определить какой интерфейс больше понравится пользователям, используются А/В тесты.

Основная цель выпускной работы – разработка веб-приложения, реализующего возможность проведения А/В тестирования на веб-страницах, а также сбор и анализ статистики с помощью современных инструментов и математических функций.

Результатом работы будет создание веб-приложения, отвечающего современным стандартам разработки, которое будет сочетать в себе удобный интерфейс и функциональные возможности, необходимые для проведения А/В тестирования.

					ПГУ 1.09.03.01.08.001 ПЗ	Лист
						7
Изм.	Лист	№ докум.	Подпись	Дата		

1 Приложение А/В тестирования пользовательских интерфейсов сайтов

1.1 Анализ предметной области

А/В тестирование – это метод исследования, при котором разным группам посетителей сайта одновременно показаны две версии одной и той же веб-страницы для определения, какая из них работает эффективнее. А/В тестирование или групповое тестирование, по сути, представляет собой эксперимент, в котором пользователям случайным образом показываются два или более варианта рекламы, маркетингового электронного письма или веб-страницы, а затем используются различные методы статистического анализа для определения какой вариант дает больше конверсий.

Суть метода заключается в разделении потока на две группы: на контрольную группу (вариант А) – пользователи, которые видят исходную версию интерфейса, и на тестовую группу (вариант В) – пользователи, которые видят версию интерфейса, в которую было внесено одно изменение (например, текст кнопки, текст заголовка, структура блока). Также система должна гарантировать, что каждый пользователь попадает в каждую из групп случайным образом и принадлежность к ней не изменится после повторного захода на сайт.

В данном программном комплексе представлены два режима разделения пользователей на группы: жесткое разделение 50/50 и многорукий бандит. Первый метод используется для проведения исследований, когда нам важно получить максимально объективные цифры. Второй используется для уменьшения потерь от конверсии от использования худшего варианта. По сути, алгоритм многоруких бандитов начинается с отправки трафика на две (или более) страницы: оригинал и его вариант. Затем он обновляется в зависимости от того, какая из вариаций является наиболее эффективной. В конце концов, алгоритм определяет лучший вариант из возможных. Разница в работе алгоритмов показана на рисунке 1.

					ПГУ 1.09.03.01.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		8

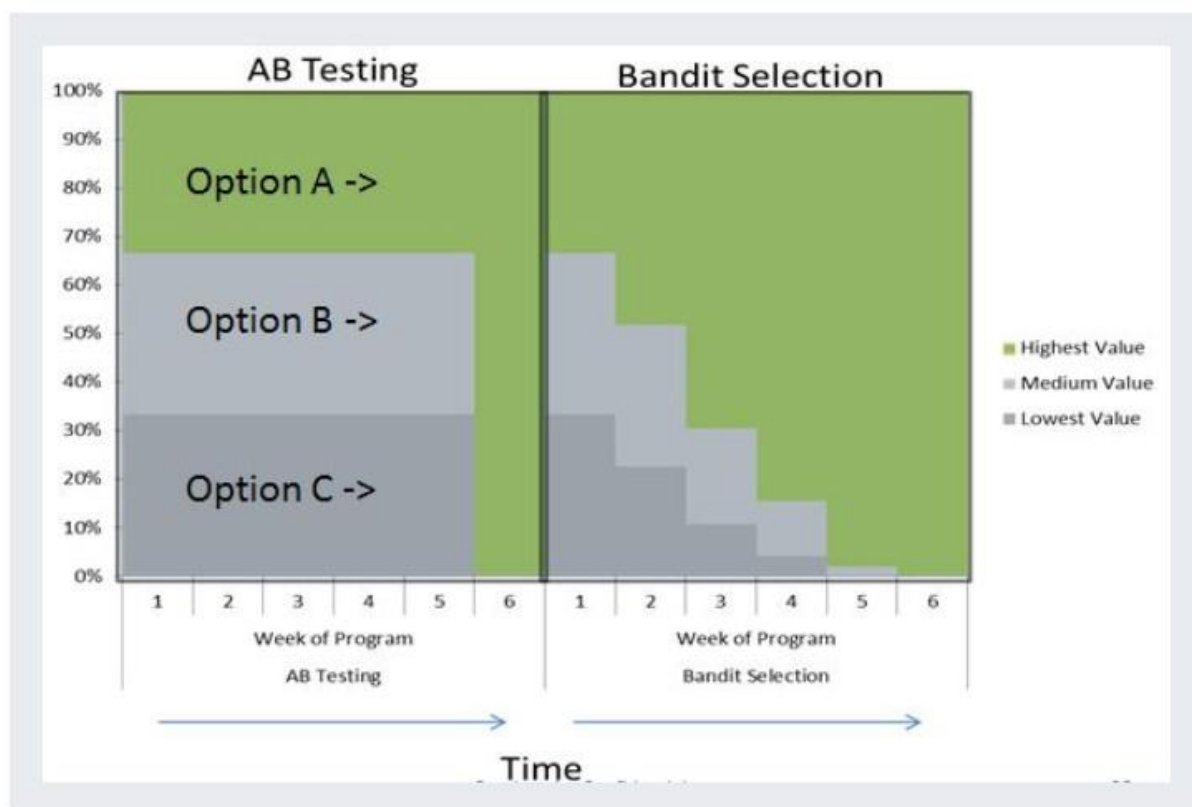


Рисунок 1 – Визуализация алгоритмов «50/50» и «многорукий бандит»

Для минимизации издержек (потерь конверсии на заведомо худших вариантах интерфейса) в системе реализуется байесовский подход к задаче о «многоруких бандитах» с помощью сэмплирования Томпсона.

В рамках данной модели априорное распределение вероятности конверсии для каждого варианта страницы моделируется с помощью бета-распределения, которое является сопряженным для биномиального распределения. Плотность вероятности Бета-распределения $B(\alpha, \beta)$, заданного формулой (1.1):

$$f(x; \alpha; \beta) = \frac{1}{B(\alpha, \beta)} \cdot x^{\alpha-1} \cdot (1-x)^{\beta-1}, \quad (1.1)$$

где $B(\alpha, \beta)$ – бета-распределение,

α – параметр распределения, интерпретируемый как количество успешных конверсий плюс 1,

β – параметр распределения, интерпретируемый как количество просмотров без совершения целевого действия плюс 1.

Алгоритм функционирует в итерационном режиме:

					Лист
ПГУ 1.09.03.01.08.001 ПЗ					9
Изм.	Лист	№ докум.	Подпись	Дата	

1. Для каждого тестируемого варианта страницы генерируется случайное число $\theta \sim B(\alpha, \beta)$.

2. Текущему пользователю демонстрируется тот вариант страницы, для которого сгенерированное значение θ оказалось максимальным.

3. По результатам взаимодействия пользователя со страницей обновляются параметры распределения выбранного варианта:

- если зафиксировано целевое действие (успех): $\alpha_{\text{new}} = \alpha + 1$;
- если целевое действие отсутствует (неудача): $\beta_{\text{new}} = \beta + 1$.

Данный механизм позволяет автоматически и плавно смещать вектор распределения трафика в пользу более эффективных вариантов интерфейса непосредственно в процессе проведения теста, решая дилемму «исследование-эксплуатация» (exploration-exploitation).

После завершения сбора информации система должна проверить данные на статистическую значимость, что позволяет отличить реальное улучшение показателей пользователей от случайных колебаний данных. Для вычисления зависимости используется проверка статистических гипотез.

Формулируются две гипотезы:

- нулевая гипотеза - статистически значимых различий между вариантами А и В нет, а наблюдаемая разница вызвана случайностью;
- альтернативная гипотеза - вариант В действительно эффективнее варианта А.

Для подтверждения гипотезы в разрабатываемом программном комплексе используется критерий согласия Z-критерий. Он позволяет рассчитать показатель p-value — вероятность того, что полученный результат был достигнут случайно. Если $p < 0,05$ (5%), нулевая гипотеза отвергается, и результат признается достоверным.

Применение А/В-тестирования позволяет реализовать концепцию проектирование на основе данных. Вместо интуитивных решений команда разработчиков и маркетологов получает точный инструмент для:

- минимизации рисков при обновлении критических узлов сайта;

					ПГУ 1.09.03.01.08.001 ПЗ	Лист
						10
Изм.	Лист	№ докум.	Подпись	Дата		

- повышения прибыли без увеличения затрат на привлечение трафика;
- объективного разрешения споров о дизайне внутри команды.

Помимо Z-критерия для более точного анализа данных в данном проекте были применены также такие статистические элементы как отношение кликов к просмотрам, процентная конверсия между вариантом В и контрольным вариантом, статистическая мощность.

Отношение кликов к просмотрам (Conversion Rate) высчитывается по формуле (1.2):

$$CR = \frac{\text{Количество кликов}}{\text{Количество просмотров}} \cdot 100\%, \quad (1.2)$$

Данная метрика показывает какое количество людей, зашедших на сайт кликнули на тестируемый элемент. Эти данные помогут нам оценить ценность элемента интерфейса не только в сравнении с другим вариантом, но и в абсолютной величине.

Прирост (Lift) показывает разницу в процентах между вариантами А и В, что делает итоговую статистику более наглядной для аналитика.

Статистическая мощность является количественным показателем данные, обновляющемся в реальном времени. Данный показатель отражает, является ли объем собранных на момент просмотра данных достаточным для формирования достоверного вывода

Сайт представляет из себя набор из огромного числа кнопок, форм, меню и прочих элементов интерфейса. Неправильный выбор элемента, цвета или места расположения может подсознательно отпугнуть пользователя от нажатия на этот элемент, поэтому дизайнеры должны уделять огромное внимание на этот аспект. Однако нет универсального рецепта как сделать сайт идеальным для всех пользователей, ведь предпочтения могут меняться в зависимости от целевой аудитории сайта, ее полового, возрастного или национального состава. И чтобы точно понять, какие элементы будут наиболее удобны для пользователя, применяются А/В-тестирования, когда разным частям аудитории дается для использования различные версии сайта, однако различия эти

					ПГУ 1.09.03.01.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		11

незначительны, они касаются только конкретных элементов. После проведения собирается и анализируется статистика, которая показывает какой вариант больше всего подходит под то, что для него задумал дизайнер.

Проанализируем основные факторы, которые выделяется для себя пользователь приложений для проведения А/В тестов:

- функциональность. Веб-приложение должно иметь следующий функционал: инструменты для изменения элементов интерфейса, разделение трафика для различных пользователей, статистическая проверка гипотез;

- удобство. При реализации приложения необходимо повысить удобство взаимодействия пользователя. Для этого должен быть разработан простой и понятный интерфейс, с использованием базовых паттернов дизайна для быстрой навигации и минимизации ошибок из-за перегруженности или не интуитивности;

- безопасность. Веб-приложение должно использовать самые современные способы шифрования и защиты информации для поддержания надежности хранения и пересылки данных клиента;

- низкая требовательность к ресурсам. Веб-приложение должно сохранять баланс между функциональностью и производительностью, так как оно будет ориентировано преимущественно на малый бизнес и скорость работы может существенно выделить его на фоне конкурентов.

1.1.1 Анализ аналогов

На данный момент существуют несколько сервисов для проведения А/В тестирования. Самые популярные из них:

- google Analytics 4;
- optimizely.

Рассмотрим достоинства и недостатки сервиса Google Analytics 4. Дизайн сервиса представлен на рисунке 2.

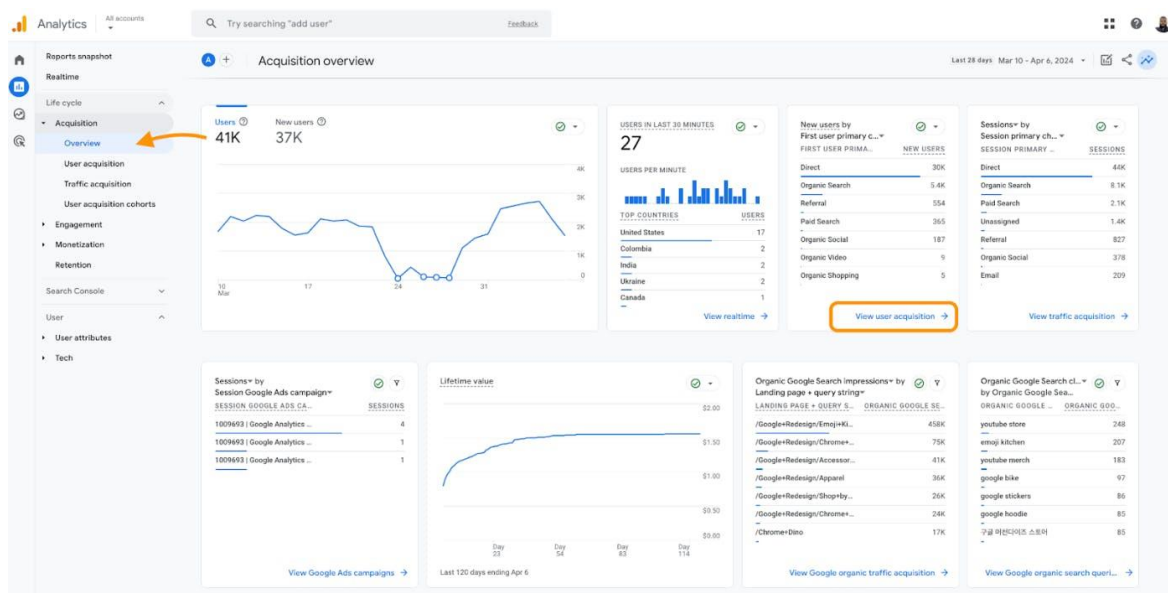


Рисунок 2 – Google Analytics 4

Достоинства:

- глубокая сегментация аудитории;
- визуальный редактор изменений;
- бесплатная базовая версия.

Недостатки:

- прекращение поддержки в 2023 году;
- сложность в настройке серверных тестов;
- передача данных на сторонние серверы;
- высокая требовательность к ресурсам.

Рассмотрим достоинства и недостатки сервиса Optimizely. Дизайн сервиса представлен на рисунке 3.

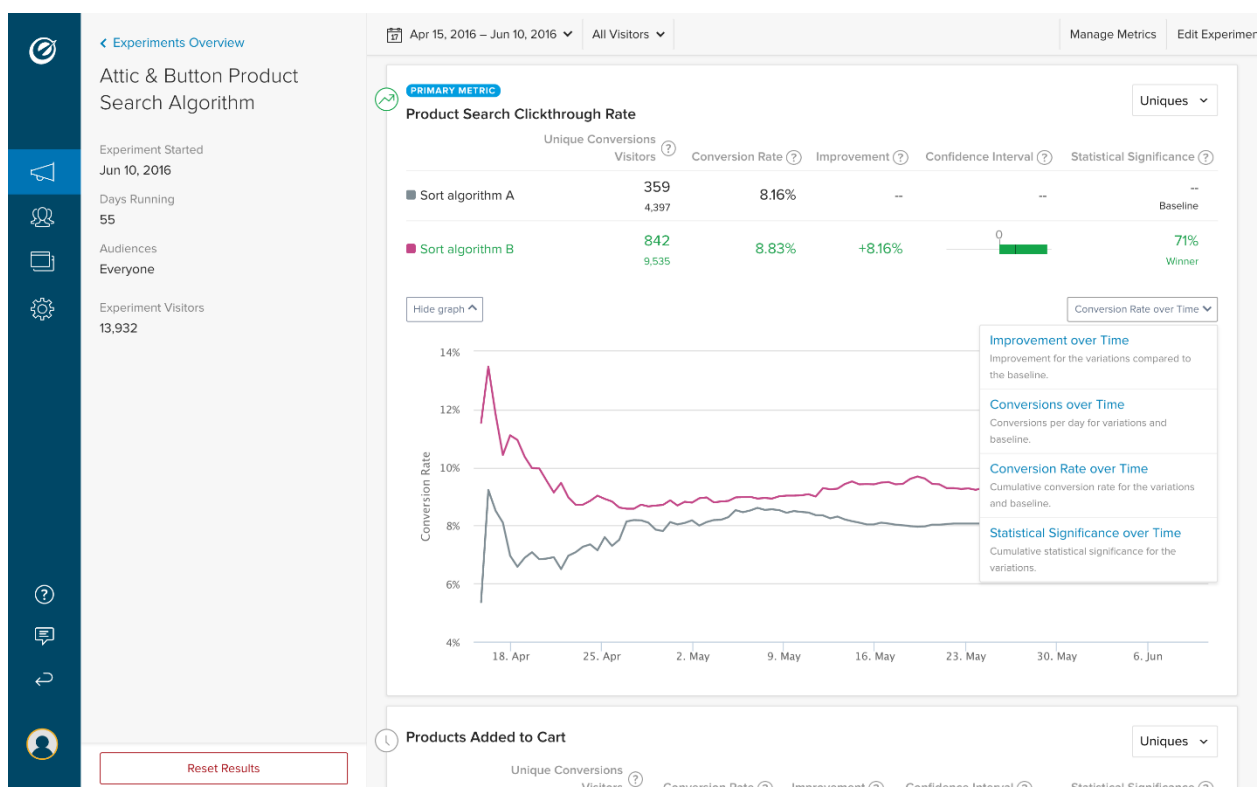


Рисунок 3 – Optimizely

Достоинства:

- высокая скорость работы;
- поддержка Full Stack тестирования;
- мощные статистические движки.

Недостатки:

- высокая стоимость подписки;
- избыточный функционал для малых и средних проектов;
- зависимость от облачной инфраструктуры.

Главным преимуществом моего проекта будет производительность, полный контроль над данными, низкая сложность интеграции, минимальная стоимость и низкая требовательность к ресурсам. Он будет уступать по функционалу конкурентам, но будет проще и дешевле в использовании, что будет особенно важно малому бизнесу, который является целевой аудиторией проекта. Также преимуществом на фоне западных приложений будет доступность постоянная доступность приложения без использования сторонних сервисов.

1.1.2 Постановка задачи

Цель – разработать веб-приложение для проведения А/В тестирования интерфейсов сайтов пользователей, с использованием максимально эффективных, но в то же время легковесных инструментов для получения максимально быстрого и дешевого приложения, обеспечивающего сбор данных о поведении пользователя и расчет статистической значимости в режиме реального времени.

Приложение должно содержать следующие критерии:

- понятный и простой интерфейс для комфортной навигации;
- возможность изменять внешний вид элементов интерфейса;
- приложение должно разделять потоки трафика для различных пользователей;
- приложение должно обрабатывать полученные данные с помощью математических формул и предоставлять пользователю итоги в понятном формате.

Проектирование архитектуры системы

- разработать логическую и физическую модели реляционной базы данных для хранения конфигураций экспериментов и логов событий;
- спроектировать архитектуру REST API, обеспечивающую минимальную задержку при регистрации событий;
- разработать алгоритм распределения трафика между вариациями на основе детерминированного хеширования.

Разработка серверной части

- реализовать на языке Python с использованием фреймворка FastAPI ядро системы, обеспечивающее обработку высоконагруженных запросов;
- разработать модуль статистических вычислений (с использованием библиотеки SciPy) для расчета p-value и определения статистической значимости эксперимента;

– реализовать механизмы сегментации данных по типам устройств (Mobile/Desktop/Tablet) на основе парсинга заголовков User-Agent;

– обеспечить возможность экспорта накопленных данных в формате CSV для дальнейшего анализа.

Разработка клиентской части

– создать административный интерфейс на базе современных веб-технологий (HTML5, CSS3, JavaScript);

– реализовать интерактивную визуализацию метрик (Conversion Rate) с использованием библиотеки Chart.js;

– разработать панель управления жизненным циклом эксперимента (создание, анализ данных, удаление);

– обеспечить адаптивность интерфейса для корректной работы на мониторах и ноутбуках.

Тестирование и отладка

– разработать симулятор пользовательской активности для проверки корректности работы алгоритмов распределения и статистических функций;

– провести отладку механизмов CORS для обеспечения безопасности взаимодействия фронтенда и API.

Нефункциональные требования:

– среднее время ответа сервера на запросы регистрации события (track event) должно составлять не более 200 мс для минимизации влияния на скорость загрузки целевого сайта;

– клиентская часть приложения должна корректно отображаться на экранах с разрешением от 1280x720 пикселей и выше;

– система должна гарантированно функционировать в последних стабильных версиях браузеров: Google Chrome, Mozilla Firefox, Microsoft Edge, Safari, Opera;

– минимальные системные требования к серверу: процессор — 2 ядра, 2.0 ГГц, ОЗУ — 4 ГБ, дисковое пространство — 10 ГБ (SSD), СУБД — SQLite/PostgreSQL.

					ПГУ 1.09.03.01.08.001 ПЗ	Лист
						16
Изм.	Лист	№ докум.	Подпись	Дата		

Ожидаемый результат:

Работоспособное Web-приложение, позволяющее организовать полный цикл А/В-тестирования без использования сторонних облачных сервисов, обеспечивающее полную автономность данных и высокую точность статистических выводов.

На рисунке 4 представлена Use Case диаграмма.

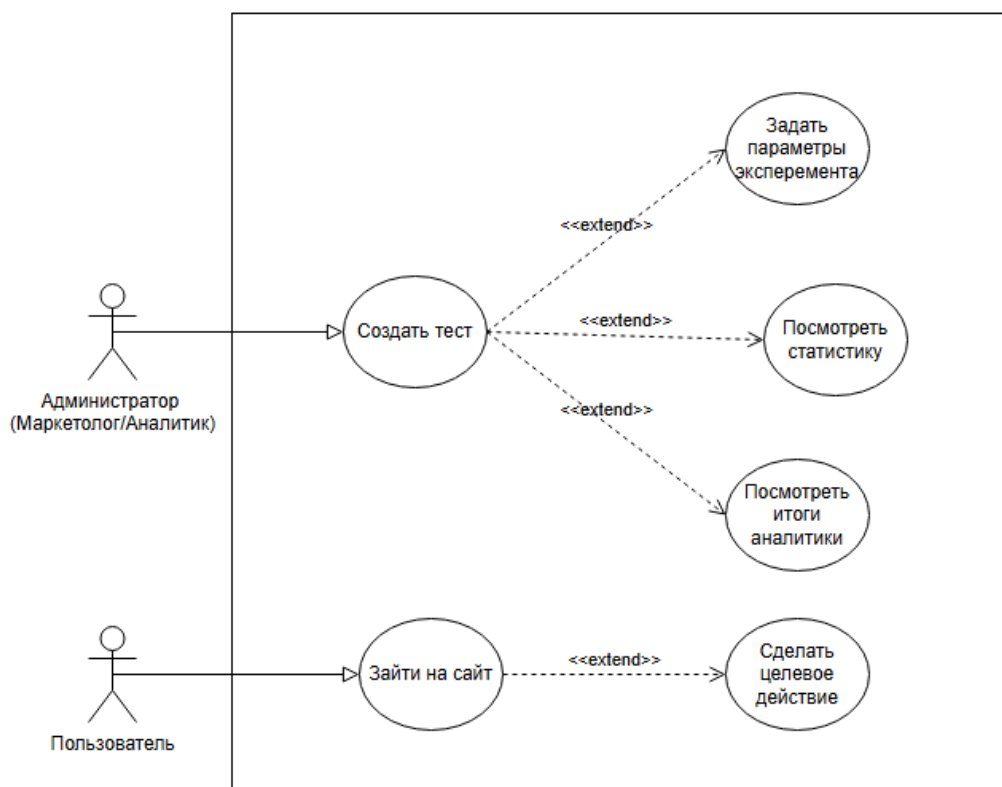


Рисунок 4 – Use Case диаграмма

1.2 Выбор инструментов разработки

Для реализации программной части был произведен анализ современных сред разработки (IDE), главными факторами выбора были: поддержка языка Python, поддержка системы контроля версий (Git), наличие инструментов отладки и производительность.

1.2.1 Выбор IDE для написания кода

IDE – это то приложение, которое будет использоваться чаще всего при написании приложения. Поэтому для повышения удобства разработчика оно должно иметь избыточный функционал для написания и отладки кода и удобный интерфейс, а для повышения производительности быть не требовательной к ресурсам.

1.2.1.1 Visual Studio Code

VS Code – это легковесный редактор исходного кода от компании Microsoft, который с помощью плагинов превращается в полноценную среду разработки. Visual Studio Code поддерживает большинство современных языков программирования и предоставляет многочисленные инструменты для удобства написания, отладки и тестирования кода.

Выделим основным плюсы и минусы данной IDE:

Плюсы:

- высокая производительность;
- возможность расширения функционала с помощью специализированных плагинов;
- внутренний терминал, упрощающий взаимодействие с Git;
- кроссплатформенность.

Минусы:

- необходимость ручной установки плагинов и языков программирования перед началом работы;
- малое количество инструментов для анализа кода.

1.2.1.2 PyCharm

Pycharm – специализированная среда разработки от компании JetBrains, ориентированная на использование языка Python.

Плюсы:

- лучшая система авто дополнения и поиска ошибок в коде Python;

					ПГУ 1.09.03.01.08.001 ПЗ	Лист
						18
Изм.	Лист	№ докум.	Подпись	Дата		

– профессиональная версия имеют глубокую интеграцию с FastAPI, позволяя визуализировать эндпоинты;

– встроенный функционал для просмотра и редактирования таблиц SQLite.

Минусы:

– требовательность к ресурсам;

– весь функционал доступен только в платной версии.

1.2.1.3 Sublime Text

Sublime Text – текстовый редактор, разработанный компанией Sublime HQ, часто используемый для веб-разработки.

Плюсы:

– самая высокая скорость работы среди всех конкурентов;

– минимализм.

Минусы:

– минимальный функционал, практически отсутствуют внутренние средства отладки;

– устаревший интерфейс и настройки.

Я сделал выбор в пользу VS Code. Он обладает всем необходимым функционалом, который можно расширить для выполнения конкретной задачи, понятным интерфейсом, высокой скоростью работы, всем необходимым инструментарием для выполнения отладки и тестирования программы. Данное IDE поддерживает все языки программирования и плагины, которые будут использованы в ходе выполнения курсового проекта, включая написание программы сервера, базы данных, программного интерфейса API и внешнего интерфейса для взаимодействия с пользователем.

1.2.2 Выбор языка программирования

Выбор языка программирования является важнейшим этапом при создании проекта. Основными критериями выбора стали: простота реализации

					ПГУ 1.09.03.01.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		19

REST-интерфейса, скорость разработки и наличие библиотек для работы с базами данных и асинхронными запросами. Самыми подходящими под эти параметры стали языки программирования Python и Java.

1.2.2.1 Java

Java – это объектно-ориентированный компилируемый язык программирования, обладающий широким функционалом для решения разного типа задач.

Плюсы:

- широкая распространенность, что упрощает поиск примеров решения проблемы в случае ее возникновения;
- строгая типизация, что уменьшает количество ошибок при написании кода;
- продвинутые инструменты для работы с многопоточностью;
- масштабируемость.

Минусы:

- сложность синтаксиса;
- высокий порог вхождения;
- медленный цикл разработки из-за длительного времени компиляции и сборки проекта.

1.2.2.2 Python

Python – это высокоуровневый интерпретируемый язык программирования, известный простым синтаксисом и мощной экосистемой библиотек.

Плюсы:

- высокая скорость написания кода;
- современные асинхронные фреймворки такие как FastAPI;
- интерпретируемость, что повышает скорость разработки из-за

отсутствия необходимости компиляции кода,

					ПГУ 1.09.03.01.08.001 ПЗ		Лист
							20
Изм.	Лист	№ докум.	Подпись	Дата			

- большое количество импортируемых библиотек;
- низкий порог вхождения.

Минусы:

- интерпретируемость из-за чего снижается скорость выполнения сложных вычислительных задач;
- узкий встроенный функционал для работы с многопоточностью.

Несмотря на преимущества Java, Python полностью покрывает весь спектр требуемых задач для написания нашего приложения. Также многие инструменты Java будут излишни для нашего проекта, так как приоритет в работе приложения – скорость и простота, а выполнения этих задач Python подойдет лучше всего из-за наличия встроенного фреймворка FastAPI, который обеспечивает производительность, сравнимую с компилируемыми языками программирования. А лаконичность и простота написания кода ускорят работу над программой и дадут больше времени для работы над самим алгоритмом, не отвлекаясь на сложности в реализации.

Для создания интерфейса приложения были выбраны язык разметки HTML5, язык стилей CSS3 и язык программирования JavaScript.

1.2.2.3 HTML5

HTML5 – пятая версия языка разметки HTML. Используется для структурирования и представления содержимого всемирной паутины, а также для создания веб-страниц и браузерных приложений.

Назначение: структурирование интерфейса – создание форм и списков для создания теста таким, чтобы он был понятен и удобен для пользователя.

1.2.2.4 CSS3

CSS3 — третья версия языка CSS. Язык разделён на модули, каждый из которых отвечает за определённую функциональность.

Назначение: настройка внешнего вида кнопок, полей ввода и статусных сообщений, использование технологий Flexbox или Grid для корректного

					ПГУ 1.09.03.01.08.001 ПЗ	Лист
						21
Изм.	Лист	№ докум.	Подпись	Дата		

отображения интерфейса на устройствах с различным разрешением экрана и реализация анимаций.

Данный язык разметки был выбран, так как это самый удобный и современный инструмент верстки интерфейса сайта, отделенный от языка верстки HTML, что упрощает поддержку и модификацию внешнего вида приложения.

1.2.2.5 JavaScript

JavaScript – язык программирования, который позволяет делать веб-страницы интерактивными.

Назначение: реализация клиентской части и скрипта внедрения. Главными преимуществами данного языка программирования являются: нативная поддержка браузерами, асинхронная модель ввода-вывода, а также кроссплатформенность и легковесность.

JavaScript является единственным языком программирования, который выполняется всеми современными веб-браузерами на стороне клиента без установки дополнительных плагинов или сред выполнения. Поскольку разрабатываемая А/Б-платформа должна бесшовно интегрироваться на любые сторонние сайты, использование JS является безальтернативным стандартом.

Кроме того, браузерный JavaScript работает на базе неблокирующей архитектуры, управляемой событиями. Это критически важно для создания скрипта отслеживания. Логирование просмотров и кликов происходит асинхронно, не замораживая интерфейс целевого сайта и не ухудшая пользовательский опыт.

Также механизм работы нашего приложения создания сниппета на языке JS, который пользователь должен вставить в код своего сайта. Использование данного языка гарантирует минимальный размер файла-сниппета, быструю скорость загрузки сети и максимальную производительность даже на мобильных устройствах.

					ПГУ 1.09.03.01.08.001 ПЗ	Лист
						22
Изм.	Лист	№ докум.	Подпись	Дата		

Еще одним плюсом использования языка JS является встроенный фреймворк React.

Использование данной библиотеки обусловлено сложностью интерфейса программы. React позволяет реализовать концепцию компонентно-ориентированной архитектуры. Интерфейс окна аналитики представляет собой множество таблиц и графиков. React позволяет декомпозировать UI на изолированные компоненты, что упрощает их отладку, масштабирование и повторное использование кода.

Также при проведении А/Б-тестов аналитические данные могут обновляться в реальном времени (например, при симуляции или активном потоке пользователей). Вместо ресурсоемкого перерисовывания всей HTML-страницы, React использует механизм сверки через Virtual DOM. Он точно обновляет в браузере только те элементы, которые реально изменились, обеспечивая высокую производительность и мгновенный отклик интерфейса.

Кроме того, использование React открывает доступ к производительным библиотекам, необходимым для аналитической платформы: Chart.js для построения динамических графиков распределения конверсий и интерактивных дашбордов, а также современным UI-китам для ускорения проектирования интерфейса.

Таким образом, все эти достоинства делают язык JS идеальным для реализации клиентской части приложения, обладающей максимальной производительностью и легкой для отладки и поиска багов.

1.2.3 Выбор СУБД

Для реализации веб-приложения был проведен сравнительный анализ систем управления базами данных. Самыми подходящими под задачи проекта оказались PostgreSQL и SQLite.

1.2.3.1 SQLite

					ПГУ 1.09.03.01.08.001 ПЗ	Лист
						23
Изм.	Лист	№ докум.	Подпись	Дата		

SQLite – компактная, встраиваемая реляционная СУБД, которая хранит всю базу данных в одном файле.

Плюсы:

- отсутствие серверной части;
- мобильность – все БД содержится в одном файле, что упрощает миграцию данных между серверами;
- высокая скорость чтения;
- надежность. СУБД поддерживает транзакционную целостность, что критично при операциях удаления секретов после прочтения.

Минусы:

- ограничения параллелизма. При очень большом количестве одновременных записей возможны блокировки файла базы данных;
- отсутствие разграничения прав.

1.2.3.2 PostgreSQL

PostgreSQL – мощная объектно-реляционная СУБД с открытым исходным кодом, ориентированная на большие нагрузки.

- высокая масштабируемость;
- продвинутая система типов данных, включая встроенную поддержку JSONB, что могло бы быть полезно для хранения метаданных секретов.

Минусы:

- сложность администрирования: Требуется установка, настройки кластера и управления пользователями БД, что избыточно для текущих задач проекта;
- высокое потребление ресурсов.

Несмотря на весомые достоинства СУБД SQLite, для реализации проекта была выбрана PostgreSQL. Будущее веб-приложение будет ориентировано на проведение тестирования для большого числа пользователей, что будет приводить к блокировке записи в SQLite. Этот недостаток отсутствует в PostgreSQL, так как она построена на базе архитектуры многоверсионного

					ПГУ 1.09.03.01.08.001 ПЗ	Лист
						24
Изм.	Лист	№ докум.	Подпись	Дата		

управления конкурентным доступом. Это позволяет процессам чтения никогда не блокировать процессы записи, а запись блокирует строки на минимально возможном уровне. Платформа может параллельно обрабатывать огромный поток логов от клиентов, не снижая скорость работы бэкенда.

1.2.4 Выбор библиотек и фреймворков

1.2.4.1 FastAPI

FastAPI – современный высокопроизводительный веб-фреймворк для создания API на языке Python, основанный на стандартных подсказках типов Python.

Назначение: обработка HTTP-запросов, маршрутизация, валидация входящих данных и генерация автоматической документации.

Этот фреймворк был использован в проекте из-за того, что он позволяет эффективно обрабатывать серверу множество одновременных подключений без блокировки потоков. Также благодаря интеграции с библиотекой Pydantic, FastAPI автоматически проверяет корректность JSON-структур, что уменьшает количество ошибок из-за переполнения и неправильного ввода. Кроме того, этот фреймворк упрощает процесс тестирования эндпоинтов благодаря интеграции с сервисом Swagger UI.

1.2.4.2 SQLAlchemy

SQLAlchemy - программная библиотека на языке Python для работы с реляционными СУБД с применением технологии ORM (Object-Relational Mapping). Служит для синхронизации объектов Python и записей реляционной базы данных.

Назначение: организация объектно-реляционного отображения для сущностей «Эксперимент» и «Событие».

Эта библиотека позволяет работать напрямую с объектами Python. Это упрощает реализацию подхода «один-ко-многим» и гарантирует безопасность данных, защищая систему от SQL-инъекций при записи новых логов.

ПГУ 1.09.03.01.08.001 ПЗ					Лист
					25
Изм.	Лист	№ докум.	Подпись	Дата	

1.2.4.3 SciPy

Назначение: выполнение высокоуровневых математических и статистических операций.

Данная библиотека является ключевой для проекта, так как содержит модуль stats, который используется для расчета Z-критерия и определения показателя p-value. Использование этой библиотеки гарантирует точность статистических выводов, на основе которых и принимаются решения о целесообразности использования нового варианта из теста.

1.2.4.4. Charts.js

Назначение: визуализация данных на стороне клиента.

Данная библиотека предоставляет инструменты для визуализации данных с помощью графиков и динамических диаграмм, которые необходимы для увеличения читабельности данных, полученных в ходе проведения тестирования.

1.3 Установка и настройка рабочих сред

1.3.1 Установка IDE Visual Studio Code

Серверная и пользовательские части разрабатываются на языках программирования Python, HTML, JavaScript. Для установки интерпретатора необходимо загрузить установочный файл с официального сайта code.visualstudio.com. Процесс установки IDE на рабочий компьютер представлен на рисунках 5, 6, 7.

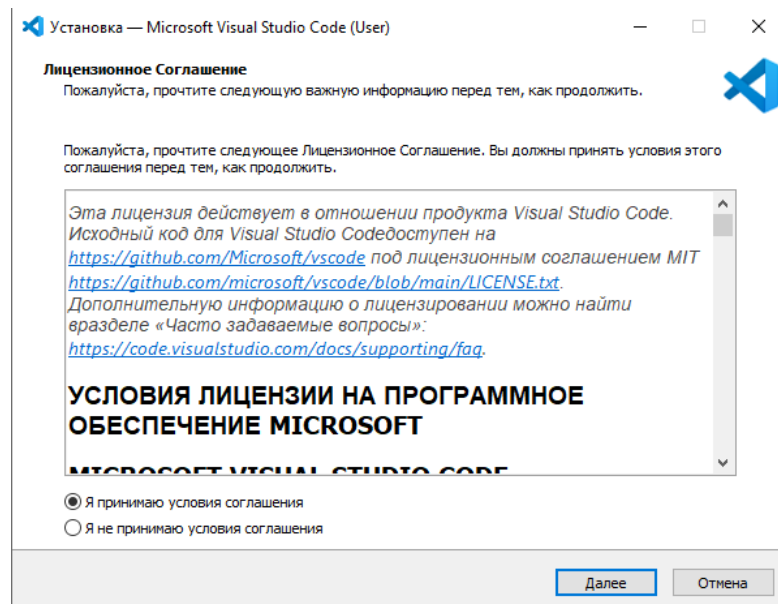


Рисунок 5 – Установка IDE

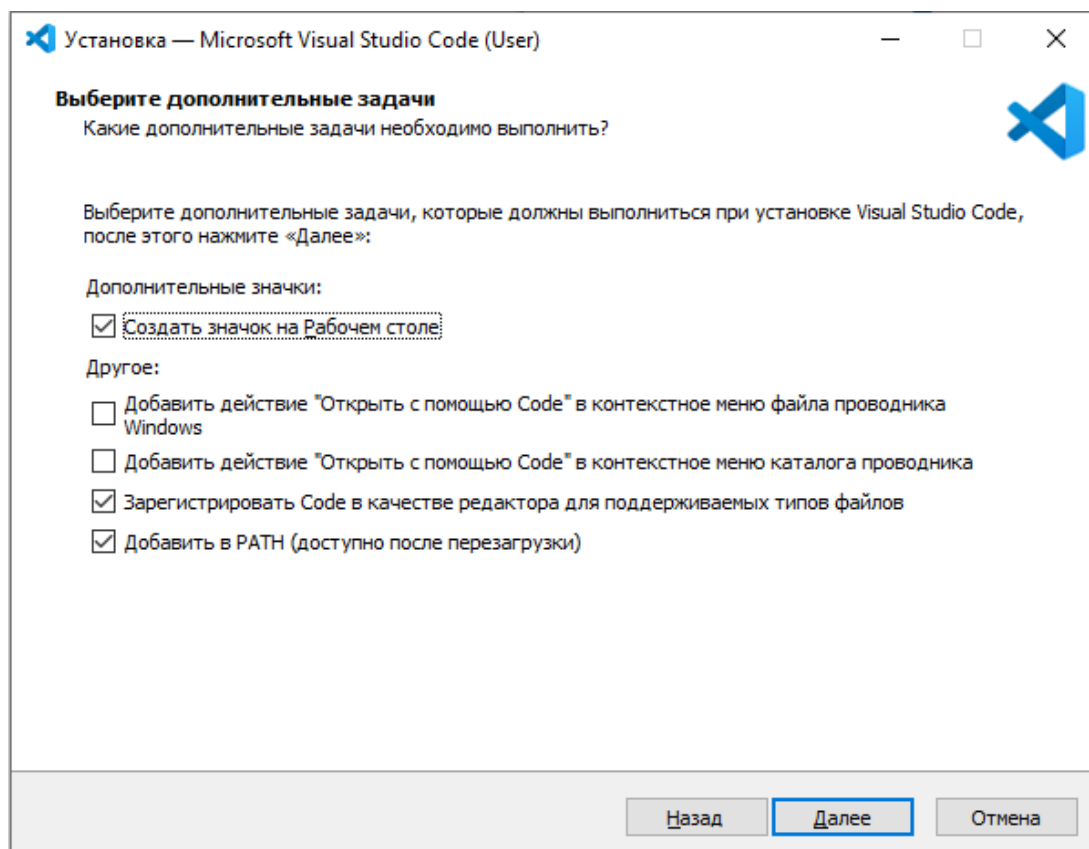


Рисунок 6 – Установка IDE

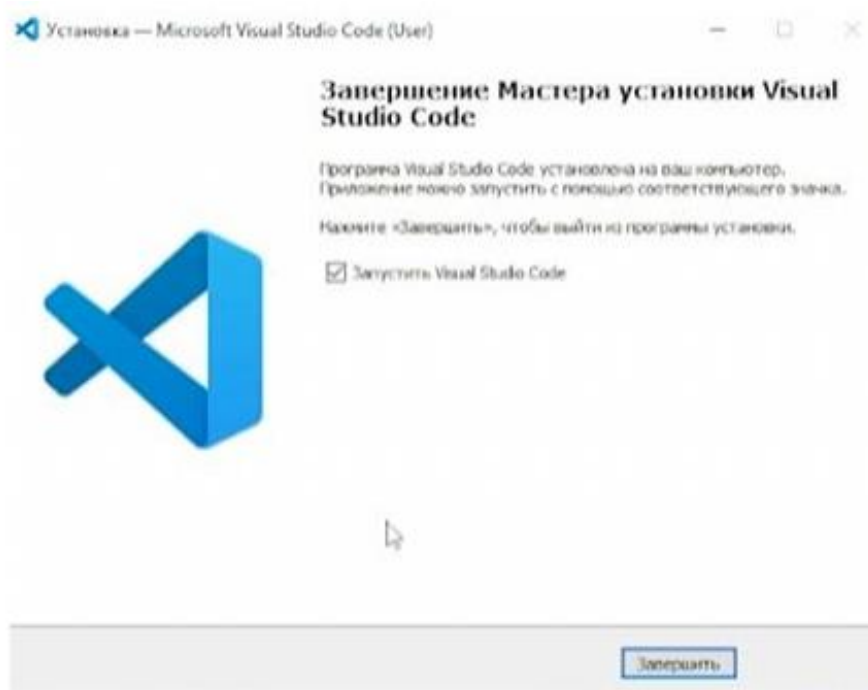


Рисунок 7 – Установка IDE

Для работы с библиотеками, которые были прописаны в пункте выше необходимо загрузить их на рабочий компьютер с помощью командной строки.

Для этого необходимо ввести команды:

- pip install fastapi uvicorn;
- pip install sqlalchemy;
- pip install scipy.

Процесс загрузки библиотек на рабочий компьютер показан на рисунках 8, 9.

```
PS C:\Users\79374> pip install scipy
Requirement already satisfied: scipy in c:\users\79374\appdata\local\packages\pythonsoftwarefoundation.python.3.10_qbz5n2kfra8p0\loc
alcache\local-packages\python310\site-packages (1.15.3)
Requirement already satisfied: numpy<2.5,>=1.23.5 in c:\users\79374\appdata\local\packages\pythonsoftwarefoundation.python.3.10_qbz5
n2kfra8p0\localcache\local-packages\python310\site-packages (from scipy) (2.2.6)

[notice] A new release of pip is available: 23.0.1 -> 26.0.1
[notice] To update, run: C:\Users\79374\AppData\Local\Microsoft\WindowsApps\PythonSoftwareFoundation.Python.3.10_qbz5n2kfra8p0\pytho
n.exe -m pip install --upgrade pip
PS C:\Users\79374> pip install sqlalchemy
Requirement already satisfied: sqlalchemy in c:\users\79374\appdata\local\packages\pythonsoftwarefoundation.python.3.10_qbz5n2kfra8p
0\localcache\local-packages\python310\site-packages (2.0.46)
Requirement already satisfied: typing-extensions>=4.6.0 in c:\users\79374\appdata\local\packages\pythonsoftwarefoundation.python.3.1
0_qbz5n2kfra8p0\localcache\local-packages\python310\site-packages (from sqlalchemy) (4.15.0)
Requirement already satisfied: greenlet>=1 in c:\users\79374\appdata\local\packages\pythonsoftwarefoundation.python.3.10_qbz5n2kfra8
p0\localcache\local-packages\python310\site-packages (from sqlalchemy) (3.3.1)

[notice] A new release of pip is available: 23.0.1 -> 26.0.1
[notice] To update, run: C:\Users\79374\AppData\Local\Microsoft\WindowsApps\PythonSoftwareFoundation.Python.3.10_qbz5n2kfra8p0\pytho
n.exe -m pip install --upgrade pip
PS C:\Users\79374>
```

Рисунок 8 – Загрузка библиотек Sqlalchemy и SciPy

					ПГУ 1.09.03.01.08.001 ПЗ	Лист
						28
Изм.	Лист	№ докум.	Подпись	Дата		

```

PS C:\Users\79374> pip install fastapi uvicorn
Requirement already satisfied: fastapi in c:\users\79374\appdata\local\packages\pythonsoftwarefoundation.python.3.10_qbz5n2kfra8p0\l
ocalcache\local-packages\python310\site-packages (0.128.0)
Requirement already satisfied: uvicorn in c:\users\79374\appdata\local\packages\pythonsoftwarefoundation.python.3.10_qbz5n2kfra8p0\l
ocalcache\local-packages\python310\site-packages (0.40.0)
Requirement already satisfied: starlette<0.51.0,>=0.40.0 in c:\users\79374\appdata\local\packages\pythonsoftwarefoundation.python.3.
10_qbz5n2kfra8p0\localcache\local-packages\python310\site-packages (from fastapi) (0.50.0)
Requirement already satisfied: annotated-doc<=0.0.2 in c:\users\79374\appdata\local\packages\pythonsoftwarefoundation.python.3.10_qb
z5n2kfra8p0\localcache\local-packages\python310\site-packages (from fastapi) (0.0.4)
Requirement already satisfied: pydantic<=2.7.0 in c:\users\79374\appdata\local\packages\pythonsoftwarefoundation.python.3.10_qbz5n2k
fra8p0\localcache\local-packages\python310\site-packages (from fastapi) (2.12.5)
Requirement already satisfied: typing-extensions<=4.8.0 in c:\users\79374\appdata\local\packages\pythonsoftwarefoundation.python.3.1
0_qbz5n2kfra8p0\localcache\local-packages\python310\site-packages (from fastapi) (4.15.0)
Requirement already satisfied: h11<=0.8 in c:\users\79374\appdata\local\packages\pythonsoftwarefoundation.python.3.10_qbz5n2kfra8p0\
localcache\local-packages\python310\site-packages (from uvicorn) (0.16.0)
Requirement already satisfied: click<=7.0 in c:\users\79374\appdata\local\packages\pythonsoftwarefoundation.python.3.10_qbz5n2kfra8p
0\localcache\local-packages\python310\site-packages (from uvicorn) (8.3.1)
Requirement already satisfied: colorama in c:\users\79374\appdata\local\packages\pythonsoftwarefoundation.python.3.10_qbz5n2kfra8p0\
localcache\local-packages\python310\site-packages (from click<=7.0->uvicorn) (0.4.6)
Requirement already satisfied: typing-inspection<=0.4.2 in c:\users\79374\appdata\local\packages\pythonsoftwarefoundation.python.3.1
0_qbz5n2kfra8p0\localcache\local-packages\python310\site-packages (from pydantic<=2.7.0->fastapi) (0.4.2)
Requirement already satisfied: pydantic-core==2.41.5 in c:\users\79374\appdata\local\packages\pythonsoftwarefoundation.python.3.10_q
bz5n2kfra8p0\localcache\local-packages\python310\site-packages (from pydantic<=2.7.0->fastapi) (2.41.5)
Requirement already satisfied: annotated-types<=0.6.0 in c:\users\79374\appdata\local\packages\pythonsoftwarefoundation.python.3.10_
qbz5n2kfra8p0\localcache\local-packages\python310\site-packages (from pydantic<=2.7.0->fastapi) (0.7.0)
Requirement already satisfied: anyio<5,>=3.6.2 in c:\users\79374\appdata\local\packages\pythonsoftwarefoundation.python.3.10_qbz5n2k
fra8p0\localcache\local-packages\python310\site-packages (from starlette<0.51.0,>=0.40.0->fastapi) (4.12.1)
Requirement already satisfied: exceptiongroup<=1.0.2 in c:\users\79374\appdata\local\packages\pythonsoftwarefoundation.python.3.10_q
bz5n2kfra8p0\localcache\local-packages\python310\site-packages (from anyio<5,>=3.6.2->starlette<0.51.0,>=0.40.0->fastapi) (1.3.1)
Requirement already satisfied: idna<=2.8 in c:\users\79374\appdata\local\packages\pythonsoftwarefoundation.python.3.10_qbz5n2kfra8p0
\localcache\local-packages\python310\site-packages (from anyio<5,>=3.6.2->starlette<0.51.0,>=0.40.0->fastapi) (3.11)

[notice] A new release of pip is available: 23.0.1 -> 26.0.1
[notice] To update, run: C:\Users\79374\AppData\Local\Microsoft\WindowsApps\PythonSoftwareFoundation.Python.3.10_qbz5n2kfra8p0\pytho
n.exe -m pip install --upgrade pip

```

Рисунок 9 – Загрузка библиотеки Fastapi Uvicorn

1.3.2 Установка СУБД PgAdmin

Для наиболее удобного использования PostgerSQL необходимо установить СУБД PgAdmin. Оно предоставляет весь необходимый функционал и инструменты для визуализации данных. Процесс установки показан на рисунках 10, 11, 12, 13 ниже.

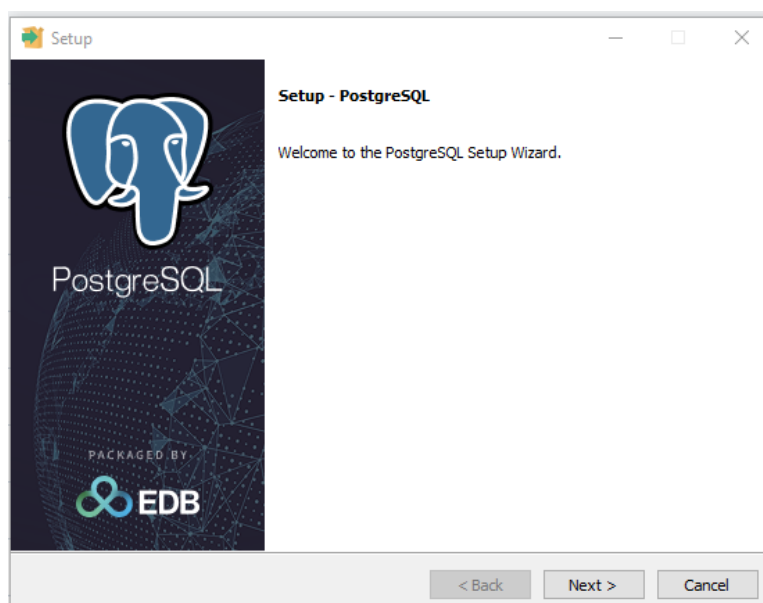


Рисунок 10 – Установка PgAdmin

					Лист
					29
Изм.	Лист	№ докум.	Подпись	Дата	

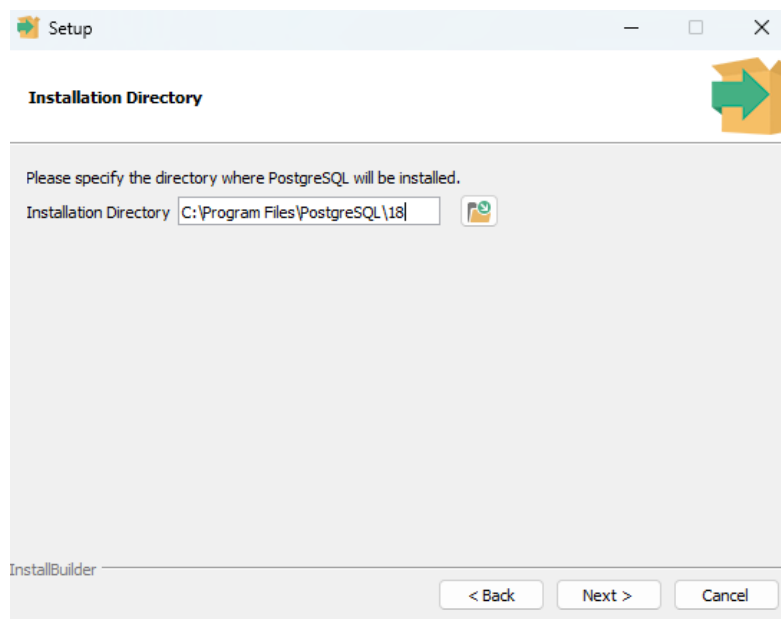


Рисунок 11 – Установка PgAdmin

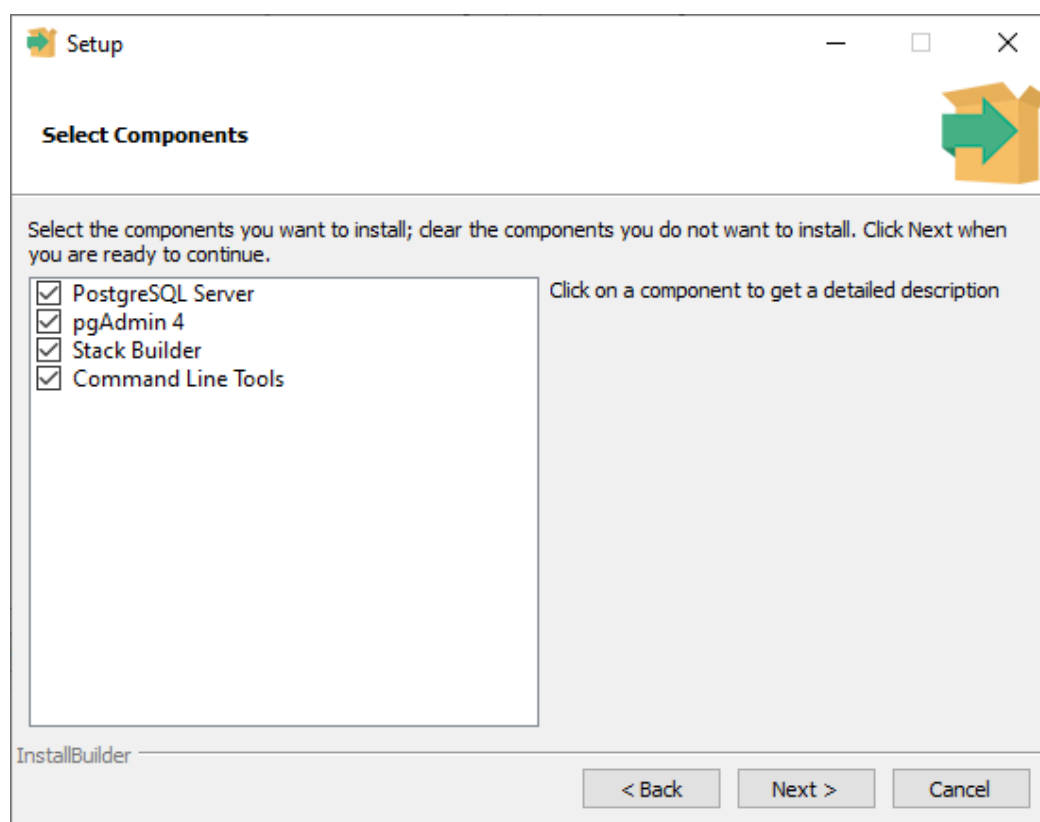


Рисунок 12 – Установка PgAdmin

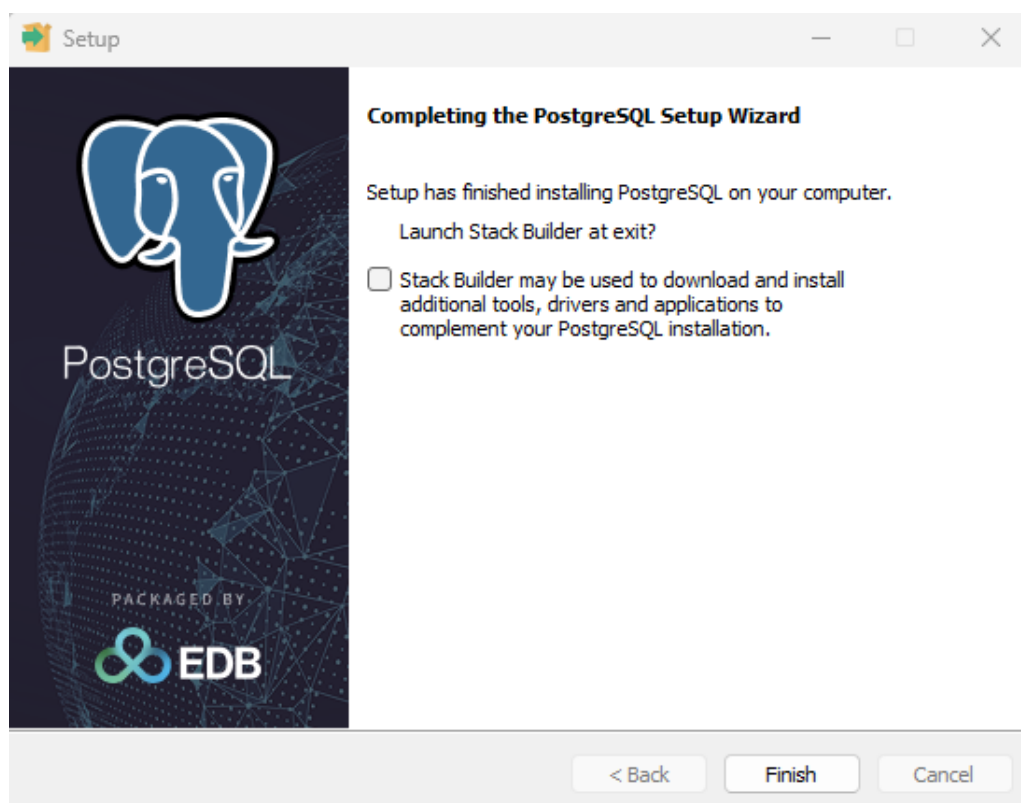


Рисунок 13 – Установка PgAdmin

2 Реализация приложения

Приложение разделено на четыре части: клиент, сервер, БД, «движок». Связь между клиентом и сервером реализуется через FastAPI.

Процесс взаимодействия частей программы представлен на рисунке 14.

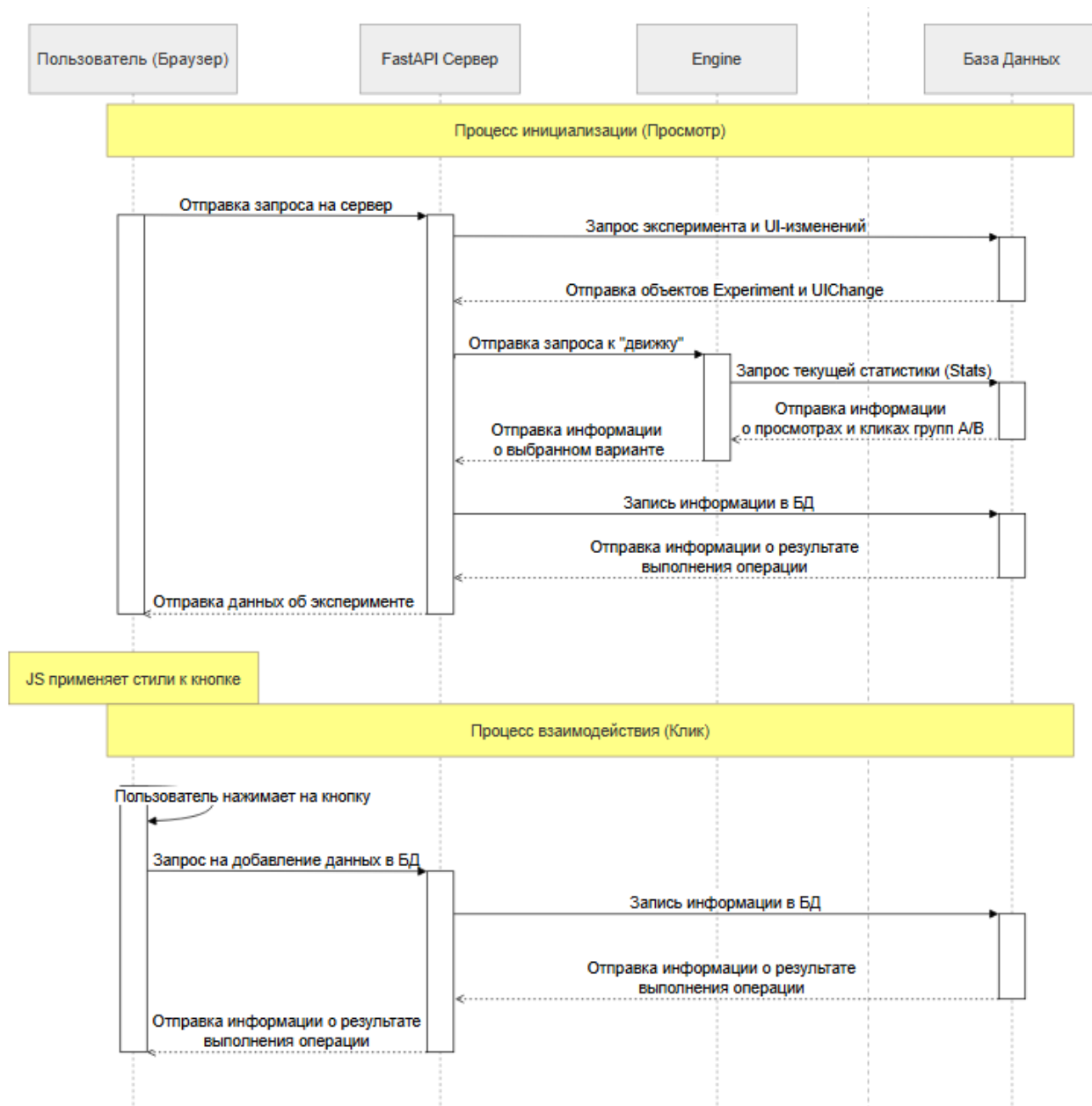


Рисунок 14 – Диаграмма последовательности

Приложение состоит из пяти модулей: analytics.py, engine.py, main.py, models.py, simulator.py.

Схема взаимодействия модулей показана на рисунке 15.

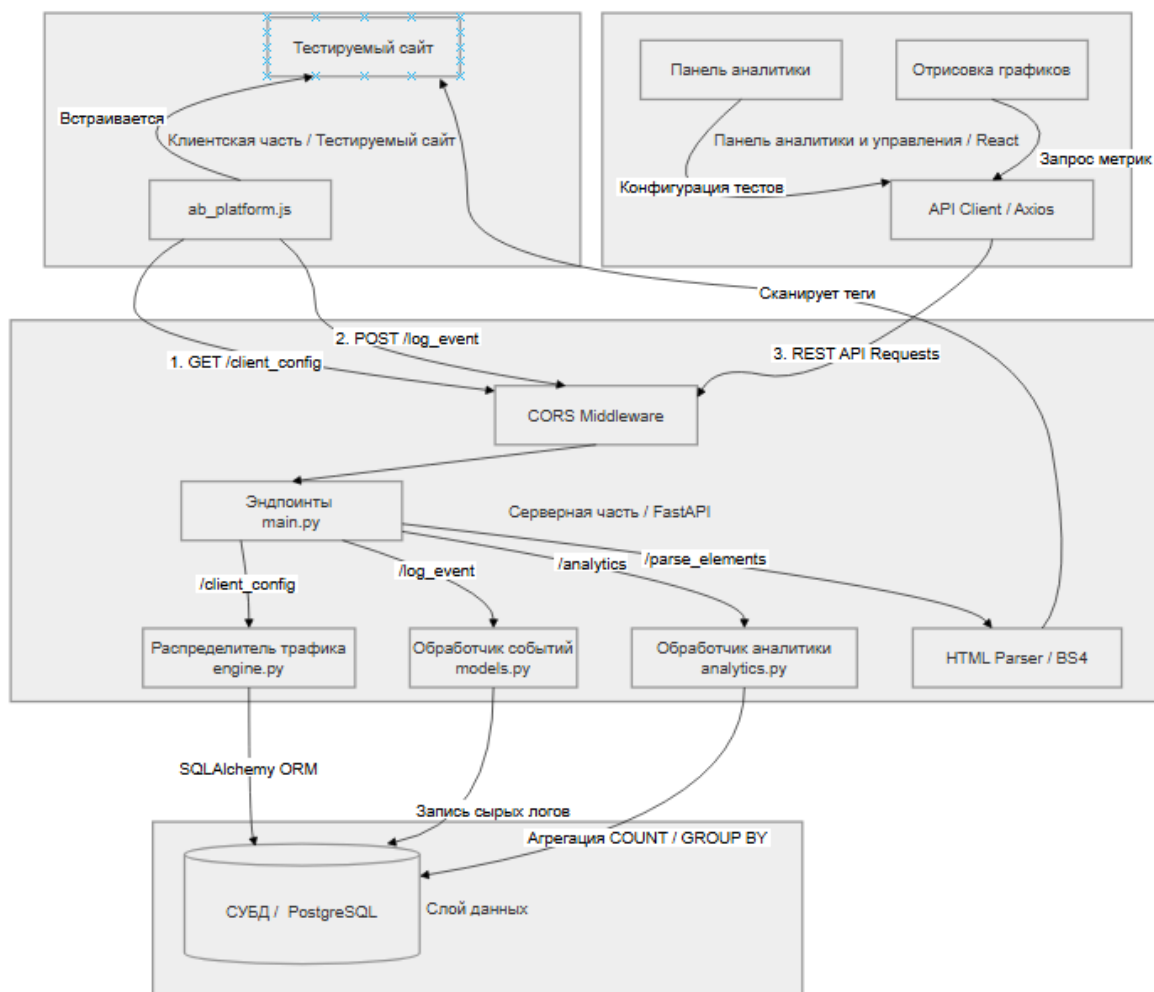


Рисунок 15 – Диаграмма компонентов

Рассмотрим каждый из них подробнее.

2.1 Разработка алгоритма распределения трафика

Данный модуль является основой серверной части приложения. Его назначение реализация сегментирования трафика и обеспечение безопасности данных.

В программе реализовано 3 режима работы: разделение «50/50», алгоритм Томпсона, многорукий бандит.

Жесткое разделение реализуется с помощью механизма хеширования.

Реализация хеширования производится с помощью библиотеки hashlib.

					ПГУ 1.09.03.01.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		33

MD5 – алгоритм хеширования, разработанный профессором Рональдом Л. Ривестом из Массачусетского технологического института в 1991 году. Предназначен для создания контрольных сумм или «отпечатков» сообщения произвольной длины и последующей проверки их подлинности. Главной особенностью алгоритма, которые мы будем использовать для однозначной идентификации, это детерминированность, то есть одинаковые входные данные всегда дадут одинаковый хеш. Также этот алгоритм оптимизирован для повышения производительности, что позволит увеличить безопасность системы, не уменьшая скорость ее работы. Также для повышения производительности было принято решение хешировать только первые 8 бит ключа, что также обеспечит высокую точность распределения и скорость преобразования в целое число.

У алгоритма MD5 есть минусы: возможность появления коллизий и слабая защита от дешифрования. Однако наше приложение рассчитано на малую целевую аудиторию, что позволяет пренебречь вероятностью возникновения коллизий. Кроме того, добавление «соли» в виде названия эксперимента к ключу позволяет полностью пренебречь вероятностью появления коллизий. Также мы не будем собирать конфиденциальные данные пользователей, поэтому даже в случае перехвата и дешифрации хеш-функции, злоумышленники не смогут получить доступ к персональным данным пользователей, а лишь увидят данные тестирования. Кроме того, использование более стойких к взлому алгоритмов привело бы к падению производительности, поэтому использование алгоритма MD5 является оптимальным с точки зрения баланса между безопасностью и скоростью работы приложения. Код реализации данного режима представлен ниже.

После применения к ключу алгоритма хеширования выполняется перевод данных в числовой формат в диапазоне [0;99], которые будут интерпретироваться системой, как процентная доля трафика. Код алгоритма представлен ниже.

```
if exp.algorithm == "classic":
```

					ПГУ 1.09.03.01.08.001 ПЗ	Лист
						34
Изм.	Лист	№ докум.	Подпись	Дата		

```

import hashlib

hash_val = int(hashlib.md5((user_id +
exp_name).encode('utf-8')).hexdigest(), 16)

return "A" if (hash_val % 100) < 50 else "B"

```

Также в данном модуле реализовано распределение трафика по алгоритму Томпсона. Алгоритм анализирует поступающий трафик в реальном времени и меняет распределение так, чтобы большая часть пользователей получала вариант, который лидирует по соотношению кликов к просмотрам. Таким образом, система будет не только проводить исследование, но и адаптировать сайт для повышения удобства пользователей, если станет понятно, что один из вариантов сильно проигрывает другому по уровню отклика от пользователей. Код алгоритма Томпсона представлен ниже.

```

if exp.algorithm == "thompson":
    alpha_a, beta_a = clicks_a + 1, (views_a - clicks_a) + 1
    alpha_b, beta_b = clicks_b + 1, (views_b - clicks_b) +
sample_a = random.betavariate(alpha_a, beta_a)
sample_b = random.betavariate(alpha_b, beta_b)

return "B" if sample_b > sample_a else "A"

```

Также в этом алгоритме используется переменная epsilon для реализации режима работы «многорукий бандит». Переменная отвечает за количество людей, для которых проводится тест. То есть если мы хотим провести исследование не на всех пользователях, а только на небольшой целевой аудитории, необходимо выбрать этот режим работы. По умолчанию, процент пользователей-участников эксперимента равен 10. Так для 90 процентов аудитории изменений не будет или они все увидят новый вариант, а для 10 будет проведен тест с распределением 50/50. Код режима представлен ниже.

```

if exp.algorithm == "epsilon_greedy" or True:
    epsilon = getattr(exp, "epsilon", 0.1)
    if random.random() < epsilon:
        return "A" if random.random() < 0.5 else "B"
    cr_a = clicks_a / views_a if views_a > 0 else 0

```

ПГУ 1.09.03.01.08.001 ПЗ					Лист
					35
Изм.	Лист	№ докум.	Подпись	Дата	

```

cr_b = clicks_b / views_b if views_b > 0 else 0
if cr_a == cr_b:
    return "A" if random.random() < 0.5 else "B"
return "B" if cr_b > cr_a else "A"

```

Полный код модуля engine.py представлен в приложении Б.

2.2 Проектирование классов

Данный модуль представляет собой архитектурное описание данных приложения. Он используется, чтобы перенести логическую модель БД в программный код.

Данный модуль написан с использованием функционала из библиотеки SQLAlchemy. Это позволит нам автоматизировать создание таблиц и гарантировать типизацию данных на уровне приложения.

Модуль состоит из классов: Experiment, Event, UIChange, Site.

Класс Experiment описывает сущность «эксперимент» из модели БД и описывает структуру таблицы БД, в которой будут храниться данные всех экспериментов.

Таблица содержит ID эксперимента, сайта, пользователя. Также она содержит поля всех статистических данных, собираемых в ходе эксперимента: статус (создан, завершен), процент трафика, пускаемого в тест, режим работы (многорукий бандит, разделение 50/50), количество пользователей, которое нужно набрать для включения режима многорукого бандита, дату создания эксперимента, ожидаемая конверсия, задаваемая пользователем.

Класс Event используется для логирования всех действий пользователей в рамках проводимых тестов. Поля класса полностью повторяют схему БД, описанную в пункте «Проектирование БД».

Использование библиотеки SQLAlchemy и ее встроенных функций таких как datetime.datetime.utcnow позволяет повысить эффективность работы приложения и читаемость кода.

Класс UIChange используется для реализации динамической подмены интерфейса. В нем хранятся данные об исходном и новом состоянии страницы, CSS-селектор, определяющий новый элемент, тип изменения и внешний ключ, который привязывает конкретное изменение к определенному тесту, а также тип и текст элемента.

Класс Site является моделью, которая отвечает за хранение всех данных о сайте. Данная структура включает в себя поля ID сайта, название сайта, домен, API ключ для взаимодействия с клиентским JS.

Схема взаимодействия классов показана на рисунке 16.

Полный код модуля models.py представлен в приложении Б

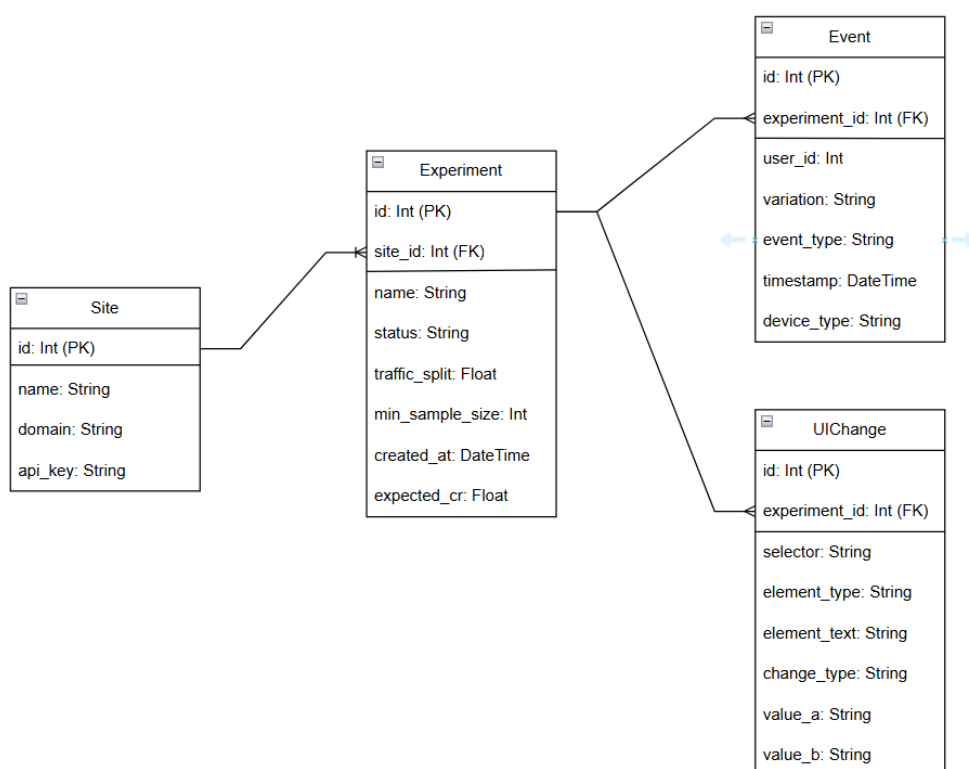


Рисунок 16 – Диаграмма классов

2.3 Разработка алгоритма аналитики данных

Модуль analytics.py отвечает за математическую обработку данных, собранных во время проведения тестирования, и проверку статистических гипотез. Основная задача определить является ли наблюдаемая разница в

выборе вариантов пользователями результатом внесенных дизайнером изменений или это лишь статистическая аномалия.

При расчетах используются четыре основных статистических показателя: коэффициент конверсии, доверительный интервал, Z-критерий, статистическая значимость (p-value). Коэффициент конверсии показывает отношение целевых действий, то есть кликов, к общему числу показов. Он рассчитывается отдельно для каждой группы. После расчета коэффициента конверсии программа рассчитывает стандартную ошибку, которая показывает уровень случайной погрешности. Математически данное вычисление завязано на формуле (2.1) биномиального распределения:

$$SE = \sqrt{\frac{p \cdot (1 - p)}{n}}, \quad (2.1)$$

где p – конверсия,

n – размер выборки.

То есть, чем больше выборка, тем меньше погрешность.

Далее на основе этой переменной мы вычисляем доверительный интервал – диапазон, в котором с вероятностью 95% находится истинная конверсия нашего сайта. Итоговое значение конверсии вычисляется по формуле (2.2):

$$CI = CR \pm 1.96 \cdot SE, \quad (2.2)$$

где CR – конверсия,

SE – погрешность выборки.

Далее мы смотрим на разницу между вариантами А и В, для этого рассчитываем Z-критерий. Он определяется формулой (2.3):

$$Z - \text{критерий} = \frac{CR_b - CR_a}{\sqrt{SE_a^2 + SE_b^2}}, \quad (2.3)$$

где CR – конверсия,

SE – погрешность.

Погрешности обоих вариантов складываются для того, чтобы оценить насколько разница между конверсия отлична от нуля. Чем больше значение Z-критерия, тем больше В отличается от варианта А.

Для окончательной проверки нулевой гипотезы (об отсутствии реальных различий в конверсиях между сегментами А и В) вычисляется показатель p -value. Математически данный показатель представляет собой вероятность получить наблюдаемое значение статистики критерия при условии истинности нулевой гипотезы.

По закону больших чисел разница между конверсиями в группах А и В стремится к стандартному нормальному распределению (кривой Гаусса). Центр этой кривой равен нулю, что означает, что разницы между кривыми нет. Вычисление Z-метрики — это вычисление конкретной точки на этой кривой. Площадь под кривой распределения – это вероятность. Поскольку общая площадь под графиком равна 1, то площадь хвоста распределения (области справа от точки) показывается вероятность получить значение 2 или больше для Z-критерия при одинаковых А и В из-за случайного шума.

Для того чтобы учесть ситуацию, когда В больше А, функция берется по модулю. Для того чтобы учесть аномалии в две стороны, мы умножаем получившееся значение на 2, получая сумму обоих экстремальных хвостов (правого и левого).

```
def calculate_ab_results(count_a, clicks_a, count_b, clicks_b):
    cr_a = clicks_a / count_a
    cr_b = clicks_b / count_b
    se_a = math.sqrt(cr_a * (1 - cr_a) / count_a)
    se_b = math.sqrt(cr_b * (1 - cr_b) / count_b)

    ci_a = [max(0, cr_a - 1.96 * se_a), min(1, cr_a + 1.96 *
se_a)]
    ci_b = [max(0, cr_b - 1.96 * se_b), min(1, cr_b + 1.96 *
se_b)]
    combined_se = math.sqrt(se_a**2 + se_b**2)
    z_score = (cr_b - cr_a) / combined_se if combined_se > 0 else
0
    p_value = stats.norm.sf(abs(z_score)) * 2
    return {
```

```

"cr_a": round(cr_a * 100, 2),
"cr_b": round(cr_b * 100, 2),
"ci_a": [round(x * 100, 2) for x in ci_a],
"ci_b": [round(x * 100, 2) for x in ci_b],
"p_value": round(p_value, 4),
"is_significant": p_value < 0.05,
"lift": round(((cr_b - cr_a) / cr_a * 100), 2) if cr_a >
0 else 0
}

```

Полный код модуля analytics.py представлен в приложении Б.

2.4 Проектирование эндпоинтов

Данный модуль используется для взаимодействия с API, координирования работы остальных модулей с БД.

Модуль реализован с использованием функций из библиотеки FastAPI, которые обеспечивают маршрутизацию HTTP-запросов, интеграцию с БД и обработку метаданных пользователей.

Все современные браузеры работают на технологии CORS, которая позволяет предоставлять веб-страницам доступ к объектам сторонних интернет-ресурсов. Поэтому в проекте есть блок кода, отвечающий за настройку CORS, так как если он настроен неправильно, браузер будет выдавать ошибку.

Настройка CORS заключается в разрешении использования HTTP-методов для фронтенда. В проекте разрешены методы GET, POST, DELETE, PUT, OPTIONS.

Логика API разделена на четыре ключевые группы: экспериментальный модуль, аналитический модуль, административный блок, модуль экспорта, модуль парсинга. У каждой группы своя логика работы и эндпоинты. Рассмотрим подробнее каждую группу.

Группа эндпоинтов аналитического модуля «/analytics» реализует логику агрегации и расчета данных.

					ПГУ 1.09.03.01.08.001 ПЗ	Лист
						40
Изм.	Лист	№ докум.	Подпись	Дата		

Модуль поддерживает фильтрацию по сегментам («tablet», «mobile», «desktop»). Здесь же происходит расчет статистической значимости, p-value, конверсии относительно исходного варианта. Также данная функция определяет тип устройства по метаданным. После расчетов данные структурируются и возвращаются фронтэнду, вызвавшему эту функцию.

В случае не нахождения сервером данных об эксперименте в БД пользователь получит ошибку 404.

В случае успешного вызова метода сервер обработает все данные по эксперименту, лежащие в БД, и вернет клиенту название эксперимента, статус-код, ожидаемую конверсию, режим работы алгоритма, тип устройства пользователя, статистику по кликам, просмотрам, конверсии по вариантам А и В, а также JSON-переменную со статистикой – p-value, bool-переменной, показывающей является ли статистика значимой, прирост конверсии.

Эндпоинты административной группы находится по адресу «/experiments».

Данная группа эндпоинтов описывает список CRUD-операций, применяемых к БД со списком тестов.

Когда сервер получает POST-запрос, он проверяет корректность вызова и в негативном случае возвращает ошибку 404. В случае корректного вызова метод создает новую запись эксперимента в БД и возвращает клиенту сообщение об успешно выполненной операции.

Когда сервер получает GET-запрос, он возвращает клиенту весь список экспериментов, лежащих в БД.

Когда сервер получает DELETE-запрос, он проверяет корректность вызова и в негативном случае возвращает ошибку 404. В случае успешного вызова сервер удаляет запись в БД с тем именем, которое он получил от клиента, и возвращает клиенту ответ об успешном вызове.

Когда сервер получает POST-запрос на очистку данных в БД, он проверяет корректность вызова и в негативном случае возвращает ошибку 404.

В случае удачного вызова сервер проводит полную очистку всех данных в БД и возвращает клиенту ответ об успешном вызове.

Когда сервер получает PUT-запрос на обновление данных, он устанавливает статус эксперимента в значение «Завершено». В случае негативного вызова он возвращает ошибку 404.

Группа эндпоинтов модуля экспорта находится по адресу «/export_csv»

Данный модуль отвечает за экспорт данных в формате CSV. Для этого сервер формирует в памяти устройства пользователя CSV-файл со всеми необходимыми полями и преобразует все данные из БД в строковый формат в случае успешного вызова возвращается пользователю через встроенную функцию потоковой передачи «StreamingResponse», что позволяет не терять в производительности даже при большом размере передаваемого файла. В случае неудачного вызова клиент получит ответ с ошибкой 404.

Эндпоинт «/parse_elements» используется для парсинга элементов с сайта пользователя.

Функция разбивает элементы HTML-кода сайта по тегам. Далее алгоритм создает пары «url-element» по принципу: Если у тега есть id, создается селектор #идентификатор, если есть классы, они склеиваются в формате тег.класс1.класс2, если ничего нет, возвращается просто имя тега (например, button).

Далее эти элементы будут показаны пользователю при создании теста, чтобы ему не приходилось вводить код элемента, на котором он хочет провести тестирование вручную, а он мог просто выбрать его в списке, сгенерированном сайтом.

В случае негативного вызова эндпоинт вернет 400 ошибку. Код эндпоинта представлен ниже.

```
@app.get("/parse_elements")
async def parse_elements(url: str):
    try:
        headers = {'User-Agent': 'Mozilla/5.0 (Windows NT 10.0;
Win64; x64) AppleWebKit/537.36'}
```

ПГУ 1.09.03.01.08.001 ПЗ					Лист
					42
Изм.	Лист	№ докум.	Подпись	Дата	

```

        async with httpx.AsyncClient() as client:
            response = await client.get(url, headers=headers,
timeout=5.0)

        if response.status_code != 200:
            raise HTTPException(status_code=400, detail="Не
удалось скачать указанную страницу")

        soup = BeautifulSoup(response.text, 'html.parser')
        elements = []

        for tag in soup.find_all(['button', 'a', 'h1', 'h2',
'span', 'div']):
            text = tag.get_text(strip=True)
            if not text or len(text) > 100:
                continue
            if tag.get('id'):
                selector = f"#{tag.get('id')}"
            elif tag.get('class'):
                selector =
f"{tag.name}.{ '.'.join(tag.get('class'))}"
            else:
                selector = tag.name
            elements.append({"text": text, "selector": selector,
"type": tag.name})
        return {"url": url, "elements": elements[:50]}
    except Exception as e:
        raise HTTPException(status_code=500, detail=str(e))

```

Эндпоинт, находящийся по адресу «/favicon.ico» является технической заглушкой.

Если он не будет добавлен, то сервисы для генерации документации, такие как Swagger, будут выдавать ошибку, так как браузер по умолчанию автоматически отправляет фоновый запрос по этому адресу, чтобы скачать и отобразить маленькую иконку сайта на вкладке.

					ПГУ 1.09.03.01.08.001 ПЗ	Лист
						43
Изм.	Лист	№ докум.	Подпись	Дата		

В случае удачного вызова эндпоинт вернет код 204.

Эндпоинт группы «/client_config» отвечает за передачу всех визуальных данных на сторону фронтенда клиента.

Сервер достает из БД эксперименты со статусом «Активно» и вызывает «движок» для каждого активного теста. В зависимости от режима тестирования система распределяет пользователей по группам. Далее сервер достает из БД изменения селектора для варианта В и пересылает их клиентам.

Фронтэнду возвращается JSON-файл с ID эксперимента, названием эксперимента, группой, в которую был распределен пользователь и изменения в селекторе, если пользователь попал в тестовую группу.

Эндпоинт группы «/log_event» отвечает за сбор и агрегацию метрик в реальном времени. Он вызывается JS-сниппетом ab_platform.js.

Для корректной работы эндпоинта создается структура, в которую будут записываться данные. Данный процесс гарантирует правильность формата отправленных данных, что упростит последующее представление данных для их будущей аналитики. Данные о работе эксперимента записываются в реальном времени и могут быть запрошены администратором в любое время. На их основании он может принять решение о прекращении или продолжении эксперимента.

В случае успешного вызова данные запишутся в структуру «EventLog» и пользователь получит сообщение об успешном вызове метода.

Полный код модуля main.py представлен в приложении Б.

2.5 Разработка скрипта-симулятора действий пользователей

Данный метод представляет собой скрипт для проверки работоспособности системы. Модуль генерирует пользователей, которые заходят на сайт и заданной вероятностью совершают ключевое действие. При запуске метода для каждого пользователя отправляется POST-запрос для записи данных в БД. Далее случайным образом выбирается группа пользователей, которые «совершили» целевое действие и для этих

					ПГУ 1.09.03.01.08.001 ПЗ	Лист
						44
Изм.	Лист	№ докум.	Подпись	Дата		

пользователей отправляется еще один POST-запрос с указанием данных пользователя, данных об устройстве пользователя и типа действия, совершенного пользователем.

После завершения работы метода мы можем проверить наполненность БД и выходную аналитику. Если все сработало верно, значит запись в БД, интеграция через API и расчет статистических метрик работают правильно.

Полный код модуля simulator.py представлен в приложении Б.

2.6 Разработка встраиваемого сниппета

Данный модуль представляет собой код JS-сниппета, который должен быть вставлен в код веб-сайта пользователя.

После создания теста сайт создает уникальную ссылку для пользователя для подключения к модулю ab_platform.js.

Данный модуль содержит все функции, необходимые для проведения теста – он отправляет запрос к FastAPI для получения данных для клиента. Бэкенд распределяет пользователей по группам и отправляет данные для низменных элементов фронтэнду. Также логированием просмотров и кликов занимается эта часть программы.

Полный код метода ab_platform.js представлен в приложении Б.

3 Тестирование приложения

При разработке программного модуля одной из важных задач является его тестирование. Проведение тестирования позволяет не только выявить возможные ошибки в коде, но и обнаружить проблемные моменты в функциональности, а также убедиться в том, что все предварительно поставленные задачи и требования выполняются в соответствии с ожидаемым результатом. В рамках текущего тестирования будут проверены и протестированы следующие функции приложения:

- создание теста;
- регистрация действий пользователя

					ПГУ 1.09.03.01.08.001 ПЗ	Лист
						45
Изм.	Лист	№ докум.	Подпись	Дата		

– работа алгоритмов распределения трафика и расчета статистической значимости.

3.1 Создание теста

Таблица 1 – Тест-кейс №1

Тест-кейс №1 предназначен для проверки корректности работы механизма создания теста	
Шаги	Ожидаемый результат
Открыть вкладку «Создание теста»	Переход пользователя в окно создания теста
Вставить ссылку на сайт	Вывод данных обо всех элементах сайта
Ввести все данные теста и подтвердить создание теста	Успешное создание теста и переход на вкладку «Аналитика» для данного теста

Открыв сайт, пользователь увидит окно создания теста. Также пользователь может перейти в эту вкладку, нажав кнопку «Новый тест». Интерфейс сайта представлен на рисунке 17.

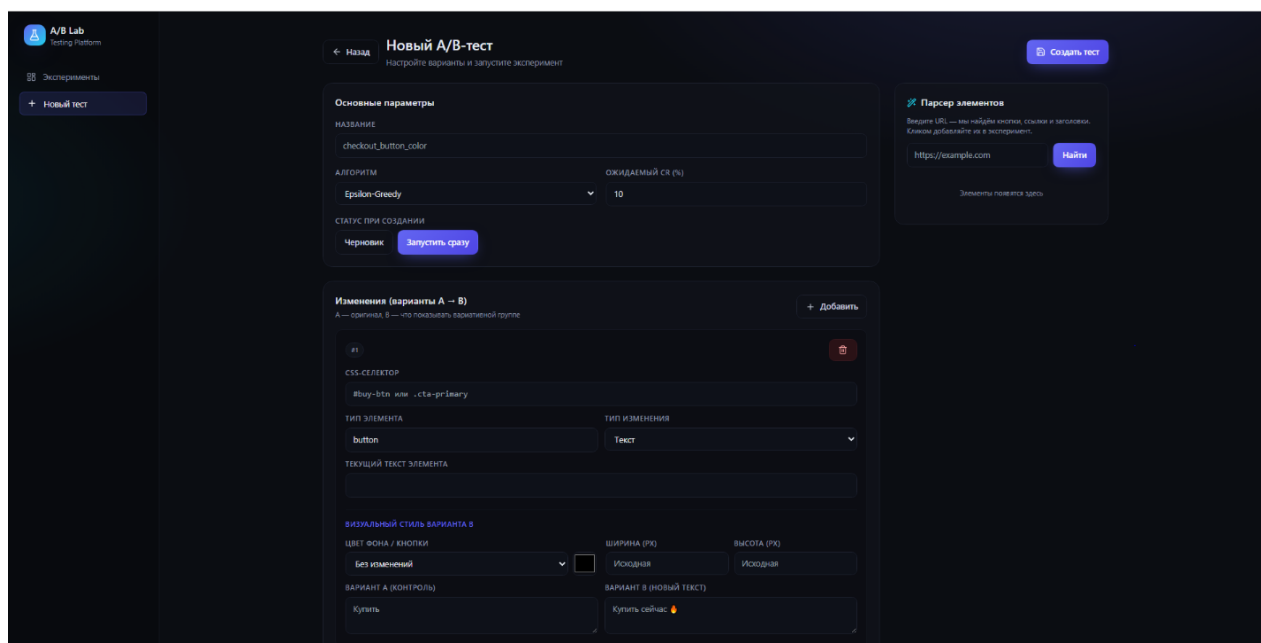


Рисунок 17 – Интерфейс сайта

Далее вставим ссылку в форму «Парсер элементов». Нажав кнопку найти, пользователь получит список из всех элементов, доступных для изменения, как это показано на рисунке 18.

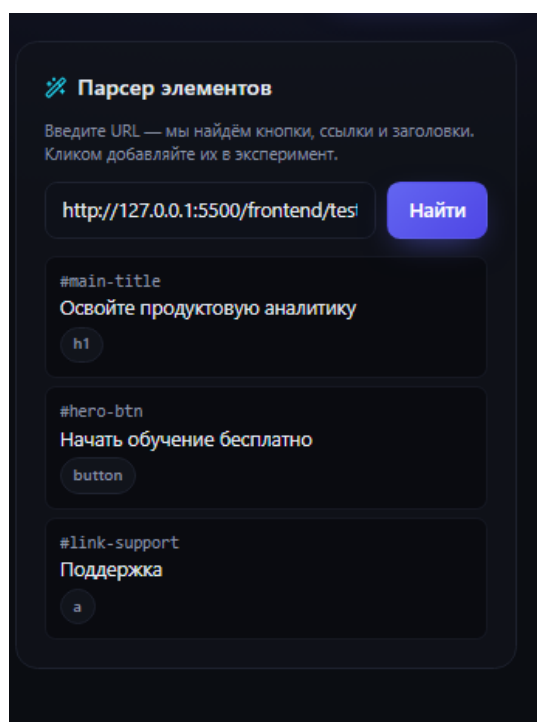


Рисунок 18 – Список элементов пользовательского сайта

Далее заполним поля создания теста, выбрав режим работы теста, ожидаемый результат конверсии, статус при создании теста и изменения элемента. Пример заполненных полей представлен на рисунке 19.

Основные параметры

НАЗВАНИЕ

example

АЛГОРИТМ

A/B 50/50

ОЖИДАЕМЫЙ CR (%)

10

СТАТУС ПРИ СОЗДАНИИ

Черновик

Запустить сразу

Изменения (варианты A → B)

A — оригинал, B — что показывать вариативной группе

+ Добавить

#1

CSS-СЕЛЕКТОР

#hero-btn

ТИП ЭЛЕМЕНТА

button

ТИП ИЗМЕНЕНИЯ

CSS-стиль

ТЕКУЩИЙ ТЕКСТ ЭЛЕМЕНТА

Начать обучение бесплатно

ВИЗУАЛЬНЫЙ СТИЛЬ ВАРИАНТА B

ЦВЕТ ФОНА / КНОПКИ

Синий

ШИРИНА (PX)

200

ВЫСОТА (PX)

100

ВАРИАНТ A (КОНТРОЛЬ)

Начать обучение бесплатно

ВАРИАНТ B (НОВЫЙ ТЕКСТ)

Приступить к обучения бесплатно

Рисунок 19 – Пример данных для создания теста

Введя все необходимые данные нажмем кнопку «Создать тест». В случае успешного выполнения операции должна открыться вкладка аналитики для нового тестирования.

Система отработала правильно и тест был успешно создан. Вкладка аналитики, созданная для теста представлена на рисунке 20.

					ПГУ 1.09.03.01.08.001 ПЗ	Лист
						48
Изм.	Лист	№ докум.	Подпись	Дата		

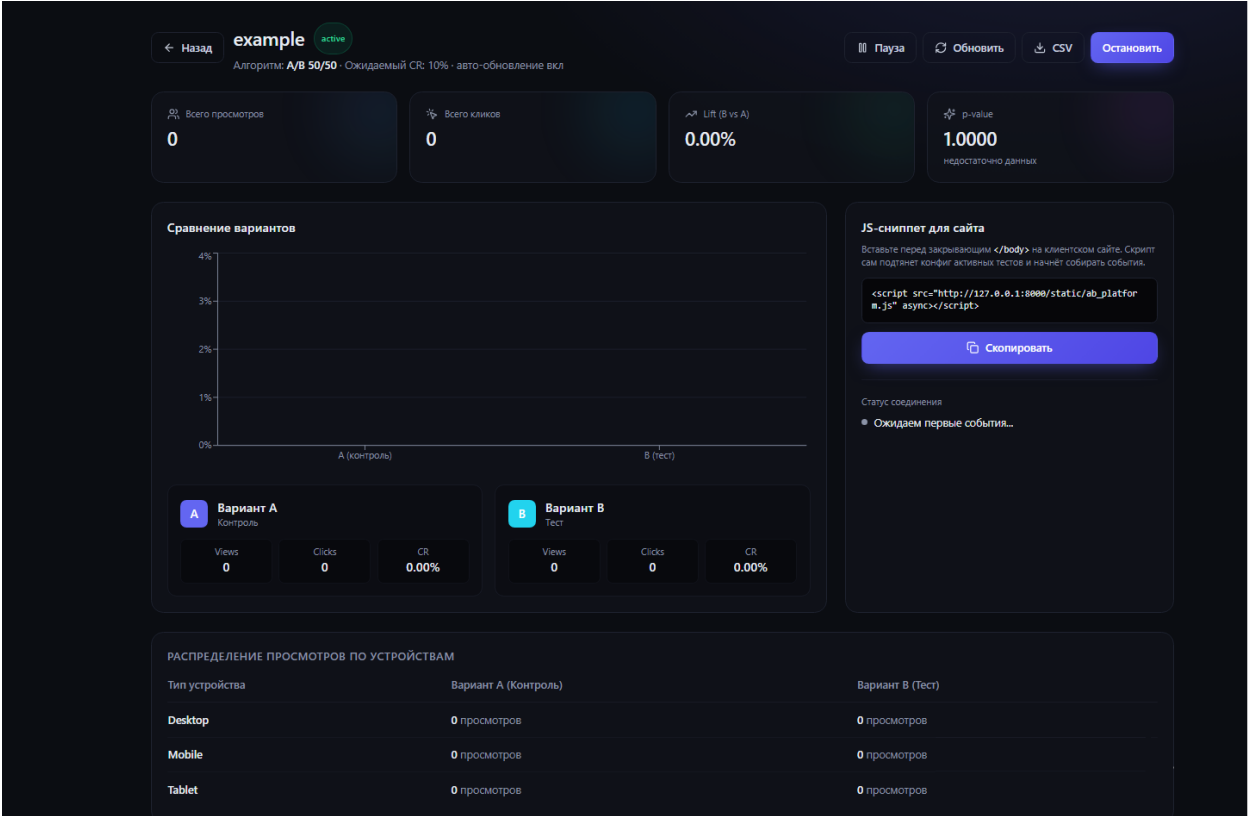


Рисунок 20 – Интерфейс вкладки аналитики

3.2 Проведение тестирования для одного пользователя

Таблица 2 – Тест-кейс №2

Тест-кейс №2 предназначен для проверки системы сбора данные о действиях пользователя	
Шаги	Ожидаемый результат
Вставить JS-сниппет в код сайта	Установка соединения между бэкендом и сайтом пользователя-администратора
Убедится в работе теста	Отображение обоих вариантов тестов для разных пользователей
Совершить целевое действие (просмотр, клик)	Отображение данные о просмотре в данных аналитики

Сайт автоматически создаст JS-сниппет и подскажет куда его необходимо вставить. Вставим в код сайта клиента как показано на рисунке 21.

```

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Демонстрационный сайт клиента</title>

  <script src="http://127.0.0.1:8000/static/ab_platform.js" async></script>
  <style>

```

Рисунок 21 – Вставка сниппета в код сайта

Тест создан. Зайдем на сайт, который был выбран для проведения тестирования. Нам выпал вариант «А». Интерфейс показан на рисунке 22.

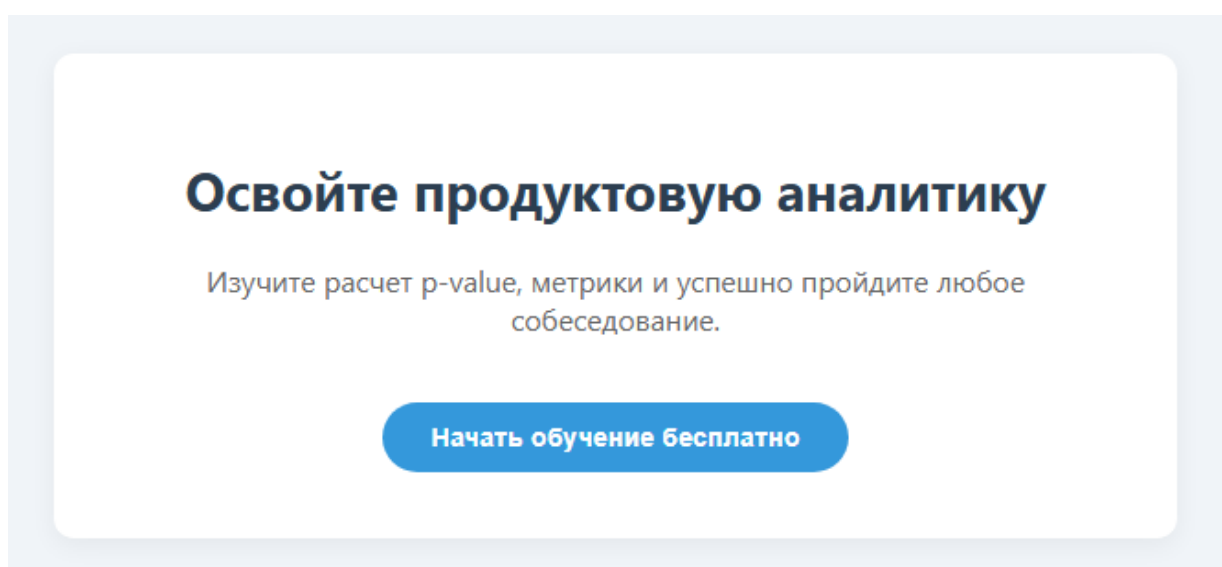


Рисунок 22 – Пример варианта А

Для того чтобы убедиться в правильности работы алгоритма зайдем на сайт еще раз с помощью вкладки инкогнито, так как сайт настроен так, чтобы с одной вкладки засчитывался только один просмотр и всегда показывался только один вариант. Интерфейс варианта «В» показан на рисунке 23, все примененные изменения были применены.

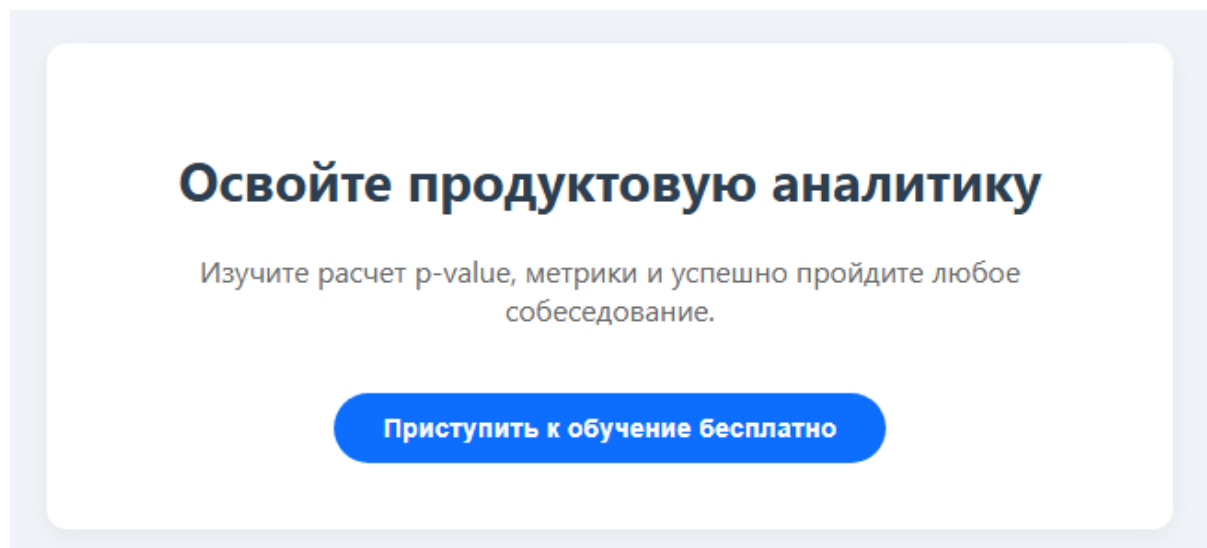


Рисунок 23 – Пример варианта В

Далее совершим действие «Клик». В случае успешной работы алгоритма в консоли браузера должно появиться уведомление об этом – ответ 200 на вызов эндпоинта «log_event». Консоль браузера представлена на рисунке 24.

test_client.html	304	docume...	Other	0.1 kB	3 ms
ab_platform.js	200	script	test_client.html:9	(memor...	1 ms
client_config?url=http%3A%2F...	200	fetch	ab_platform.js:27	0.5 kB	15 ms
log_event	200	fetch	ab_platform.js:63	0.2 kB	14 ms

Рисунок 24 – Консоль браузера

Чтобы проверить результаты всех проверок вернемся во вкладку аналитики, где должны отразиться данные по кликам и просмотрам. Результат работы представлен на рисунке 25.

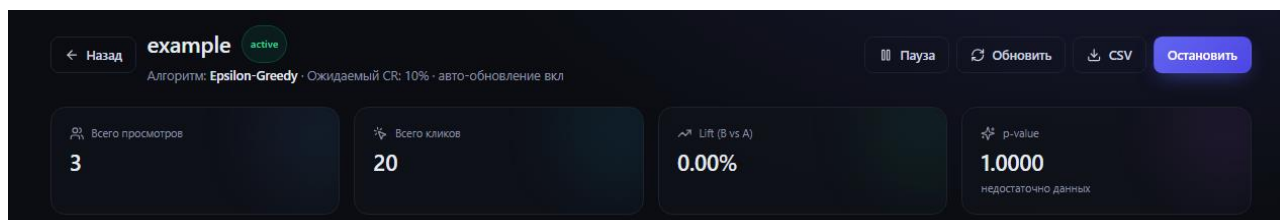


Рисунок 25 – Окно аналитики

3.3 Тестирование с помощью скрипта

					ПГУ 1.09.03.01.08.001 ПЗ	Лист
						51
Изм.	Лист	№ докум.	Подпись	Дата		

Таблица 3 – Тест-кейс №3

Тест-кейс №3 предназначен для проверки работы алгоритмов распределения трафика и алгоритмов расчета статистическая значимости и прироста.	
Шаги	Ожидаемый результат
Запустить скрипт для симуляции действий 1000 пользователей в режиме «многорукий бандит»	Отображение в данных аналитики результатов теста с распрделением пользователей в соотношении примерно «50/950»
Запустить скрипт для симуляции действий 1000 пользователей в режиме «50/50»	Отображение в данных аналитики результатов теста с распрделением пользователей в соотношении примерно «500/500»
Запустить скрипт для симуляции действий 1000 пользователей в режиме «алгоритм Томпсона»	Отображение в данных аналитики результатов теста с распрделением пользователей в соотношении примерно «10/900»

Запустим скрипт simulator.py, как это показано на рисунке 26.

```
PS C:\Users\new_v-main> python simulator.py
Запуск симуляции 1000 пользователей с разных устройств...
Симуляция успешно завершена!
```

Рисунок 26 – Запуск скрипта

Убедимся в правильности работы скрипта мы можем, посмотрев в консоль сервера. В случае успешной работы сервер должен возвращать код 200 на каждый запрос. Логи сервера представлены на рисунке 27.

```
INFO: 127.0.0.1:49281 - "GET /analytics?experiment_name=example HTTP/1.1" 200 OK
INFO: 127.0.0.1:49281 - "GET /experiments HTTP/1.1" 200 OK
INFO: 127.0.0.1:49281 - "GET /analytics?experiment_name=example HTTP/1.1" 200 OK
INFO: 127.0.0.1:49281 - "GET /experiments HTTP/1.1" 200 OK
INFO: 127.0.0.1:49281 - "GET /analytics?experiment_name=example HTTP/1.1" 200 OK
INFO: 127.0.0.1:49281 - "GET /experiments HTTP/1.1" 200 OK
INFO: 127.0.0.1:49281 - "GET /analytics?experiment_name=example HTTP/1.1" 200 OK
INFO: 127.0.0.1:49281 - "GET /experiments HTTP/1.1" 200 OK
INFO: 127.0.0.1:49361 - "GET /analytics?experiment_name=example HTTP/1.1" 200 OK
INFO: 127.0.0.1:49361 - "GET /experiments HTTP/1.1" 200 OK
INFO: 127.0.0.1:49402 - "GET /analytics?experiment_name=example HTTP/1.1" 200 OK
INFO: 127.0.0.1:49402 - "GET /experiments HTTP/1.1" 200 OK
INFO: 127.0.0.1:49447 - "GET /analytics?experiment_name=example HTTP/1.1" 200 OK
INFO: 127.0.0.1:49447 - "GET /experiments HTTP/1.1" 200 OK
```

Рисунок 27 – Логи сервера

После завершения работы скрипта, открыв окно аналитики сайта, мы увидим, что просмотры и клики «пользователей» были зафиксированы и отображены в статистике. Результаты работы алгоритма распределения совпали с теоретическим значением «50/950», рассчитанным перед запуском теста. Этих данных хватило для расчета статистической значимости. Результаты работы тестирования представлены на рисунке 28. Как можно видеть на этом рисунке, в этом тесте победил вариант В, прирост составил 156%.

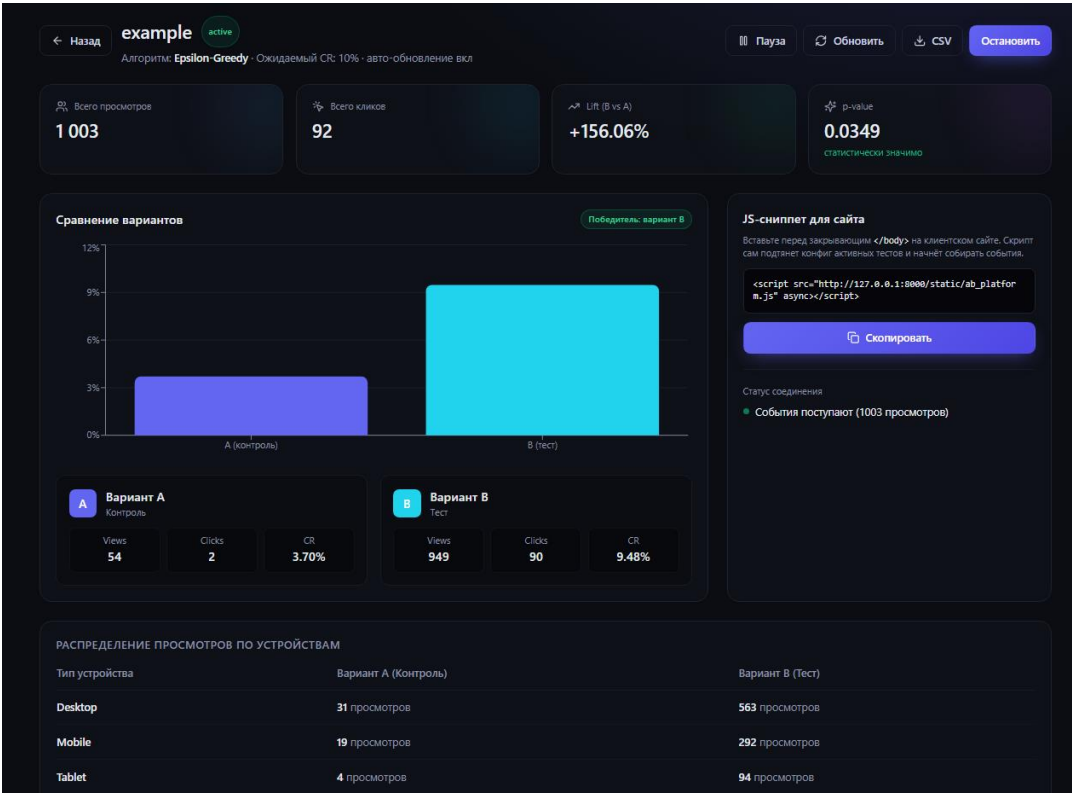


Рисунок 28 – Результат работы в режиме «многорукий бандит»

Далее, чтобы убедиться в правильности работы алгоритмов распределения трафика, прогоним скрипт еще 2 раза с разными настройками режима работы.

На рисунке 29 представлен результат работы алгоритма в режиме «50/50». Как видно по результатам режим работает правильно. Однако тысячи пользователей не хватило, чтобы аналитика считалась достоверной, поэтому для получения статистически значимых результатов пришлось запустить скрипт 2 раза. Распределение также примерно соответствует эталонному значению «50/50».

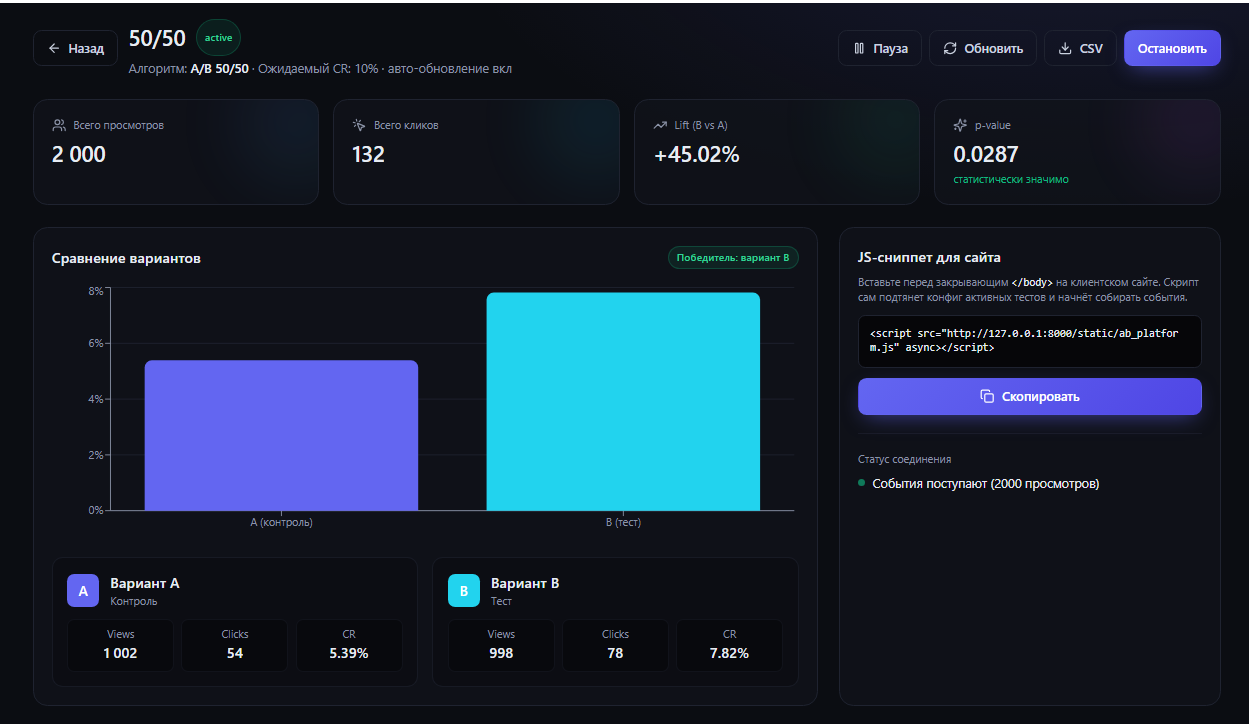


Рисунок 29 – Результат работы в режиме «50/50»

Также создадим тест в режиме «алгоритм Томпсона». На рисунке 30 представлен результат работы. В ходе проведения тестирования тысячью пользователей случилась статистическая аномалия и алгоритм выдал почти равное разделение. Такое возможно, так как первые 100 пользователей делятся в пропорции 50/50, а дальше вариант, получивший преимущество в конверсии на первом этапе должен получать большее количество показов. Однако, если на первом было получено равенство и сохранялось далее, то просмотры могут продолжить делиться в пропорции 50/50. Чтобы доказать, что это лишь

статистическая аномалия, и алгоритм работает правильно, запустим скрипт еще 2 раза. В итоге преимущество должно появиться у варианта В, так как на данный момент его конверсия выше, и должно продолжать расти с большим количеством пользователей. Результаты следующих запусков представлены на рисунках 31, 32.

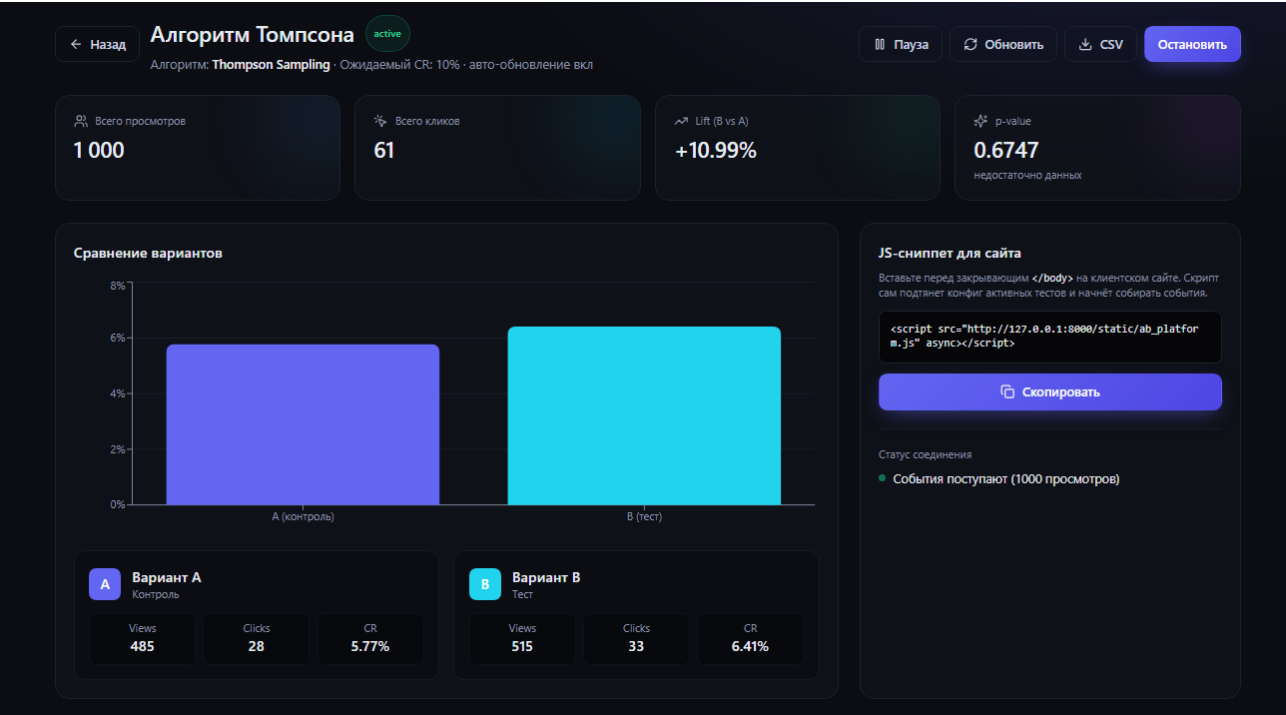


Рисунок 30 – Первый запуск в режиме «алгоритм Томпсона»

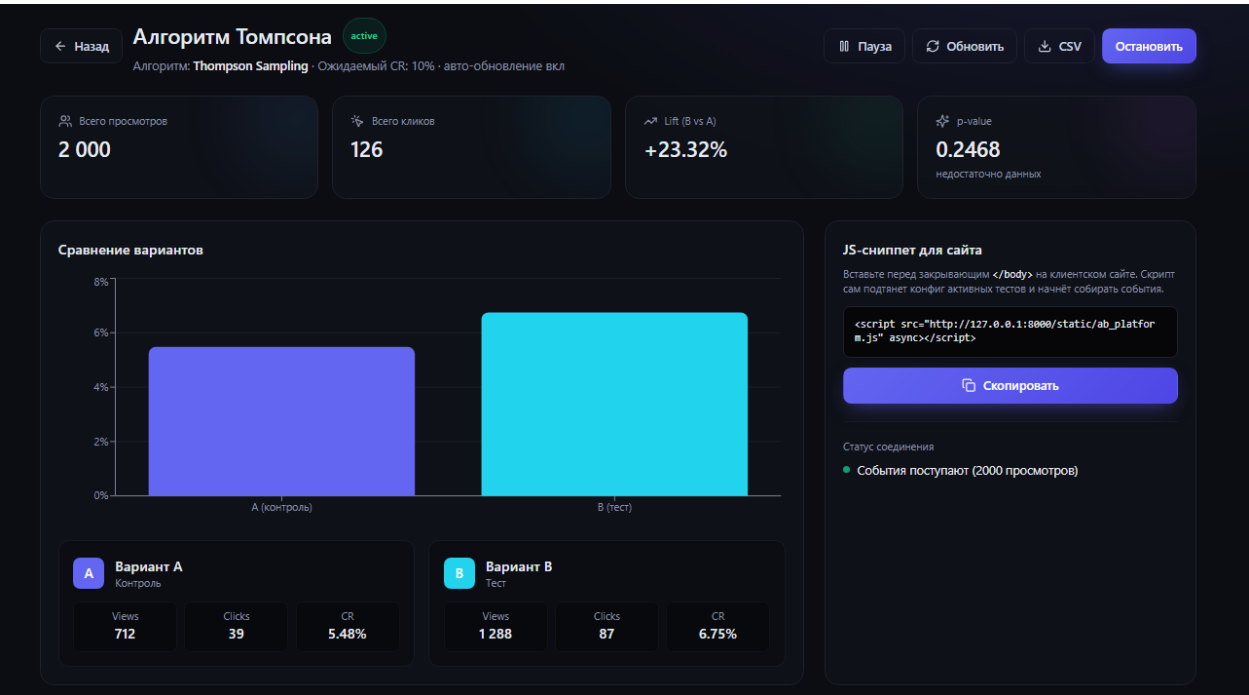


Рисунок 31 – Второй запуск в режиме «алгоритм Томпсона»

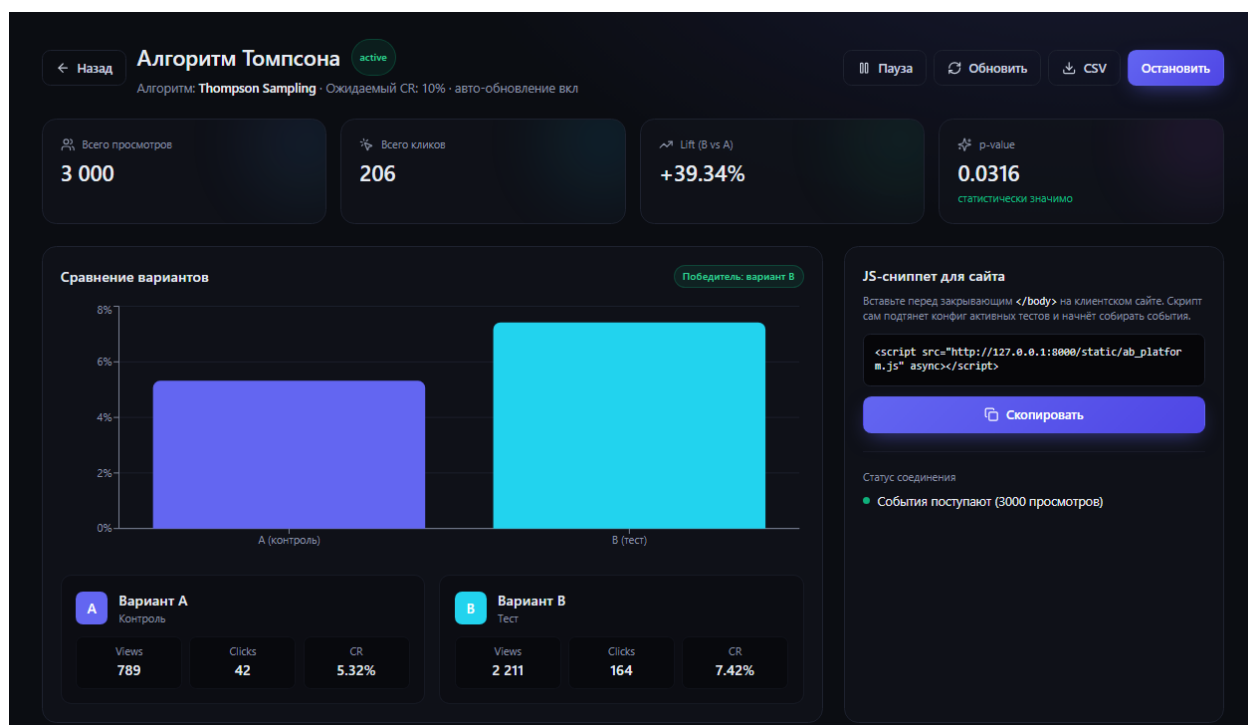


Рисунок 32 – Третий запуск в режиме «алгоритм Томпсона»

Как видно по результатам тестов гипотеза подтвердилась – с каждым последующим запуском количество показов варианта А уменьшалось.

4 Безопасность жизнедеятельности

В наше время работа за компьютером стала неотъемлемой частью жизни многих людей, что приводит к появлению различных проблем со здоровьем, таких как напряжение глаз, неправильная осанка, туннельный синдром и других заболеваний. В особенности это касается людей чья профессиональная деятельность связана с долгим нахождением перед экраном монитора. Для того, чтобы обезопасить себя от проблем продолжительной работы за компьютером, необходимо соблюдать ряд правил. В этой главе мы рассмотрим необходимые меры, при соблюдении которых, можно предотвратить все возможные проблемы, связанные с долгой работой за компьютером.

4.1 Требования к организации рабочего места

					ПГУ 1.09.03.01.08.001 ПЗ	Лист
						56
Изм.	Лист	№ докум.	Подпись	Дата		

При организации рабочего места необходимо соблюдать ряд рекомендаций, которые помогут минимизировать риски для здоровья.

Рассмотрим список предлагаемых рекомендаций:

- эргономичное кресло. Для комфортной работы за компьютером важно приобрести хорошее и качественное кресло, которое обладает возможностью регулировки высоты и угла наклона, и необходимой формой для спины и комплекции человека;

- правильная высота стола. Рабочий стол должен быть на такой высоте, чтобы руки находились под углом 90 градусов при работе с клавиатурой и мышью;

- положение монитора. Чтобы предотвратить проблемы со зрением, монитор должен находиться на комфортном расстоянии от глаз. Рекомендуемым значением является 50-70 см от монитора. Положение монитора должно быть такое чтобы, верхняя часть экрана была на уровне глаз.

Пример соблюдения всех правил можно увидеть на рисунке 33.



Рисунок 33 – Правильное положение при работе за компьютером

4.2 Требования к освещению

					ПГУ 1.09.03.01.08.001 ПЗ	Лист
						57
Изм.	Лист	№ докум.	Подпись	Дата		

Одна из главных проблем, которая возникает при долгой работе за компьютером – это вред для глаз. Неправильное освещение помещения и настроек яркости монитора, может привести к усталости глаз, сухости, покраснению, размытости зрения и головным болям. Чтобы предотвратить это, необходимо соблюдать некоторые требования к освещению в помещении:

– естественное освещение. Рекомендуется, по возможности, использовать естественный свет, при этом избегать попадания прямых солнечных лучей на монитор. Естественный свет снижает напряжение глаз и улучшает общее самочувствие;

– искусственное освещение. Для работы в вечернее время рекомендуется использовать искусственное освещение, такие как настольное лампа, комнатное освещение и т.д. Предпочтительно комбинировать освещения для более лучшего эффекта. Освещение должно быть ярким, чтобы избежать напряжения глаз, но не слишком интенсивным чтобы исключить появления бликом на экране монитора. Также нужно учитывать правильное расположение источников света, Локальный источник, например настольная лампа, должен быть установлен сбоку от монитора, чтобы не создавать прямых бликов на экране;

– настройка монитора. Правильная настройка яркости и контрастности монитора, также может уменьшить напряжение глаз при длительной работе за ПК. Рекомендуется в вечернее время использовать ночной режим, который делает цвета экрана более теплыми. Это поможет уменьшить нагрузку на глаза, а также улучшить качество сна.

4.3 Требования к режиму отдыха

Чтобы избежать проблем со здоровьем при длительной работе за компьютером, необходимо ежечасно делать небольшой перерыв, для выполнения разминки, чтобы избежать мышечного напряжения и улучшить кровообращение в суставах и мышцах.

					ПГУ 1.09.03.01.08.001 ПЗ	Лист
						58
Изм.	Лист	№ докум.	Подпись	Дата		

Не стоит забывать и про глаза, которые так же подвержены воздействию, поэтому будет полезно сделать зарядку, которая включает в себя круговые движения открытыми глазами, смена точки фокусировки взгляда с близкой до дальней.

Важно также соблюдать строгий график работы и отдыха, ведь постоянный труд может привести к негативным последствиям. Необходимо составить четкое расписание и расписать план работы, для оптимизации своего времени. Не следует работать в дни отдыха, переработки могут плохо сказаться на вашем физическом и ментальном здоровье.

При соблюдении данных рекомендаций вы сможете обеспечить себе комфортную работу с минимальным риском для своего здоровья.

					ПГУ 1.09.03.01.08.001 ПЗ	Лист
						59
Изм.	Лист	№ докум.	Подпись	Дата		

Заключение

В ходе выполнения работы был проведён глубокий анализ предметной области А/В-тестирования и существующих аналогичных решений, на основании которых сформулирована и обоснована постановка задачи разработки автономной системы аналитики.

Основной этап работы был посвящён выбору и настройке инструментария, отвечающего требованиям производительности и кроссплатформенности. Была выполнена установка и настройка интерпретатора Python 3.9+ и развернута изолированная среда разработки в Visual Studio Code.

Для реализации серверной логики и обеспечения высокоскоростного обмена данными был подготовлен проект на базе фреймворка FastAPI. В рамках настройки программной среды осуществлено подключение библиотеки SQLAlchemy для взаимодействия с базой данных PostgreSQL, а также интегрированы специализированные модули SciPy для проведения статистических вычислений и расчёта достоверности экспериментов.

Итоговым результатом проделанной работы является полностью сформированная техническая база и настроенное окружение, позволяющие перейти к этапу проектирования архитектуры и непосредственной реализации алгоритмов распределения трафика и визуализации аналитических данных Web-приложения.

					ПГУ 1.09.03.01.08.001 ПЗ	Лист
						60
Изм.	Лист	№ докум.	Подпись	Дата		

Список использованных источников

1. ГОСТ Р ИСО/МЭК 12207-2010. Информационная технология. Системная и программная инженерия. Процессы жизненного цикла программных средств. – Введ. 2012–03–01. – Москва : Стандартинформ, 2011. – 106 с.
2. Котляров, В. П. Основы тестирования программного обеспечения : учебное пособие / В. П. Котляров, Т. В. Коликова. – Москва : Интернет-Университет Информационных Технологий, 2020. – 285 с.
3. Кохви, Р. Доверительное управление данными. Инженерия и культура А/В-тестирования / Р. Кохви, Д. Танг, Я. Сюй ; перевод с английского А. Ю. Гинько. – Москва : Альпина Паблишер, 2021. – 452 с.
4. Любанович, Б. Simple Python. Современный стиль программирования / Б. Любанович. – 2-е изд. – Санкт-Петербург : Питер, 2021. – 592 с.
5. Фаулер, М. Рефакторинг: улучшение проекта существующего кода / М. Фаулер. – 2-е изд. – Москва : Вильямс, 2019. – 448 с.
6. FastAPI Framework Documentation [электронный ресурс] – URL: <https://fastapi.tiangolo.com> (дата обращения: 15.03.2026).
7. SQLAlchemy 2.0 Documentation [электронный ресурс] – URL: <https://docs.sqlalchemy.org/en/20/> (дата обращения: 18.03.2026).
8. SciPy v1.11.0 Reference Guide [электронный ресурс] – URL: <https://docs.scipy.org/doc/scipy/> (дата обращения: 20.03.2026).
9. Chart.js: Open source HTML5 Charts for your website [электронный ресурс] – URL: <https://www.chartjs.org/docs/latest/> (дата обращения: 22.03.2026).
10. Visual Studio Code: официальный сайт [электронный ресурс] – URL: <https://code.visualstudio.com> (дата обращения: 28.05.2026).
11. pgAdmin : официальный сайт [электронный ресурс] – URL: <https://www.pgadmin.org> (дата обращения: 28.05.2026).

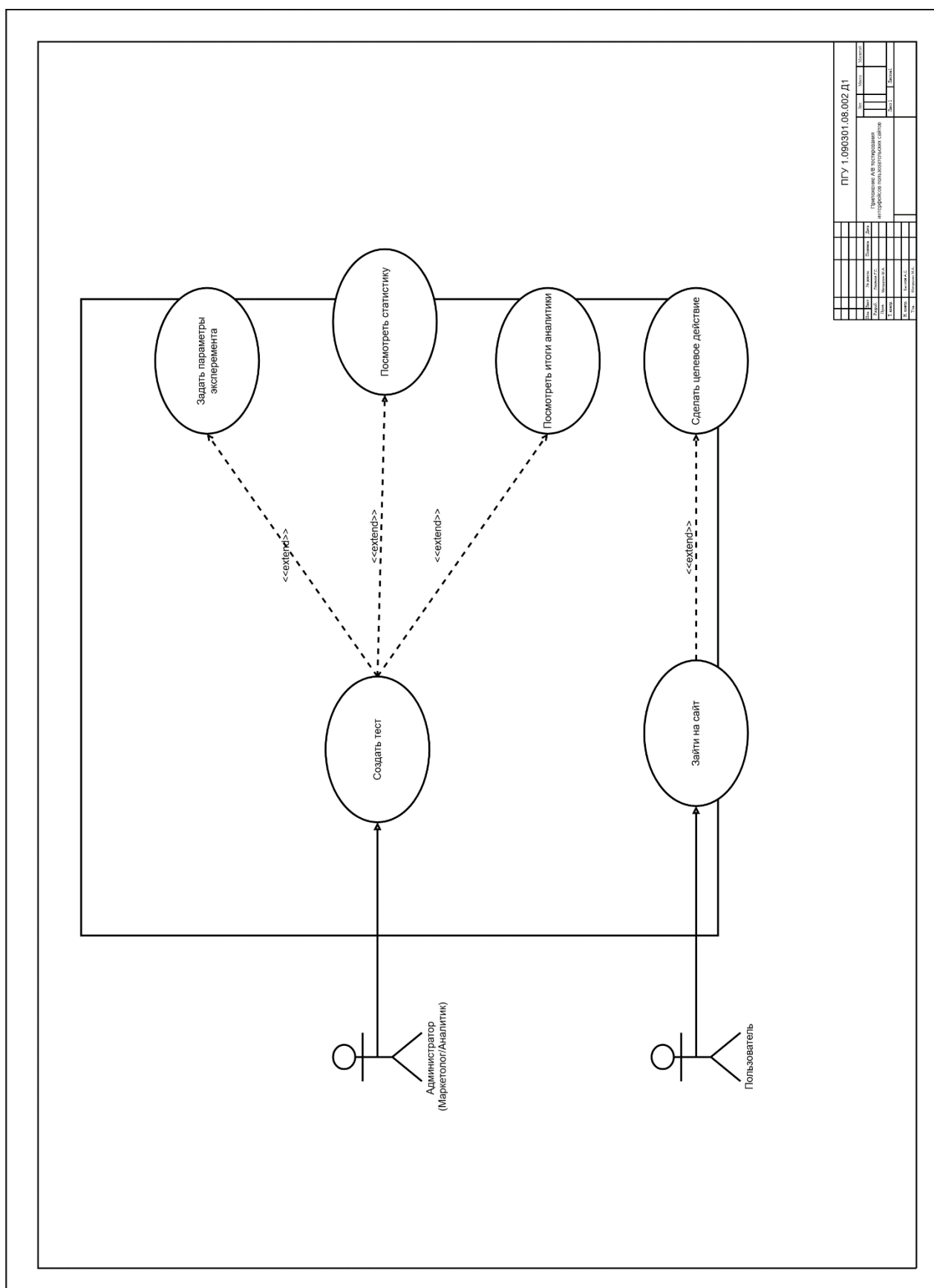
Приложение А

(Обязательное)

Приложение для А/В тестирования пользовательских интерфейсов сайтов
UML диаграммы

					ПГУ 1.09.03.01.08.001 ПЗ	Лист
						62
Изм.	Лист	№ докум.	Подпись	Дата		

Приложение А.1 – Use-case диаграмма



ПГУ 1.09.03.01.08.002 Д1									
№	№ докум.	Дата	Исполнитель	Проверенный	Дата	Исполнитель	Проверенный	Дата	Исполнитель
1									
2									
3									
4									
5									
6									
7									
8									
9									
10									

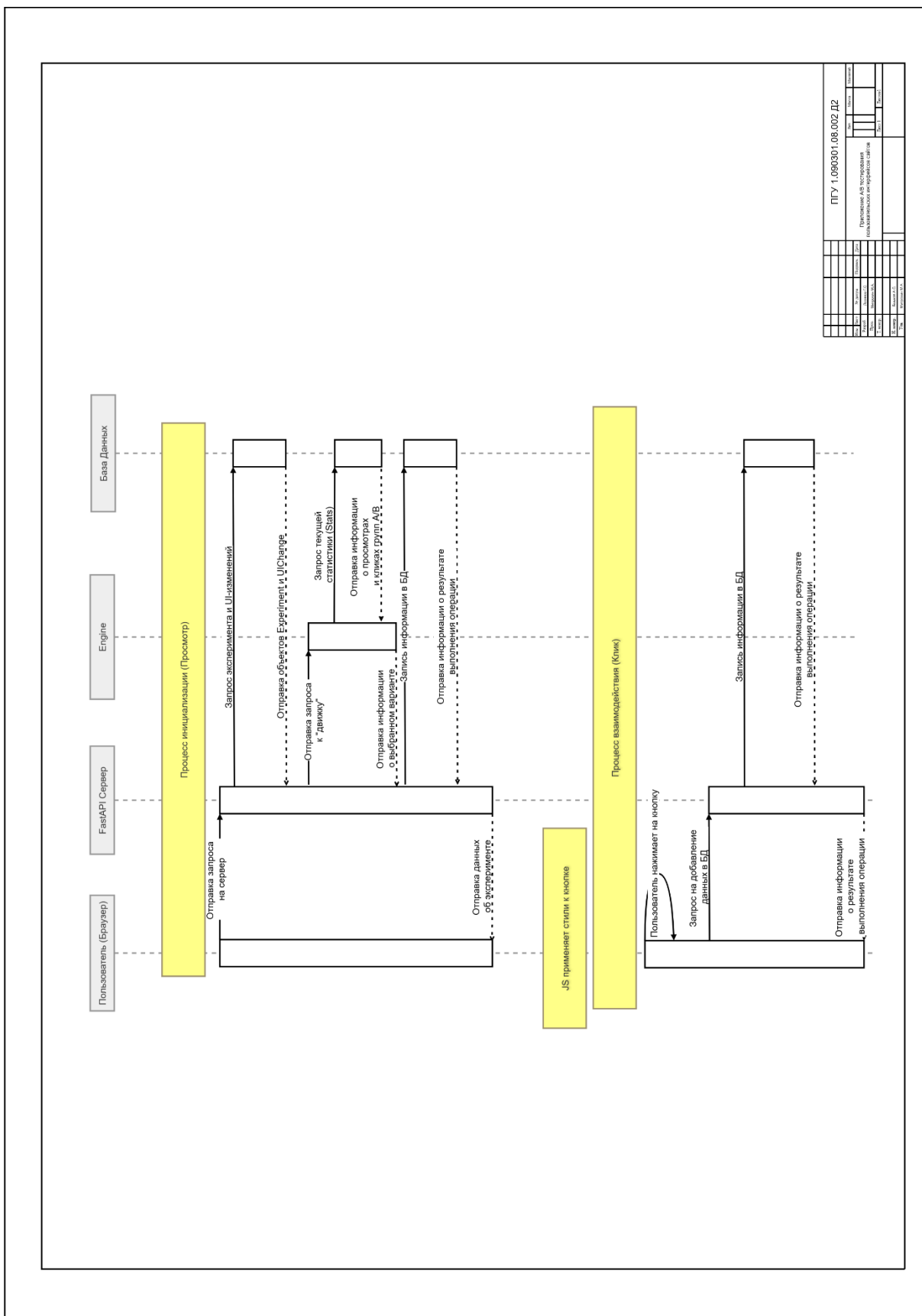
Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.09.03.01.08.001 ПЗ

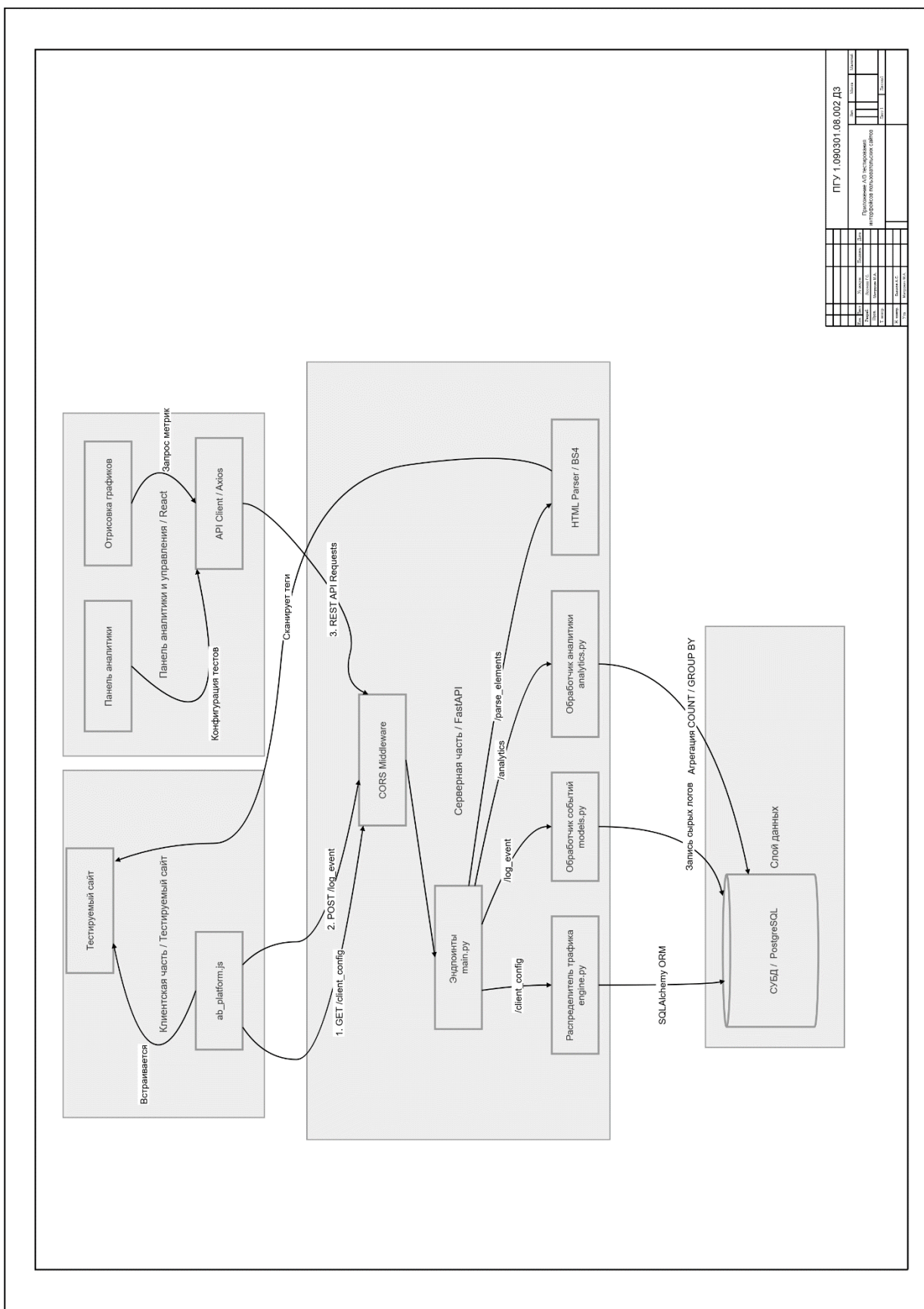
Лист

63

Приложение А.2 – UML-диаграмма последовательности



Приложение А.3 – UML-диаграмма компонентов



ПГУ 1.090301.08.002 ДЗ	
Имя	Иванов И.И.
Фамилия	Иванов
Имя отчество	Иванов И.И.
Пол	Мужской
Дата рождения	1990.01.01
Место рождения	Москва
Подпись	

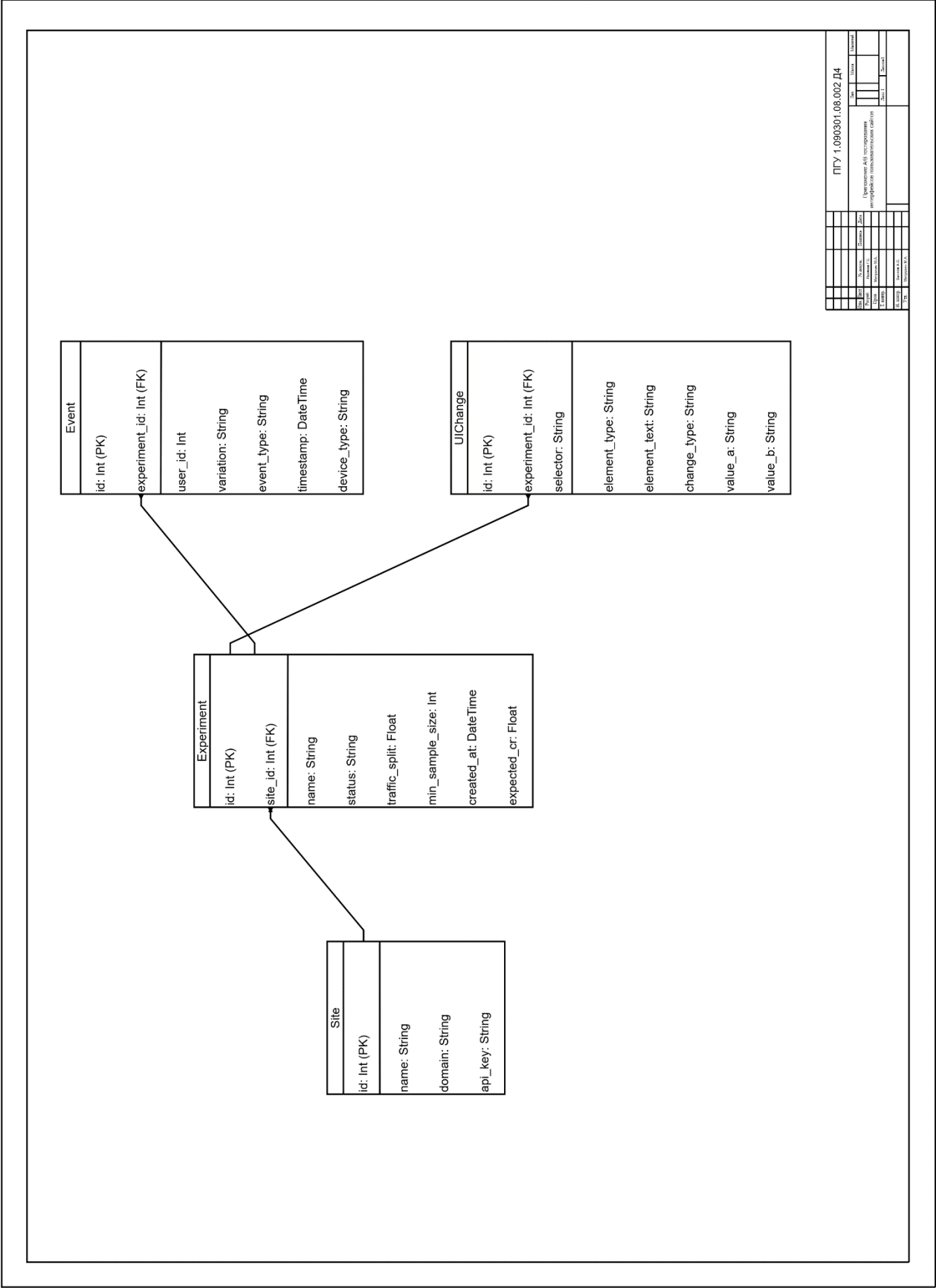
Приложение А.4 – UML диаграмма классов

ПГУ 1.09.03.01.08.001 ПЗ

Лист

65

Изм.	Лист	№ докум.	Подпись	Дата



Приложение Б

(Обязательное)

Приложение для А/В тестирования пользовательских интерфейсов
сайтов

Листинг программы

					ПГУ 1.09.03.01.08.001 ПЗ	Лист
						67
Изм.	Лист	№ докум.	Подпись	Дата		

Файл main.py

```
# python -m uvicorn main:app --reload

import csv
import math
import os
import random
import re
from io import StringIO
from typing import List

from fastapi import FastAPI, Depends, Request, Response,
HTTPException
from fastapi.middleware.cors import CORSMiddleware
from sqlalchemy.exc import IntegrityError
from fastapi.responses import StreamingResponse
from fastapi.staticfiles import StaticFiles
from pydantic import BaseModel
from sqlalchemy import create_engine
from sqlalchemy.orm import sessionmaker, Session
import httpx
from bs4 import BeautifulSoup

from analytics import calculate_ab_results
import models
from models import Event, Experiment, UIChange, Site
from engine import get_smart_variation

app = FastAPI(title="A/B Testing Platform API")

DATABASE_URL = os.getenv("DATABASE_URL", "sqlite:///./ab_test.db")

if DATABASE_URL.startswith("postgres://"):
```

					ПГУ 1.09.03.01.08.001 ПЗ	Лист
						68
Изм.	Лист	№ докум.	Подпись	Дата		

```

        DATABASE_URL      =      DATABASE_URL.replace("postgres://",
"postgresql://", 1)

if "sqlite" in DATABASE_URL:
    engine = create_engine(
        DATABASE_URL,
        connect_args={"check_same_thread": False, "timeout": 30}
    )
else:
    engine = create_engine(
        DATABASE_URL,
        pool_size=20,
        max_overflow=10,
        pool_timeout=30,
        pool_recycle=1800
    )

SessionLocal  =  sessionmaker(autocommit=False,  autoflush=False,
bind=engine)

models.Base.metadata.create_all(bind=engine)

app.add_middleware(
    CORSMiddleware,
    allow_origins=["*"],
    allow_credentials=True,
    allow_methods=["*"],
    allow_headers=["*"],
)

app.mount("/static",                                StaticFiles(directory="static"),
name="static")

class UIChangeItem(BaseModel):
    selector: str
    element_type: str
    element_text: str
    change_type: str

```

		change_type: str			ПГУ 1.09.03.01.08.001 ПЗ	Лист
						69
Изм.	Лист	№ докум.	Подпись	Дата		

value_a: str

value_b: str

```
class ExperimentCreate(BaseModel):
```

name: str

algorithm: str

expected_cr: float = 10.0

status: str = "draft"

changes: List[UIChangeItem]

```
class ExperimentOut(BaseModel):
```

id: int

site_id: int

name: str

algorithm: str

expected_cr: float

status: str

traffic_split: float

epsilon: float

min_sample_size: int

```
class Config:
```

from_attributes = True

```
class LogEventRequest(BaseModel):
```

user_id: str

experiment_name: str

variation: str

event_type: str

device_type: str = "Desktop"

```
def get_db():
```

db = SessionLocal()

try:

yield db

```
finally:
```

					ПГУ 1.09.03.01.08.001 ПЗ	Лист
						70
Изм.	Лист	№ докум.	Подпись	Дата		

```

db.close()

@app.post("/log_event")
def log_event(request: LogEventRequest, db: Session = Depends(get_db)):
    exp = db.query(Experiment).filter(Experiment.name == request.experiment_name).first()
    if not exp:
        raise HTTPException(status_code=404, detail="Эксперимент не найден")

    new_event = Event(
        user_id=request.user_id,
        experiment_id=exp.id,
        variation=request.variation,
        event_type=request.event_type,
        device_type=request.device_type
    )
    db.add(new_event)
    db.commit()
    return {"status": "success", "event_id": new_event.id}

@app.get("/client_config")
def get_config(url: str, user_id: str, db: Session = Depends(get_db)):

    experiments = db.query(Experiment).filter(Experiment.status.in_(["active", "in_progress"])).all()
    active_exps = []

    for exp in experiments:
        variant = get_smart_variation(user_id, exp.name, db)
        changes = db.query(UIChange).filter(UIChange.experiment_id == exp.id).all()

```

					ПГУ 1.09.03.01.08.001 ПЗ	Лист
						71
Изм.	Лист	№ докум.	Подпись	Дата		

```

        active_exps.append({
            "id": exp.id,
            "name": exp.name,
            "variant": variant,
            "changes": [
                {
                    "selector": c.selector,
                    "change_type": c.change_type,
                    "value_b": c.value_b
                } for c in changes
            ]
        })

    return {"active_experiments": active_exps}

@app.post("/experiments", response_model=ExperimentOut)
def create_experiment(data: ExperimentCreate, db: Session = Depends(get_db)):
    site = db.query(Site).first()
    if not site:
        site = Site(name="Default Site", domain="localhost",
            api_key="12345")
        db.add(site)
        db.commit()

        if db.query(Experiment).filter(Experiment.name ==
data.name).first():
            raise HTTPException(status_code=409, detail="Эксперимент с
таким названием уже существует")

    new_exp = Experiment(
        site_id=site.id,
        name=data.name,
        algorithm=data.algorithm,
        expected_cr=data.expected_cr,
        status=data.status

```

									Лист
									72
Изм.	Лист	№ докум.	Подпись	Дата	ПГУ 1.09.03.01.08.001 ПЗ				

```

    )
    db.add(new_exp)
    db.commit()
    db.refresh(new_exp)

for change in data.changes:
    ui_change = UIChange(
        experiment_id=new_exp.id,
        selector=change.selector,
        change_type=change.change_type,
        value_a=change.value_a,
        value_b=change.value_b,
        element_type=change.element_type,
        element_text=change.element_text
    )
    db.add(ui_change)

db.commit()
return new_exp

@app.get("/experiments")
def list_experiments(db: Session = Depends(get_db)):
    return
    db.query(Experiment).order_by(Experiment.id.desc()).all()

@app.delete("/experiments/{exp_id}")
def delete_experiment(exp_id: int, db: Session = Depends(get_db)):
    exp = db.query(Experiment).filter(Experiment.id ==
exp_id).first()
    if exp:
        db.query(UIChange).filter(UIChange.experiment_id ==
exp_id).delete()
        db.query(Event).filter(Event.experiment_id ==
exp_id).delete()
        db.delete(exp)
    db.commit()

```

					ПГУ 1.09.03.01.08.001 ПЗ	Лист
						73
Изм.	Лист	№ докум.	Подпись	Дата		

```

        return {"status": "deleted"}

@app.put("/experiments/{exp_id}/status")
def update_experiment_status(exp_id: int, status: str, db: Session = Depends(get_db)):
    exp = db.query(Experiment).filter(Experiment.id == exp_id).first()
    if not exp:
        raise HTTPException(status_code=404, detail="Эксперимент не найден")

    exp.status = status
    db.commit()
    return {"status": "success", "new_status": status}

@app.get("/analytics")
def get_detailed_analytics(experiment_name: str, db: Session = Depends(get_db)):
    exp = db.query(Experiment).filter(Experiment.name == experiment_name).first()
    if not exp:
        raise HTTPException(status_code=404, detail="Эксперимент не найден")

    def count_events(variation, event_type):
        return db.query(Event).filter(
            Event.experiment_id == exp.id,
            Event.variation == variation,
            Event.event_type == event_type
        ).count()

    views_a = count_events("A", "view")
    clicks_a = count_events("A", "click")
    views_b = count_events("B", "view")
    clicks_b = count_events("B", "click")

```

					ПГУ 1.09.03.01.08.001 ПЗ	Лист
						74
Изм.	Лист	№ докум.	Подпись	Дата		

```

        results = calculate_ab_results(views_a, clicks_a, views_b,
clicks_b)

from sqlalchemy import func
device_counts = db.query(
    Event.variation,
    Event.device_type,
    func.count(Event.id)
).filter(Event.experiment_id == exp.id, Event.event_type ==
"view").group_by(Event.variation, Event.device_type).all()

devices = {"Desktop": {"A": 0, "B": 0}, "Mobile": {"A": 0, "B":
0}, "Tablet": {"A": 0, "B": 0}}
for var, dev, count in device_counts:
    if dev in devices and var in devices[dev]:
        devices[dev][var] = count

return {
    "experiment_name": exp.name,
    "status": exp.status,
    "expected_cr": exp.expected_cr,
    "algorithm": exp.algorithm,
    "devices": devices,
    "variants": {
        "A": {"views": views_a, "clicks": clicks_a, "cr":
results.get("cr_a", 0)},
        "B": {"views": views_b, "clicks": clicks_b, "cr":
results.get("cr_b", 0)}
    },
    "statistics": {
        "p_value": results.get("p_value", 1.0),
        "significant": results.get("is_significant", False),
        "lift": results.get("lift", 0)
    }
}

```

					ПГУ 1.09.03.01.08.001 ПЗ	Лист
						75
Изм.	Лист	№ докум.	Подпись	Дата		

```

@app.get("/export_csv")
def export_csv(experiment_name: str, db: Session = Depends(get_db)):
    exp = db.query(Experiment).filter(Experiment.name == experiment_name).first()
    if not exp:
        raise HTTPException(status_code=404, detail="Эксперимент не найден")

    events = db.query(Event).filter(Event.experiment_id == exp.id).all()

    output = StringIO()
    writer = csv.writer(output)
    writer.writerow(['ID', 'User ID', 'Variation', 'Event Type', 'Timestamp'])

    for event in events:
        writer.writerow([event.id, event.user_id, event.variation, event.event_type, event.timestamp])

    output.seek(0)
    return StreamingResponse(
        iter([output.getvalue()]),
        media_type="text/csv",
        headers={"Content-Disposition": f"attachment; filename=export_{experiment_name}.csv"}
    )

@app.get("/parse_elements")
async def parse_elements(url: str):
    try:
        headers = {'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36'}
        async with httpx.AsyncClient() as client:

```

					ПГУ 1.09.03.01.08.001 ПЗ	Лист
						76
Изм.	Лист	№ докум.	Подпись	Дата		

```

        response = await client.get(url, headers=headers,
timeout=5.0)

        if response.status_code != 200:
            raise HTTPException(status_code=400, detail="Не удалось
скачать указанную страницу")

        soup = BeautifulSoup(response.text, 'html.parser')
        elements = []

        for tag in soup.find_all(['button', 'a', 'h1', 'h2', 'span',
'div']):
            text = tag.get_text(strip=True)
            if not text or len(text) > 100:
                continue

            if tag.get('id'):
                selector = f"#{tag.get('id')}"
            elif tag.get('class'):
                selector = f"{tag.name}.{tag.get('class')}"
            else:
                selector = tag.name

            elements.append({"text": text, "selector": selector,
"type": tag.name})

        return {"url": url, "elements": elements[:50]}
    except Exception as e:
        raise HTTPException(status_code=500, detail=str(e))

@app.get('/favicon.ico', include_in_schema=False)
async def favicon():
    return Response(status_code=204)

```

if __name__ == "__main__":					ПГУ 1.09.03.01.08.001 ПЗ	Лист 77
Изм.	Лист	№ докум.	Подпись	Дата		

```
import uvicorn

uvicorn.run(app, host="127.0.0.1", port=8000)
```

Файл models.py

```
from sqlalchemy import Column, Integer, String, DateTime,
ForeignKey, Float, Boolean, Text
from sqlalchemy.orm import relationship
from sqlalchemy.ext.declarative import declarative_base
from datetime import datetime
```

```
Base = declarative_base()
```

```
class Site(Base):
```

```
    __tablename__ = "sites"
```

```
    id = Column(Integer, primary_key=True, index=True)
```

```
    name = Column(String, unique=True)
```

```
    domain = Column(String)
```

```
    api_key = Column(String, unique=True)
```

```
    experiments = relationship("Experiment", back_populates="site")
```

```
class Experiment(Base):
```

```
    __tablename__ = "experiments"
```

```
    id = Column(Integer, primary_key=True, index=True)
```

```
    site_id = Column(Integer, ForeignKey("sites.id"))
```

```
    name = Column(String, unique=True)
```

```
    status = Column(String, default="draft")
```

```
    traffic_split = Column(Float, default=1.0)
```

```
    algorithm = Column(String, default="epsilon_greedy")
```

```
    epsilon = Column(Float, default=0.1)
```

```
    min_sample_size = Column(Integer, default=100)
```

					ПГУ 1.09.03.01.08.001 ПЗ	Лист
						78
Изм.	Лист	№ докум.	Подпись	Дата		

Файл engine.py

```
import random
from sqlalchemy.orm import Session
from sqlalchemy import func
from models import Event, Experiment

def get_smart_variation(user_id: str, exp_name: str, db: Session):
    exp = db.query(Experiment).filter(Experiment.name ==
exp_name).first()
    if not exp:
        return "A"

    if exp.algorithm == "classic":
        import hashlib
        hash_val = int(hashlib.md5((user_id +
exp_name).encode('utf-8')).hexdigest(), 16)
        return "A" if (hash_val % 100) < 50 else "B"

    counts = db.query(
        Event.variation,
        Event.event_type,
        func.count(Event.id)
            ).filter(Event.experiment_id ==
exp.id).group_by(Event.variation, Event.event_type).all()

    data = {"A": {"view": 0, "click": 0}, "B": {"view": 0, "click":
0}}
    for variation, event_type, count in counts:
        if variation in data and event_type in data[variation]:
            data[variation][event_type] = count

    views_a = data["A"]["view"]
    clicks_a = data["A"]["click"]
```

					ПГУ 1.09.03.01.08.001 ПЗ	Лист
						80
Изм.	Лист	№ докум.	Подпись	Дата		

```

views_b = data["B"]["view"]
clicks_b = data["B"]["click"]

if exp.algorithm == "thompson":
    alpha_a, beta_a = clicks_a + 1, (views_a - clicks_a) + 1
    alpha_b, beta_b = clicks_b + 1, (views_b - clicks_b) + 1

    sample_a = random.betavariate(alpha_a, beta_a)
    sample_b = random.betavariate(alpha_b, beta_b)

    return "B" if sample_b > sample_a else "A"

if exp.algorithm == "epsilon_greedy" or True:
    epsilon = getattr(exp, "epsilon", 0.1)

    if random.random() < epsilon:
        return "A" if random.random() < 0.5 else "B"

    cr_a = clicks_a / views_a if views_a > 0 else 0
    cr_b = clicks_b / views_b if views_b > 0 else 0

    if cr_a == cr_b:
        return "A" if random.random() < 0.5 else "B"

    return "B" if cr_b > cr_a else "A"

```

Файл analytics.py

```

from scipy import stats
import pandas as pd

import math
from scipy import stats

def calculate_ab_results(count_a, clicks_a, count_b, clicks_b):

```

					ПГУ 1.09.03.01.08.001 ПЗ	Лист
						81
Изм.	Лист	№ докум.	Подпись	Дата		

```

if count_a < 5 or count_b < 5:
    return {"error": "Insufficient data"}

cr_a = clicks_a / count_a
cr_b = clicks_b / count_b

se_a = math.sqrt(cr_a * (1 - cr_a) / count_a)
se_b = math.sqrt(cr_b * (1 - cr_b) / count_b)

ci_a = [max(0, cr_a - 1.96 * se_a), min(1, cr_a + 1.96 * se_a)]
ci_b = [max(0, cr_b - 1.96 * se_b), min(1, cr_b + 1.96 * se_b)]

combined_se = math.sqrt(se_a**2 + se_b**2)
z_score = (cr_b - cr_a) / combined_se if combined_se > 0 else
0
p_value = stats.norm.sf(abs(z_score)) * 2
is_significant = bool(p_value < 0.05)

lift_value = round(((cr_b - cr_a) / cr_a * 100), 2) if cr_a >
0 else 0.0
lift = float(lift_value)

return {
    "cr_a": round(cr_a * 100, 2),
    "cr_b": round(cr_b * 100, 2),
    "ci_a": [round(x * 100, 2) for x in ci_a],
    "ci_b": [round(x * 100, 2) for x in ci_b],
    "p_value": float(p_value),
    "is_significant": is_significant,
    "lift": lift
}

```

Файл simulator.py

```
import asyncio
```

					ПГУ 1.09.03.01.08.001 ПЗ	Лист
						82
Изм.	Лист	№ докум.	Подпись	Дата		

```

import httpx
import random
import uuid

API_URL = "http://127.0.0.1:8000"
EXP_NAME = "Алгоритм Томпсона"
TOTAL_USERS = 1000
CONCURRENCY = 20

TRUE_CR_A = 0.05
TRUE_CR_B = 0.08

DEVICES = ["Desktop", "Mobile", "Tablet"]
DEVICE_WEIGHTS = [0.60, 0.30, 0.10]

async def simulate_user(client, semaphore):
    async with semaphore:
        user_id = f"sim_{uuid.uuid4().hex[:8]}"

        device_type = random.choices(DEVICES,
weights=DEVICE_WEIGHTS, k=1)[0]

        try:

            test_url = "http://localhost/"

            resp = await
client.get(f"{API_URL}/client_config?url={test_url}&user_id={user
_id}")

            if resp.status_code != 200:
                return

            config_data = resp.json()

            if not config_data.get("active_experiments"):

```

									Лист
Изм.	Лист	№ докум.	Подпись	Дата	ПГУ 1.09.03.01.08.001 ПЗ				
					83				

```

        return

        exp_config = next((e for e in
config_data["active_experiments"] if e["name"] == EXP_NAME), None)
        if not exp_config:
            return

        variant = exp_config["variant"]

        await client.post(f"{API_URL}/log_event", json={
            "user_id": user_id,
            "experiment_name": EXP_NAME,
            "variation": variant,
            "event_type": "view",
            "device_type": device_type
        })

        threshold = TRUE_CR_A if variant == "A" else TRUE_CR_B
        if random.random() < threshold:
            await client.post(f"{API_URL}/log_event", json={
                "user_id": user_id,
                "experiment_name": EXP_NAME,
                "variation": variant,
                "event_type": "click",
                "device_type": device_type
            })

    except Exception as e:
        pass

async def main():
    semaphore = asyncio.Semaphore(CONCURRENCY)
    async with httpx.AsyncClient(timeout=10) as client:
        tasks = [simulate_user(client, semaphore) for _ in
range(TOTAL_USERS)]
        print(f"Запуск симуляции {TOTAL_USERS} пользователей с
разных устройств...")

```

разных устройств...")					ПГУ 1.09.03.01.08.001 ПЗ	Лист 84
Изм.	Лист	№ докум.	Подпись	Дата		

```

        await asyncio.gather(*tasks)
        print("Симуляция успешно завершена!")

if __name__ == "__main__":
    asyncio.run(main())

```

Файл ab_platform.js

```

(function () {
    if (window.__ab_platform_initialized) return;
    window.__ab_platform_initialized = true;

    const scriptEl = document.currentScript;
    let SERVER_URL = "http://127.0.0.1:8000";

    if (scriptEl && scriptEl.src) {
        const urlObj = new URL(scriptEl.src);
        SERVER_URL = urlObj.origin;
    }

    function getDeviceType() {
        const ua = navigator.userAgent;

        if
        (/ (tablet|ipad|playbook|silk) | (android(?!. *mobi)) /i.test(ua))
        return "Tablet";

        if
        (/ Mobile|Android|iP(hone|od) | IEMobile|BlackBerry|Kindle|Silk-
Accelerated| (hpw|web) OS|Opera M(obi|ini) /i.test(ua)) return
        "Mobile";

        return "Desktop";
    }

    const deviceType = getDeviceType();

    const userId = localStorage.getItem('ab_user_id') || 'u_' +
    Math.random().toString(36).substr(2, 9);
    localStorage.setItem('ab_user_id', userId);

```

					ПГУ 1.09.03.01.08.001 ПЗ	Лист
						85
Изм.	Лист	№ докум.	Подпись	Дата		

```

    async function initAB() {
        try {
            const currentUrl = window.location.href;

            const res = await
fetch(`${SERVER_URL}/client_config?url=${encodeURIComponent(currentUrl)}&user_id=${userId}`);
            if (!res.ok) return;
            const data = await res.json();

            if (!data.active_experiments ||
data.active_experiments.length === 0) return;

            data.active_experiments.forEach(exp => {
                const variant = exp.variant;

                if (variant === 'B' && exp.changes)
{
                    exp.changes.forEach(change => {
                        document.querySelectorAll(change.selector
).forEach(el => {
                            try {
                                const settings =
JSON.parse(change.value_b);
                                if (settings.text) el.innerText =
settings.text;
                                if (settings.color)
el.style.backgroundColor = settings.color;
                                if (settings.width) el.style.width
= settings.width + 'px';
                                if (settings.height) el.style.height
= settings.height + 'px';
                            } catch (e) {
                                el.innerText = change.value_b;
                            }
                        })
                    })
                }
            });

```

					ПГУ 1.09.03.01.08.001 ПЗ	Лист
						86
Изм.	Лист	№ докум.	Подпись	Дата		

```

        });
    }

    if (exp.changes) {
        exp.changes.forEach(change => {
            document.querySelectorAll(change.selector)
                .forEach(el => {
                    if (!el.dataset.abTracked) {
                        el.addEventListener('click', function
(event) {
                            if (el.tagName === 'A' ||
el.tagName === 'BUTTON') {
                                event.preventDefault();

                                fetch(`${SERVER_URL}/log_event`,
{
                                    method: 'POST',
                                    headers: { 'Content-Type':
'application/json' },
                                    body: JSON.stringify({
                                        user_id: userId,
                                        experiment_name:
exp.name,
                                        variation: variant,
                                        event_type: "click",
                                        device_type: deviceType
                                    })
                                })
                                    .then(() => console.log(`Click
logged for ${exp.name}`))
                                    .catch(err =>
console.error(err));
                            });
                            el.dataset.abTracked = "true";
                        }
                    }
                });
        });
    }
}

```

					ПГУ 1.09.03.01.08.001 ПЗ	Лист
						87
Изм.	Лист	№ докум.	Подпись	Дата		

```

        });
    });
}

const viewKey = `ab_view_sent_${exp.name}`;
if (!sessionStorage.getItem(viewKey)) {
    fetch(`${SERVER_URL}/log_event`, {
        method: 'POST',
        headers: { 'Content-Type': 'application/json'
    },
        body: JSON.stringify({
            user_id: userId,
            experiment_name: exp.name,
            variation: variant,
            event_type: "view",
            device_type: deviceType
        })
    })
    .then(() => {
        sessionStorage.setItem(viewKey, "true");
        console.log(`View logged for
        ${exp.name}`);
    })
    .catch(err => console.error(err));
}

});
} catch (e) {
    console.error("A/B Platform error:", e);
}
}

if (document.readyState === 'complete' || document.readyState
=== 'interactive') {
    initAB();
} else {

```

document.addEventListener('DOMContentLoaded', initAB);

ПГУ 1.09.03.01.08.001 ПЗ

Лист

88

Изм.	Лист	№ докум.	Подпись	Дата

}
}) () ;

Приложение В

(Обязательное)

Приложение А/В тестирования пользовательских интерфейсов

сайтов

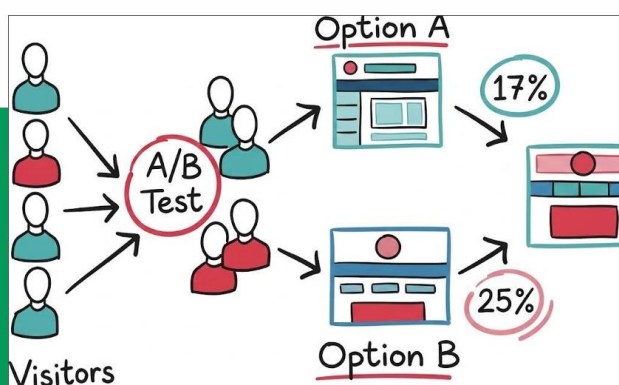
Презентация

					ПГУ 1.09.03.01.08.001 ПЗ	Лист
						90
Изм.	Лист	№ докум.	Подпись	Дата		

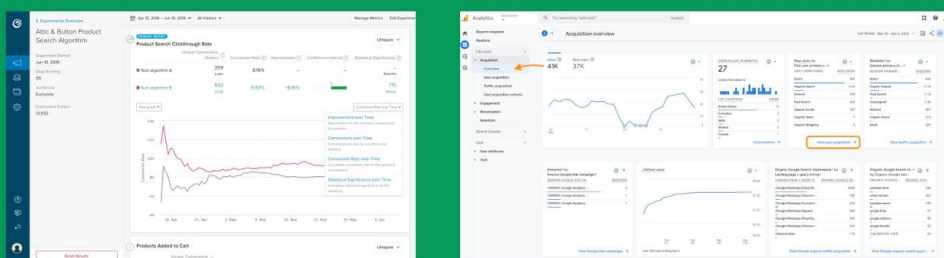
ПРИЛОЖЕНИЕ А/В ТЕСТИРОВАНИЕ ПОЛЬЗОВАТЕЛЬСКИХ ИНТЕРФЕЙСОВ САЙТОВ

- Автор работы: студент гр. 22ВВВ1 Лосяков Г.С.
- Научный руководитель: д.т.н. Митрохин М.А.

ЧТО ТАКОЕ А/В
ТЕСТИРОВАНИЕ?



РАЗБОР АНАЛОГОВ



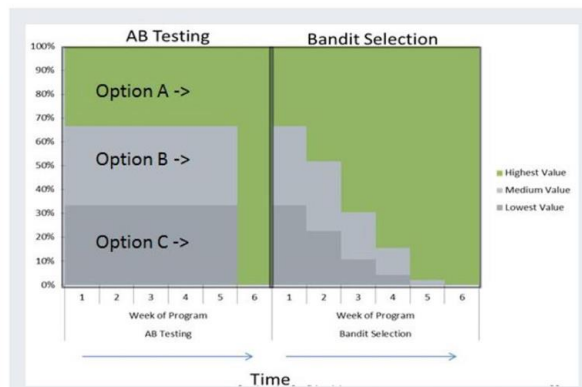
Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.09.03.01.08.001 ПЗ

Лист

91

АЛГОРИТМЫ РАСПРЕДЕЛЕН ИЯ ТРАФИКА



$$\text{Конверсия} = \frac{\text{Количество кликов}}{\text{Количество просмотров}} \times 100\%$$

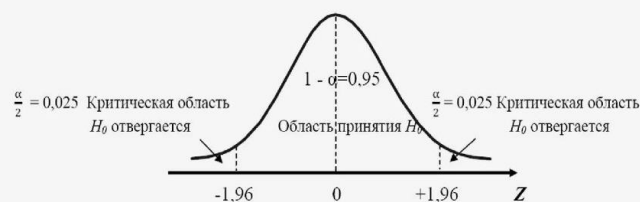


ДИАГРАММА ПОСЛЕДОВАТЕЛЬНОСТИ

