МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«Тольяттинский государственный университет»

Кафедра «Прикладная математика и информатика»
(наименование)

09.03.03 Прикладная информатика
(код и наименование направления подготовки, специальности)

Разработка программного обеспечения
(направленность (профиль) / специализация)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
(БАКАЛАВРСКАЯ РАБОТА)

на тему «Разработка сайта ИТ-компании на основе машинного обучения»

Обучающийся А.В. Балаев
(Инициалы Фамилия) (личная подпись)

Руководитель Доктор социологических наук, доцент, Е. В. Желнина
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Консультант кандидат фил. наук, доцент, М.В. Дайнеко
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Тольятти 2025

Аннотация

Тема выпускной квалификационной работы: «Разработка сайта ИТ-компании на основе машинного обучения».

Целью работы является разработка сайта ИТ-компании, предоставляющего услуги CRM и HRM систем, использующего алгоритмы машинного обучения для анализа активности пользователей, предсказания оттока клиентов и предоставления персонализированных уведомлений, направленных на повышение удержания клиентов.

Объект исследования – сайт ИТ-компании, предоставляющий услуги CRM и HRM систем, использующий машинное обучение для предсказания оттока клиентов. Предмет исследования – процесс разработки сайта ИТ-компании, использующего машинное обучение для предсказания оттока клиентов и предоставления персонализированных уведомлений.

Бакалаврская работа состоит из введения, трех глав, заключения, списка используемой литературы.

Во введении формируется актуальность, цель работы, объект и предмет исследования, задачи, методы исследования.

В первой главе проводится анализ предметной области и ставится задача на разработку системы. Проводится анализ функций и возможностей аналогов программного обеспечения, описываются инструменты, используемые в процессе разработки, определяются функциональные требования к системе. Во второй главе описывается процесс разработки архитектуры системы, проектирования и построения базы данных MongoDB. В третьей главе описывается процесс разработки клиентской и серверной части сайта ИТ-компании и механизма отслеживания активности клиентов, разработки и интеграции сервиса машинного обучения, разработки панели администратора и системы уведомлений, а также тестирования системы.

Бакалаврская работа состоит из 75 страниц и включает 46 рисунков, 7 таблиц, 27 источников.

Abstract

The title of the graduation work is Developing an IT company website based on machine learning. The aim of this graduation work is to develop a website of an IT company providing CRM and HRM systems services, using the machine learning algorithms to analyze the users' activity, predict the customer churn, and provide personalized notifications aimed at increasing the customer retention.

The object of the graduation work is the website of the IT company.

The subject of the graduation work is the process of developing the website for the IT company.

The key issue of the graduation work is to solve the uncontrolled customer churn problem from the IT company's website using machine learning.

The graduation work may be divided into several logically connected parts which are setting the development task, analyzing the analogues, defining the functional requirements, selecting the tools, designing the architecture and the database, as well as developing and testing the system.

The first part conducts an analysis of the subject area, sets the task related to developing the system, analyzes the software analogues, defines the functional requirements and selects the tools for developing the system.

The second part dwells on the results of developing the system architecture, as well as designing and constructing the MongoDB database.

The third part describes the process of developing the client and server parts of the IT company's website and the mechanism of tracking the customer activity. This part also considers the development and integration of a machine learning service, an administrator panel and a notification system, as well as functional, integration and load testing.

In conclusion, it should be noted that the IT company's developed website based on machine learning automates the user behaviour analysis, predicts the risk of their churn and provides personalized notifications, which increases the customer loyalty and the company's revenue due to the customer retention.

Оглавление

Введение

Прогнозирование оттока клиентов представляет собой одну из приоритетных задач в сфере деятельности ИТ-компаний, особенно в контексте предоставления услуг CRM и HRM систем, поскольку позволяет оперативно выявлять клиентов с высокой вероятностью ухода и реализовывать стратегии их удержания. В условиях растущей конкуренции и активного внедрения цифровых технологий разработка сайта ИТ-компании, использующего алгоритмы машинного обучения для прогнозирования оттока клиентов, становится актуальной задачей, способствующей повышению эффективности бизнес-процессов и укреплению позиций компании на рынке.

Актуальность данной работы обусловлена возрастающей потребностью ИТ-компаний в автоматизации анализа поведения пользователей и предсказания оттока клиентов, что позволяет своевременно принимать меры для их удержания благодаря использованию алгоритмов машинного обучения.

Объект исследования – сайт ИТ-компании, предоставляющий услуги CRM и HRM систем, использующий машинное обучение для предсказания оттока клиентов.

Предмет исследования – процесс разработки сайта ИТ-компании, использующего машинное обучение для предсказания оттока клиентов и предоставления персонализированных уведомлений.

Целью работы является разработка сайта ИТ-компании, предоставляющего услуги CRM и HRM систем, использующего алгоритмы машинного обучения для анализа активности пользователей, предсказания оттока клиентов и предоставления персонализированных уведомлений, направленных на повышение удержания клиентов.

Для достижения данной цели необходимо решить следующие задачи:

– Провести анализ предметной области и поставить задачу на разработку системы, провести анализ функций и возможностей аналогов программного обеспечения, определить инструменты для

разработки системы и определить функциональные требования к системе;

- Разработать архитектуру системы и спроектировать базу данных MongoDB;
- Реализовать клиентскую и серверную части сайта ИТ-компании, включая механизм отслеживания активности клиентов;
- Разработать и интегрировать сервис машинного обучения;
- Разработать панель администратора и систему отправки уведомлений;
- Провести тестирование системы, включая функциональное, интеграционное и нагрузочное тестирование.

Выпускная квалификационная работа состоит из введения, трёх глав, заключения и списка используемой литературы.

В работе применялись методы системного анализа для исследования предметной области и разработки архитектуры системы, методы машинного обучения для построения и обучения модели прогнозирования оттока, а также методы веб-разработки для реализации клиентской и серверной частей приложения. Для оценки системы использовались методы функционального, интеграционного и нагрузочного тестирования с использованием специализированных инструментов.

Глава 1 Постановка задачи на разработку и определение функциональных требований системы

1.1 Постановка задачи на разработку системы

Предметная область охватывает деятельность ИТ-компаний, специализирующихся на разработке и внедрении CRM и HRM систем, которые направлены на автоматизацию управления клиентами и персоналом. Одной из ключевых проблем данных ИТ-компаний является неконтролируемый отток клиентов, что приводит к снижению доходов и утрате конкурентных позиций на рынке. Использование машинного обучения позволяет анализировать поведение пользователей, выявлять клиентов с высокой вероятностью оттока и предоставлять персонализированные уведомления для их удержания, что делает разработку таких систем востребованной. Современный рынок ИТ-услуг характеризуется высоким спросом на системы, способные автоматизировать анализ данных и предоставлять персонализированные решения, что подчёркивает актуальность разработки сайта ИТ-компании с интегрированной моделью машинного обучения для прогнозирования оттока клиентов.

На основе проведённого анализа поставлена задача на разработку сайта ИТ-компании, предоставляющего услуги CRM и HRM систем, использующего алгоритмы машинного обучения для анализа активности пользователей, предсказания оттока клиентов и предоставления персонализированных уведомлений, направленных на повышение удержания клиентов.

1.2 Анализ аналогов программного обеспечения

В процессе анализа существующих программных решений были изучены платформы, использующие алгоритмы машинного обучения для анализа клиентской активности и разработки стратегий удержания. Рассмотрены сайты таких ИТ-компаний, как Freshworks, Zendesk и IBM.

Представленные сайты применяют машинное обучение для обработки данных клиентов, прогнозирование оттока и автоматизацию удержания клиентов, что показывает эффективность данного подхода в решении текущей задачи.

Сравнение аналогов проводилось по их функциональным характеристикам, включая возможности прогнозирования оттока, меры по удержанию клиентов и аналитическую обработку данных. Результаты анализа представлены в таблице 1.

Таблица 1 – Сравнительный анализ аналогов программного обеспечения

Функция/Решение	Freshworks	Zendesk	IBM
Предоставляемые услуги	SaaS-услуги, включая поддержку клиентов, продажи и управление ИТ-услугами.	Платформа для поддержки клиентов.	Аппаратное и программное обеспечение, ИТ-сервисы, консалтинговые услуги.
Прогнозирование оттока клиентов	Использует Freddy AI для анализа поведенческих данных клиентов и оценки вероятности их ухода.	Использует Zendesk AI для идентификации клиентов с повышенным риском оттока на основе взаимодействий.	Использует Watson для анализа клиентских данных с целью прогнозирования вероятности оттока.
Меры по удержанию клиентов	Обеспечивает автоматизированные рекомендации для удержания клиентов.	Реализует автоматизацию ответов и предоставляет инструменты для персонализированных стратегий удержания.	Предоставляет прогноз оттока клиентов и рекомендации по их удержанию.
Аналитическая обработка данных клиентов	Обеспечивает обработку данных клиентов для аналитических целей через Freddy AI. Предоставляет панель управления и аналитические отчеты в реальном времени.	Выполняет анализ данных клиентов для персонализации и аналитики с использованием Sunshine. Обеспечивает аналитические инструменты с панелями управления и средствами оценки качества взаимодействия.	Осуществляет обработку больших объемов клиентских данных через Watson. Реализует глубокую аналитику данных через Watson Analytics.

На основании проведённого анализа можно сделать вывод, что сайты ИТ-компаний Freshworks, Zendesk и IBM демонстрируют эффективность применения машинного обучения для прогнозирования оттока клиентов и

реализации мер по их удержанию, однако они не являются точными аналогами разрабатываемой системы, так как в полной мере не решают поставленные задачи и не реализуют необходимый функционал. Это обуславливает разработку сайта ИТ-компании, использующего алгоритмы машинного обучения для прогнозирования оттока клиентов и предоставления персонализированных уведомлений для их удержания.

1.3 Функциональные требования к разработке системы

Для разработки системы были определены функциональные и нефункциональные требования по методологии FURPS+. Функциональные требования представляют собой описание конкретных функций и возможностей разрабатываемой системы, определяющих, что она должна делать. Нефункциональные требования описывают качества и характеристики системы, такие как удобство использования, надёжность, производительность, требования к поддержке и проектные ограничения. Требования к разрабатываемой системе представлены в таблице 2.

Таблица 2 – Требования к системе прогнозирования оттока клиентов по методологии FURPS+.

Требование	Статус	Полезность	Риск	Стабильность
Функциональность				
Обеспечение интерфейса сайта ИТ-компании и обработки HTTP-запросов API-сервером.	Допустимый	Критическая	Низкий	Высокая
Регистрация и аутентификация пользователя на сайте ИТ-компании.	Допустимый	Критическая	Средний	Высокая
Предоставление функционала для просмотр списка решений, оформления подписки на решение, создания заявки на консультацию, возможности оставить отзыв на решение.	Допустимый	Критическая	Низкий	Высокая

Продолжение таблицы 2

Требование	Статус	Полезность	Риск	Стабильность
Реализация механизма сбора данных о	Допуст	Критич	Низк	Высока

взаимодействиях клиентов с сайтом ИТ-компании.	имый	еская	ий	я
Обеспечение работы трёх фоновых сервисов для обработки клиентских данных, постоянной тренировки ML-модели на актуальных данных и расчёта актуальной вероятности оттока клиентов.	Допустимый	Критическая	Средний	Высокая
Точность предсказаний ML-модели не менее 85% по метрикам AUC-ROC, F1-score, Accuracy.	Допустимый	Критическая	Низкий	Высокая
Панель администратора для анализа метрик взаимодействия клиентов с системой и возможности отправки персонализированных уведомлений.	Допустимый	Критическая	Средний	Высокая
Удобство использования				
Интерфейс панели администратора должен быть интуитивным, иметь возможность просмотра и анализа метрик взаимодействия клиентов и предоставлять механизм отправки персонализированных уведомлений.	Допустимый	Критическая	Низкий	Высокая
Надёжность				
Обеспечение обработки и логирования ошибок.	Допустимый	Критическая	Низкий	Высокая
Устойчивость к пиковым нагрузкам до 10 000 пользователей без потери данных или сбоев.	Допустимый	Критическая	Низкий	Высокая
Производительность				
Сервер должен обрабатывать 10000 HTTP GET-запросов, отправленных за 5 секунд, со средним временем ответа не более 2 секунд.	Допустимый	Критическая	Средний	Высокая
Получение предсказаний для 10000 клиентов за время, не превышающее 3 минуты.	Допустимый	Критическая	Средний	Высокая
Обучение модели машинного обучения на 10000 записей за время, не превышающее 10 минут.	Допустимый	Важная	Средний	Высокая
Проектные ограничения				
Использование не реляционной NoSQL базы данных MongoDB.	Допустимый	Важная	Низкий	Высокая
Разработка серверной части и сервиса машинного обучения на платформе ASP.NET Core с использованием библиотеки Microsoft.ML для работы с моделью машинного обучения.	Допустимый	Критическая	Средний	Высокая
Разработка клиентской части с использованием библиотеки React.	Допустимый	Важная	Низкий	Высокая

Функциональные и нефункциональные требования, представленные в таблице 2 демонстрируют ключевые функции и качества разрабатываемой системы. Они обеспечивают основу для создания масштабируемой и эффективной системы.

1.4 Используемые инструменты, библиотеки и средства

Для разработки системы были выбраны инструменты, библиотеки и средства, обеспечивающие высокую производительность, масштабируемость и удобство разработки и тестирования. Использованные технологии охватывают языки программирования, фреймворки, базу данных, средства разработки, библиотеки для работы с моделью машинного обучения, инструменты и библиотеки для тестирования.

Языки программирования и фреймворки:

- C# – высокоуровневый язык программирования от Microsoft, предназначенный для создания безопасных и производительных приложений;
- ASP.NET Core – кроссплатформенный фреймворк от Microsoft для построения веб-приложений и API;
- JavaScript – язык программирования для создания интерактивных веб-приложений;
- React – JavaScript-библиотека для построения пользовательских интерфейсов, использующая компонентный подход для создания динамичных клиентских приложений.

Инструменты разработки и документации:

- Microsoft Visual Studio – интегрированная среда разработки (IDE) от Microsoft;
- Microsoft Visual Studio Code – лёгкий редактор исходного кода, разработанный Microsoft;
- Git – система контроля версий;
- GitHub – платформа для хостинга репозиториев;
- Swashbuckle.AspNetCore.Swagger, Swashbuckle.AspNetCore.SwaggerGen, Swashbuckle.AspNetCore.SwaggerUI – набор библиотек для автоматической генерации документации API.

Библиотеки и инструменты для машинного обучения:

– Microsoft.ML – библиотека машинного обучения от Microsoft для создания, обучения и развёртывания ML-моделей;
– Microsoft.ML.LightGbm – расширение для библиотеки Microsoft.ML, добавляющее поддержку алгоритма LightGBM;
– Bogus – библиотека для генерации синтетических тестовых данных;

Базы данных:

– MongoDB – документоориентированная NoSQL база данных, используемая для хранения пользовательских данных;
– MongoDB.Driver – официальный драйвер для работы с MongoDB в коде C#;
– Mongo2Go – библиотека для запуска локального экземпляра MongoDB в тестовой среде.

Инструменты для тестирования:

– FluentAssertion – библиотека с набором методов расширения, которые позволяют более естественно указывать ожидаемый результат модульных тестов;
– Moq – инструмент для создания мок-объектов в тестах;
– xUnit – фреймворк для модульного тестирования;
– Microsoft.NET.Test.Sdk, Microsoft.TestPlatform – инструменты для поддержки тестовой инфраструктуры в .NET, обеспечивающие запуск и управление тестами;
– Microsoft.AspNetCore.Mvc.Testing – библиотека для тестирования ASP.NET Core приложений, позволяющая эмулировать HTTP-запросы к API;
– Apache JMeter – инструмент для проведения нагрузочного тестирования, разрабатываемый Apache Software Foundation.

Приведённый набор инструментов и библиотек обеспечивает эффективную разработку системы, включая этапы проектирования, разработки и тестирования. Использование данных технологий позволит

создать надёжную, масштабируемую и функциональную систему, соответствующую поставленным требованиям.

Выводы по главе 1

В результате анализа предметной области поставлена задача на разработку сайта ИТ-компании, предоставляющего услуги CRM и HRM систем, использующего алгоритмы машинного обучения для анализа активности пользователей, предсказания оттока клиентов и предоставления персонализированных уведомлений, направленных на повышение удержания клиентов. Анализ аналогов сайтов ИТ-компаний Freshworks, Zendesk и IBM, применяющих машинное обучение для обработки данных клиентов, прогнозирование оттока, автоматизацию удержания клиентов выявил их ключевые функции и определил, что они в полной мере не решают поставленные задачи и не реализуют необходимый функционал системы. Это обусловило разработку собственного решения. Были определены функциональные и нефункциональные требования к разрабатываемой системе по методологии FURPS+.

Глава 2 Проектирование системы

2.1 Проектирование архитектуры системы

Для наглядного представления архитектуры системы была разработана диаграмма развёртывания, которая иллюстрирует взаимодействие между её ключевыми компонентами. Данная диаграмма позволяет понять распределение функциональных элементов системы, а также их взаимодействие в процессе сбора, обработки данных, обучения ML-модели, получения предсказаний оттока. Диаграмма представлена на рисунке 1 и отображает физическое размещение компонентов системы, их взаимосвязи, протоколы для передачи данных.

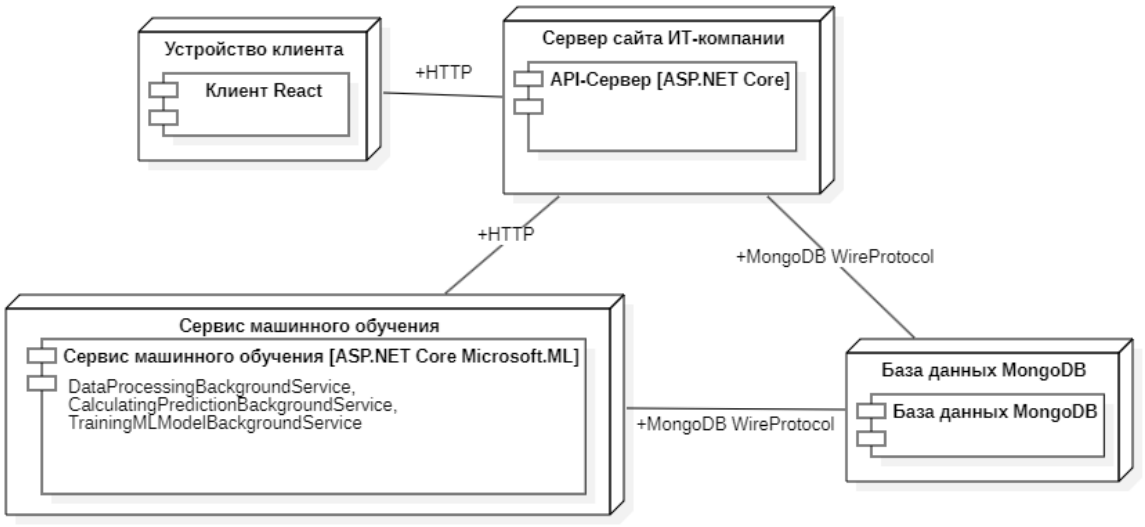


Рисунок 1 – Диаграмма развёртывания системы

Диаграмма развёртывания, показанная на рисунке 1, включает 4 основные компонента системы [2].

– Клиентское приложение представляет клиентскую часть системы, реализованную с помощью библиотеки React [26]. Компонент отвечает за визуализацию интерфейса сайта, сбор информации о

взаимодействиях клиентов с сайтом. Взаимодействует с API-сервер с помощью отправки HTTP-запросов;

- API-сервер представляет центральный компонент системы, обеспечивает обработку запросов от клиентского приложения и обеспечивает передачу данных в сервис машинного обучения через HTTP-запросы [27]. Интегрирован с базой данных MongoDB [6];
- Сервис машинного обучения представляет компонент системы, отвечающий да обработку клиентских данных, обучение модели машинного обучения и получение предсказаний оттока клиентов [11]. Интегрирован с API-сервером и базой данных MongoDB;
- База данных MongoDB используется для хранения данных, обеспечивающих работу сайта, данных клиентов, детализированных метрик взаимодействия клиентов с системой и результатов предсказаний. Обеспечивает выполнение CRUD-операций над данными, инициируемых API-сервером и сервисом машинного обучения [6].

Разработанная диаграмма развёртывания демонстрирует физическую архитектуру системы, взаимодействие между 4 основными компонентами, протоколы для передачи данных.

Для описания логической структуры системы была разработана диаграмма компонентов, отражающая основные компоненты системы, их внутреннюю структуру, связи и зависимости. Диаграмма представлена на рисунке 2.

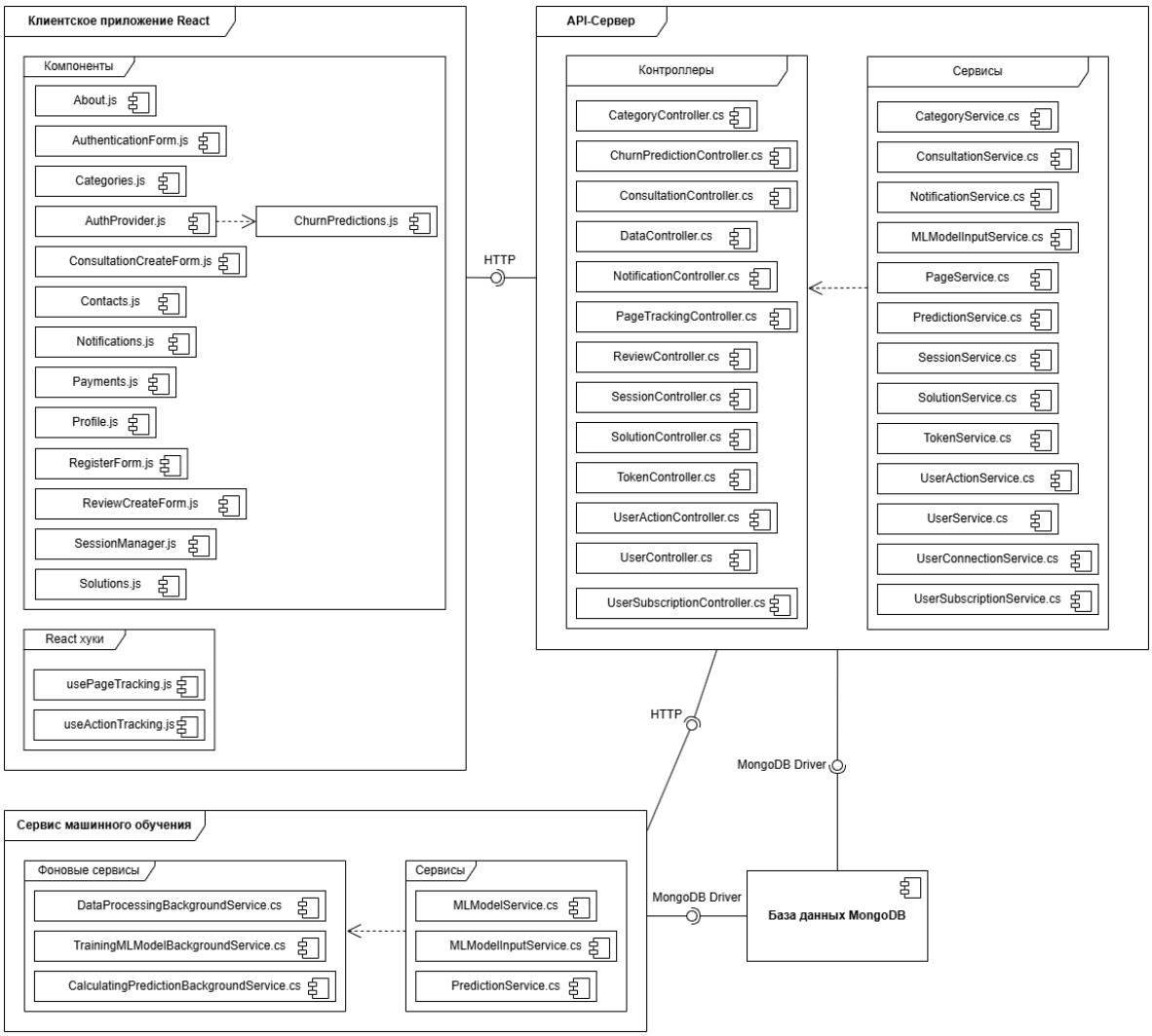


Рисунок 2 – Диаграмма компонентов системы

Представленная диаграмма компонентов включает 4 ключевых модуля, их внутреннюю структуру, взаимодействия и связи [2].

– Клиентское приложение React включает в себя компоненты для регистрации и аутентификации пользователей, отображения списка решений и категорий, представления информации о компании и контактов для связи, форму для отправки запросов на консультацию, а также компонент профиля пользователя, списка уведомлений, панели администратора. Также содержит React-хуки useActionTracking, usePageTracking для сбора данных о действиях клиентов и отслеживание их перемещениях по страницам сайта [16].

Клиентское приложение взаимодействует с API-сервером через HTTP-запросы [7];

- API-сервер предоставляет список контроллеров для обработки запросов с клиентского приложения и список соответствующих сервисов для выполнения CRUD-операций с данными в базе данных MongoDB. Также на сервере реализован механизм аутентификации и авторизации на основе JWT-токена [21];
- Сервис машинного обучения включает фоновые сервисы DataProcessingBackgroundService, TrainingMLModelBackgroundService, CalculatingPredictionBackgroundService и сервис MLModelInput, обеспечивающие обработку данных клиентов и работу с моделью машинного обучения для её тренировки и получения предсказаний [4];
- База данных MongoDB отвечает за хранение данных, обеспечивающих функциональность сайта, клиентских данных, детализированных метрик взаимодействия клиентов с системой и предсказаний оттока [6].

Разработанная диаграмма компонентов отражает логическую архитектуру системы, демонстрируя структуру её основных программных модулей, а также внутренние связи и взаимодействия между ними.

Для представления функциональных возможностей системы и описание взаимодействия с ней пользователей была разработана диаграмма вариантов использования. Диаграмма позволяет идентифицировать ключевых актёров, их цели и взаимосвязи между действиями. Диаграмма представлена на рисунке 3.

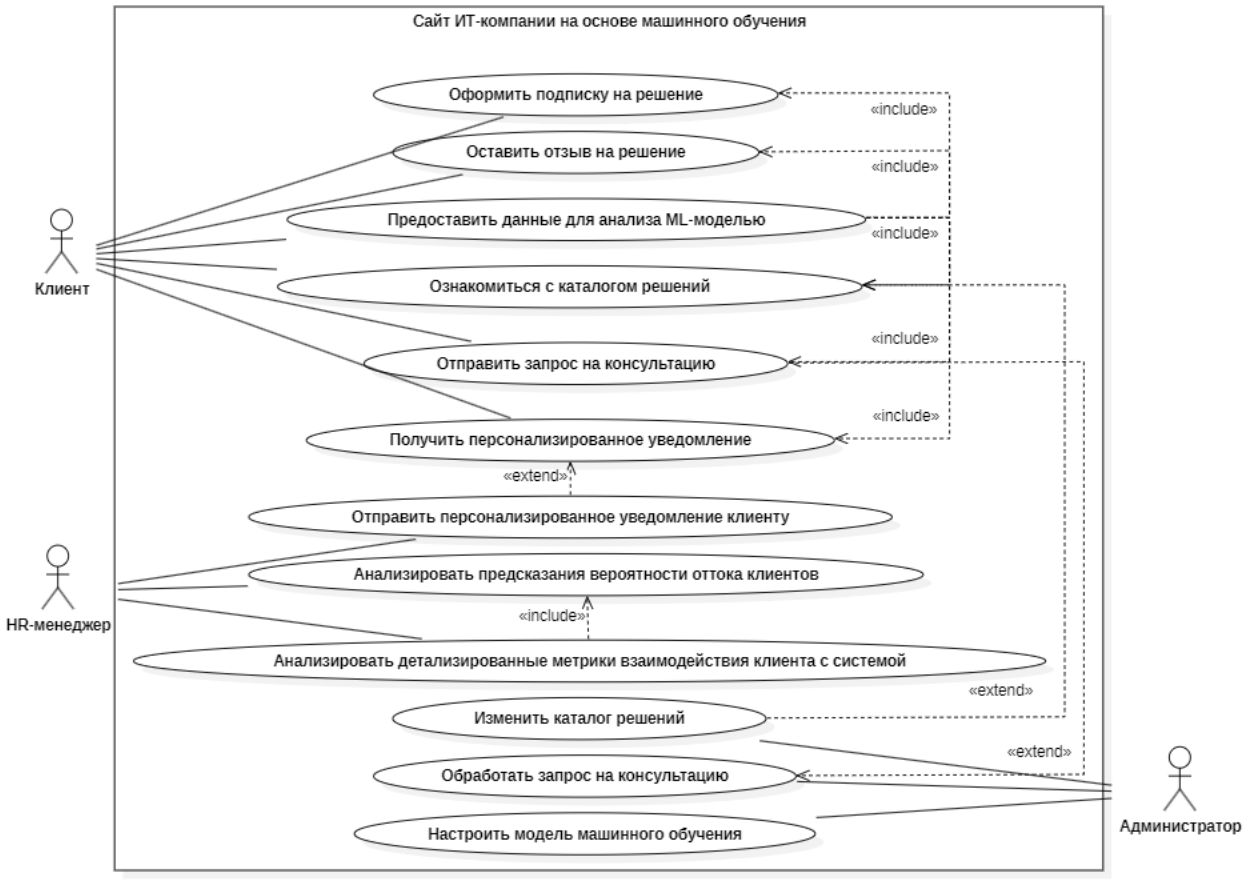


Рисунок 3 – Диаграмма вариантов использования системы

Диаграмма вариантов использования, показанная на рисунке 3, демонстрирует взаимодействие 3 актёров с системой, а также взаимосвязи между их действиями [2].

Клиент системы представляет собой обычного пользователя, который взаимодействует с системой для получения услуг и предоставления данных для анализа моделью машинного обучения. Действия клиента включают следующие сценарии:

- Клиент имеет возможность оформить подписку на одно из предлагаемых решений;
- Клиент имеет возможность ознакомиться с каталогом решений;
- Клиент имеет возможность отправить запрос на консультацию для получения дополнительной информации или помощи;

– Клиент имеет возможность оставить отзыв о решении;
– Клиент, взаимодействуя с сайтом, предоставляет данные для анализа моделью машинного обучение. Данный сценарий включает все остальные представленные действия клиента;
– Клиент имеет возможность получать персонализированные уведомления от системы. Данный сценарий расширяется условием: если HR-менеджер отправляет персональное уведомление, клиент его получает.

HR-менеджер представляет собой сотрудника компании, ответственного за анализ оттока клиентов и взаимодействие с ними. Действия HR-менеджера включают следующие сценарии:

– HR-менеджер имеет возможность отправлять персонализированные уведомления клиентам;
– HR-менеджер имеет возможность проводить анализ предсказаний вероятности оттока клиентов;
– HR-менеджер имеет возможность проводить анализ детализированных метрик взаимодействия клиентов с системой. Данный сценарий включает анализ предсказаний вероятности оттока клиентов;

Администратор представляет собой сотрудника компании, отвечающего за настройку системы и управление контентом на сайте. Действия администратора включают следующие сценарии:

– Администратор имеет возможность изменить каталог решений. Данный сценарий расширяет сценарий ознакомления с каталогом решений клиента;
– Администратор имеет возможность обработать запрос на консультацию. Данный сценарий расширяет сценарий отправки запроса на консультацию клиентом;

– Администратор имеет возможность настраивать модель машинного обучение.

Разработанная диаграмма вариантов использования отражает функциональные возможности системы, а также роли её пользователей и их сценарии. Она выделяет 3 основных актёров – клиента системы, HR-менеджера и администратора и описывает их действия в системе. Применение отношений include и extend помогают выявить зависимости между сценариями, что улучшает точность проектирования системы.

2.2 Проектирование и построение базы данных MongoDB

Для реализации системы была выбрана не реляционная NoSQL база данных MongoDB, так как она обеспечивает гибкость схемы данных, поддержку JSON-подобных документов, высокую эффективность обработки больших объёмов информации и лёгкую интеграцию с API-сервером с помощью драйвера MongoDB.Driver [18].

Для визуализации структуры данных системы была разработана физическая модель базы данных MongoDB. Она представлена на рисунке 4 и обеспечивает детализированное представление архитектуры базы данных и отношения между классами.

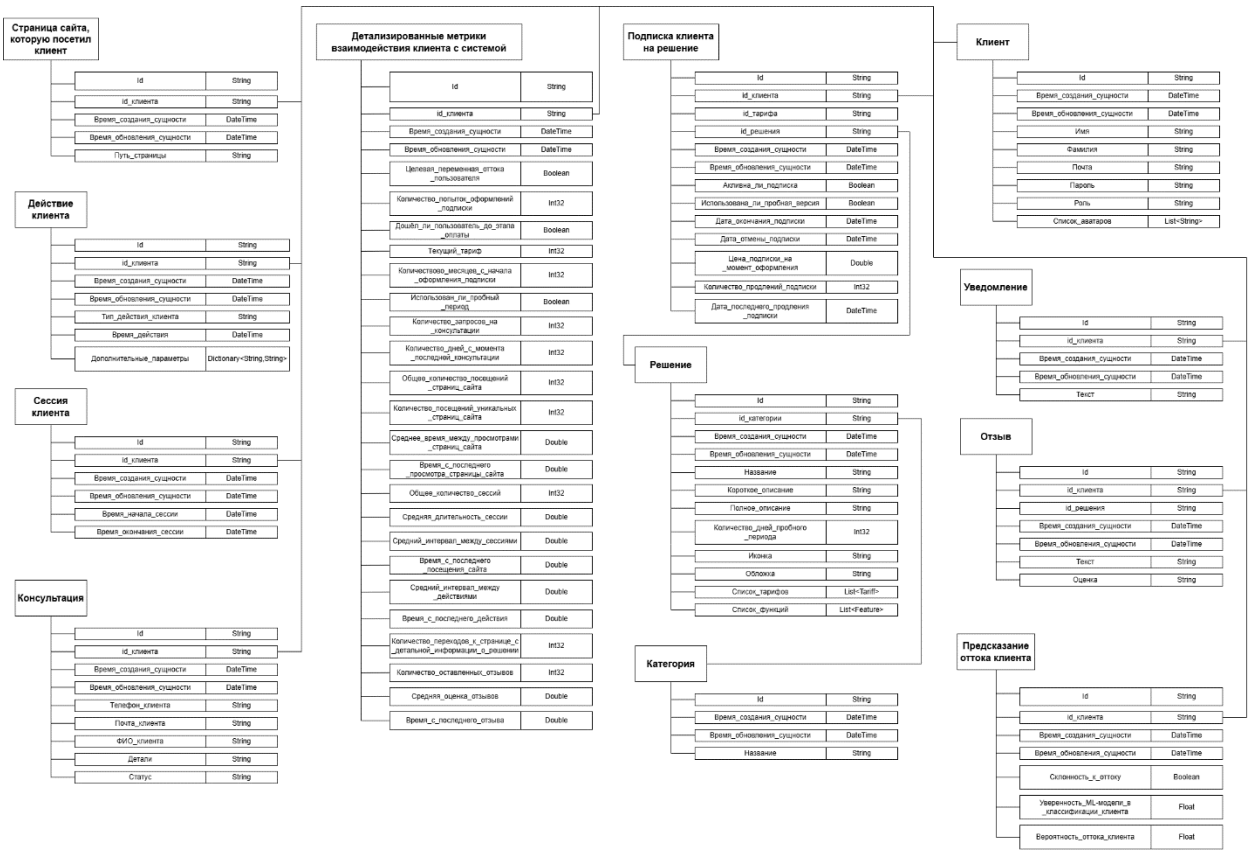


Рисунок 4 – Физическая модель базы данных MongoDB

Физическая модель состоит из 12 коллекций MongoDB, спроектированных для поддержки функциональности сайта ИТ-компании и сервиса машинного обучения. Ниже приведено описание каждой коллекции базы данных.

- Categories – коллекция, предназначенная для хранения данных о категориях решений;
- Consultations – коллекция, предназначенная для хранения информации о консультациях клиентов;
- MLModelInputs – коллекция, предназначенная для хранения детализированных метрик взаимодействий клиентов с системой;
- Notifications – коллекция, предназначенная для хранения уведомлений клиентов;

- Pages – коллекция, предназначенная для хранения страниц, которые посещали клиента на сайте ИТ-компании;
- Predictions – коллекция, предназначенная для хранения результатов предсказаний вероятности оттока клиентов;
- Reviews – коллекция, предназначенная для хранения отзывов клиентов о решениях;
- Sessions – коллекция, предназначенная для хранения сессий клиентов;
- Solutions – коллекция, предназначенная для хранения данных о решениях, доступных в системе;
- UserActions – коллекция, предназначенная для хранения действий пользователей на сайте компании;
- UserSubscriptions – коллекция, предназначенная для хранения подписок клиентов на решения;
- Users – коллекция, предназначенная для хранения данных клиентов системы.

Данная структура коллекций обеспечивает масштабируемость и гибкость, позволяя эффективно управлять данными, необходимыми для корректной работы сайта, анализа данных и предсказания оттока клиентов.

Для описания логической структуры базы данных разработана диаграмма классов, необходимая для моделирования и анализа иерархической организации данных в системе. Диаграмма классов, изображённая на рисунке 5, отражает логическую модель данных и показывает взаимосвязь между классами системы.

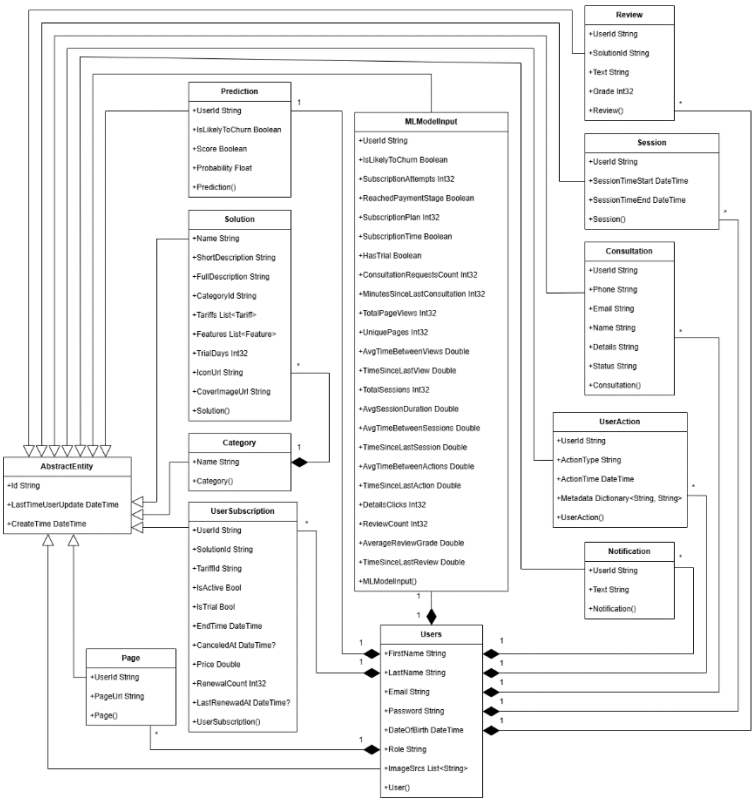


Рисунок 5 – Диаграмма классов системы

Диаграмма классов включает 1 базовый абстрактный класс AbstractEntity и 12 производных от него классов [22]. Классы наследуют 3 свойства базового класса [25].

– Id – идентификатор сущности типа данных String;
– CreateTime – дата и время создания сущности;
– LastTimeUserUpdate – дата и время последнего обновления сущности.

Также диаграмма отражает, что все классы, за исключением Solution и Category, связаны с классом User через поле UserId. Связь представляет собой отношение один ко многим или один к одному.

Диаграмма отображает следующие классы и их роли в системе:

– AbstractEntity – абстрактный базовый класс. Определяет базовый набор свойств для каждого класса наследника;
– Category – класс, представляющий категорию решения;
– Consultation – класс, представляющий консультацию клиента;

– MLModelInput – класс, представляющий детализированные метрики взаимодействия клиента с системой;
– Notification – класс, представляющий уведомление клиента;
– Page – класс, представляющий вкладку сайта, которую посетил клиент;
– Prediction – класс, представляющий предсказание оттока клиента;
– Review – класс, представляющий отзыв клиента на решение;
– Session – класс, представляющий сессию клиента;
– Solution – класс, описывающий решение, доступное на сайте компании;
– UserAction – класс, представляющий действие клиента на сайте;
– UserSubscription – класс, представляющий подписку клиента на решение;
– User – класс, представляющий клиента системы;

Диаграмма классов также иллюстрирует отношения между классами, типы данных полей, что обеспечивает полное представление о структуре данных и их взаимосвязях.

Для обеспечения эффективного функционирования системы был организован последовательный поток данных между её компонентами. Ниже представлено описание ключевых процессов передачи, обработки и хранения данных, охватывающих взаимодействие клиентского приложения, API-сервера, сервиса машинного обучения и базы данных.

– Клиентское приложение собирает данные клиентов в процессе их взаимодействия с системой с помощью механизмов сбора информации. Собранные данные передаются на API-сервер посредством отправки HTTP POST-запросов [7];
– API-сервер обрабатывает запросы и сохраняет полученные данные в соответствующие коллекции базы данных MongoDB [6];

– Фоновый сервис DataProcessingBackgroundService, предназначенный для обработки клиентских данных, каждый час отправляет HTTP GET-запросы к API-серверу для получения данных клиентов [20]. Сервер загружает данные из коллекций MongoDB, формирует объекты класса UserData, содержащие сведения о клиентах, сериализует в строковый формат и возвращаются в качестве ответа сервису машинного обучения [15];
– Фоновый сервис DataProcessingBackgroundService обрабатывает полученные от сервера данные, формируя экземпляры класса MLModelInputs, описывающие детализированные метрики взаимодействий клиентов с системой [27]. Обработанные данные затем сохраняются в базе данных, обеспечивая подготовку данных для обучения ML-модели;
– Фоновый сервис TrainingMLModelBackgroundService, выполняющий периодическую тренировку ML-модели каждые 4 часа, загружает данные из коллекции MLModelInputs базы дынных MongoDB и передаёт их в метод TrainModel для тренировки модели машинного обучения [11];
– Фоновый сервис CalculatingPredictionBackgroundService, предназначенный для прогнозирования актуальной вероятности оттока клиентов, раз в 30 минут получает данные из коллекции MLModelInputs базы дынных MongoDB [6]. Полученные данные преобразуются в класс MLModelInputDto, в котором числовые признаки приводятся к типу данных float, а целевой признак – поле IsLikelyToChurn к типу bool [22]. Преобразованные данные передаются в ML-модель для генерации предсказания. Предсказание представляет собой класс Prediction [11]. Результаты предсказаний сохраняются в коллекцию Predictions базы данных MongoDB [6];
– Визуализация данных осуществляется посредством отправки HTTP GET-запроса клиентским приложением к API-сервису для получения

актуальных данных о вероятности оттока [7]. API-сервис извлекает данные из коллекций Predictions базы данных MongoDB и возвращает в ответе клиентскому приложению. Для получения подробной информации об оттоке конкретного клиента клиентское приложение также отправляет HTTP GET-запросы к API-сервису, который загружает соответствующую запись из коллекций MLModelInputs MongoDB и возвращает в ответе. Полученные данные визуализируются в интерфейсе клиентского приложения [9].

Данный поток данных обеспечивает интегрированную и эффективную работу системы, позволяя собирать, обрабатывать, анализировать и визуализировать данные для корректности работы процесса прогнозирования оттока клиентов.

Выводы по главе 2

Проведено проектирование архитектуры сайта ИТ-компании, что обеспечивает основу для её дальнейшей реализации. Была разработана архитектура системы, представленная в виде диаграммы развёртывания, отражающей структуру компонентов и их взаимодействие, диаграмма компонентов, детально описывающая структуру системы, а также диаграмма вариантов использования, демонстрирующей функциональные возможности системы для разных ролей пользователей. Была спроектирована и построена база данных MongoDB, включая создание физической модели данных, диаграммы классов и описание потока данных в системе.

Глава 3 Разработка системы

3.1 Разработка клиентской и серверной части сайта ИТ-компании и механизма отслеживания активности клиентов

Клиентская часть сайта IT-компании реализована с использованием библиотеки React и обеспечивает динамичный интерфейс для взаимодействия с услугами CRM и HRM систем. Для обеспечения связи с сервером реализована отправка HTTP-запросов к API-серверу с помощью библиотеки Axios [7]. Данные запросы используются для получения данных, создания новых записей, изменения и удаления данных, обеспечивая полноценное взаимодействие с серверной частью. Авторизация осуществляется с помощью JWT-токена, который добавляется в заголовок Authorization каждого запроса, отправляемого на сервер [24]. На рисунке 6 представлен пример отправки HTTP GET-запроса к API-серверу для получения списка решений.

```
/**Загружает список решений */
const getAsync = async () => {
    try{
        setIsLoading(true);
        const response = await axios.get("https://localhost:7299/api/solution", {
            headers:{
                "Authorization": `Bearer ${token}`
            }
        });

        if(response.status && response.status === 200) {
           if(response.data && response.data.solutions) {
            setSolutions(response.data.solutions)
           }
        }
    }
    catch (error){
        await handleRequestError(error);
    } finally {
        setIsLoading(false);
    }
}
```

Рисунок 6 – Отправка HTTP GET-запроса для получения списка решений

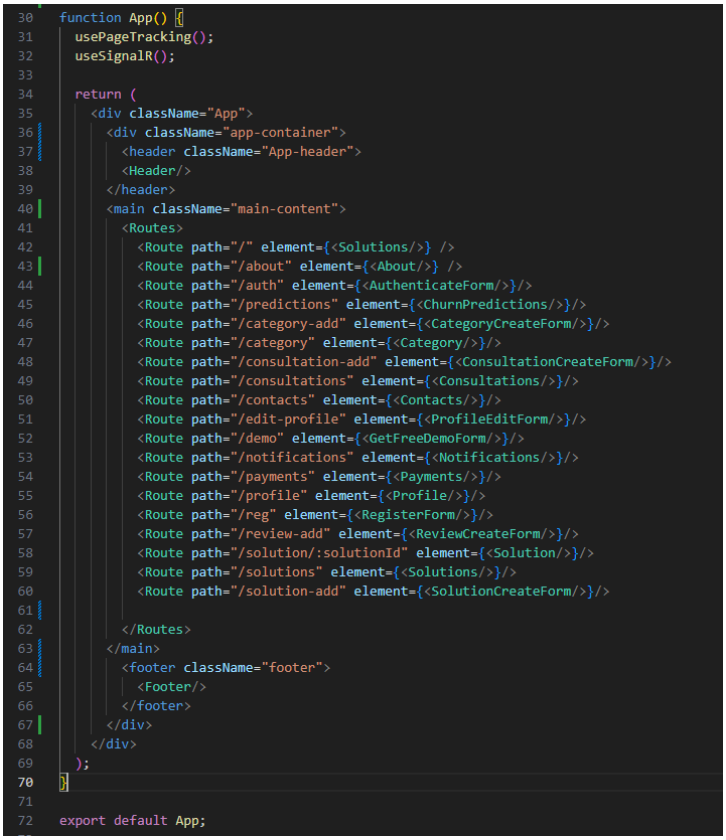
Для управления аутентификацией и предоставления глобального доступа к данным клиента в приложении разработан компонент AuthProvider [26]. Он создаёт контекст AuthContext, который позволяет компонентам

получать доступ к данным и методам аутентификации с помощью React-хука useContext. AuthProvider предоставляет методы для входа клиента на сайт, регистрации нового пользователя, получения данных клиента с сервера, обновления информации о клиенте, обновления JWT-токена при истечении его срока действия, а также для обработки ошибок HTTP-запросов [16]. На рисунке 7 представлен код компонент AuthProvider.

```
export const AuthProvider = ({children}) => {

    /** jwt-токен */
    const [token, setToken] = useState(localStorage.getItem("token") || "");
    /** пользователь*/
    const [user, setUser] = useState(null);
    /** обновлён ли токен или нет*/
    const [isRefreshing, setIsRefreshing] = useState(false);


    /**
     * Аутентификация пользователя
     * @param {*} email
     * @param {*} password
     */
    const login = async (email,password) => {
        try{
            var response = await axios.post("https://localhost:7299/api/user/auth",
                {
                    email: email,
                    password: password,
                });
            if(response.status === 200)
            {
                const authToken = response.data.token;

                const token = authToken.replace("Bearer", "")
                localStorage.setItem("token", token)

            }
        }
        catch (error){
            await handleRequestError(error);
        }
    };

    /**
     * Выйти из аккаунта
     */
    const logout = () => {
        setUser(null);
        setToken("");
        localStorage.removeItem("token");
    }

    /**
     * Регистрация пользователя
     * @param {*} firstName
     * @param {*} lastName
     * @param {*} email
     * @param {*} password
     */
    const register = async (firstName, lastName, email, password) => {
        try{
            var response = await axios.post("https://localhost:7299/api/user/reg",
                {
                    firstName: firstName,
                    lastName: lastName,
                    email: email,
                    password: password,
                });
            if(response.status === 200)
            {
                const authToken = response.data.token;

                const token = authToken.replace("Bearer", "")
                localStorage.setItem("token", token)
                window.location.reload();
            }
        }
        catch (error){
            await handleRequestError(error);
        }
    };

    /**
     * Получить информацию о пользователе
     */
    const getUser = async () => {
        try{
            if(token){
```

Рисунок 7 – Код компонента AuthProvider

React использует компонентный подход для построения интерфейса [16]. Главный компонент App.js, представленный на рисунке 8, определяет маршруты сайта с помощью библиотеки react-router-dom [17].



```jsx
function App() {
  usePageTracking();
  useSignalR();

  return (
    <div className="App">
      <div className="app-container">
        <header className="App-header">
        <Header/>
      </header>
      <main className="main-content">
        <Routes>
          <Route path="/" element={<Solutions/>} />
          <Route path="/about" element={<About/>} />
          <Route path="/auth" element={<AuthenticateForm/>}/>
          <Route path="/predictions" element={<ChurnPredictions/>}/>
          <Route path="/category-add" element={<CategoryCreateForm/>}/>
          <Route path="/category" element={<Category/>}/>
          <Route path="/consultation-add" element={<ConsultationCreateForm/>}/>
          <Route path="/consultations" element={<Consultations/>}/>
          <Route path="/contacts" element={<Contacts/>}/>
          <Route path="/edit-profile" element={<ProfileEditForm/>}/>
          <Route path="/demo" element={<GetFreeDemoForm/>}/>
          <Route path="/notifications" element={<Notifications/>}/>
          <Route path="/payments" element={<Payments/>}/>
          <Route path="/profile" element={<Profile/>}/>
          <Route path="/reg" element={<RegisterForm/>}/>
          <Route path="/review-add" element={<ReviewCreateForm/>}/>
          <Route path="/solution/:solutionId" element={<Solution/>}/>
          <Route path="/solutions" element={<Solutions/>}/>
          <Route path="/solution-add" element={<SolutionCreateForm/>}/>

        </Routes>
      </main>
        <footer className="footer">
          <Footer/>
        </footer>
      </div>
    </div>
  );
}

export default App;
```

Рисунок 8 – Маршруты сайта, определённые в компоненте App.js

В качестве примера рассмотрен компонент Solutions, отображающий список решений, загруженных с сервера. При монтировании Solutions отправляет HTTP GET-запрос к API-серверу для получения данных [7]. Если данные были успешно получены, список решений отображается в интерфейсе страницы. На рисунке 9 представлен код компонента Solutions, а на рисунке 10 представлен интерфейс сайта ИТ-компании с компонентом Solutions, демонстрирующий список решений [9].

```jsx
/**Компонент со списком решений */
const Solutions = () => {

    const {user, token, handleRequestError} = useContext(AuthContext);

    /**Список решений */
    const [solutions, setSolutions] = useState([]);
    const [isLoading, setIsLoading] = useState(true);
    const [error, setError] = useState(null);
    const [expandedSolutionId, setExpandedSolutionId] = useState(null);

    /**Метод для отслеживания действий пользователя */
    const { trackUserAction } = useActionTracking();

    /**Загружает список решений */
    const getAsync = async () => {
        try{
            setIsLoading(true);
            const response = await axios.get("https://localhost:7299/api/solution", {
                headers:{
                    "Authorization": `Bearer ${token}`
                }
            });

            if(response.status && response.status === 200) {
                if(response.data && response.data.solutions) {
                setSolutions(response.data.solutions)
                }
            }
        }
        catch (error){
            await handleRequestError(error);
        } finally {
            setIsLoading(false);
        }
    }

    /**Загружает список решений при монтировании компонента */
    useEffect(() => {
        getAsync();
    }, [])

    const formatPrice = (price) => {
        return new Intl.NumberFormat('ru-RU', {
            style: 'currency',
            currency: 'RUB',
            minimumFractionDigits: 0
        }).format(price);
    };

    const getCategoryName = async (categoryId) => {
        if(categoryId){
        const response = await axios.get(`https://localhost:7299/api/category/${categoryId}`, {
                headers:{
                    "Authorization": `Bearer ${token}`
                }
            });
            if(response.status && response.status === 200) {
                if(response.data && response.data.category) {
                    return response.data.category.name;
                }
            }
        }
    };

    /**Создаёт действие пользователя "Details" */
    const sendDetailsAction = async () => {
        await trackUserAction("Details", {
            userId: user.id
        });
    };

    return (
        <div className="container py-5">
    <div className="d-flex justify-content-between align-items-center mb-4">
        <h1 className="mb-0 fw-bold">Наши решения</h1>
        {user?.role === "Admin" &&
            <Link to="/solution-add" className="btn btn-success shadow-sm">
                <i className="bi bi-plus-circle me-2"></i> Добавить решение
            </Link>
        }
    </div>

    {isLoading ? (
        <div className="text-center py-5">
            <div className="spinner-border text-primary" role="status">
                <span className="visually-hidden">Загрузка</span>
            </div>
        </div>
```

Рисунок 9 – Код компонента Solutions

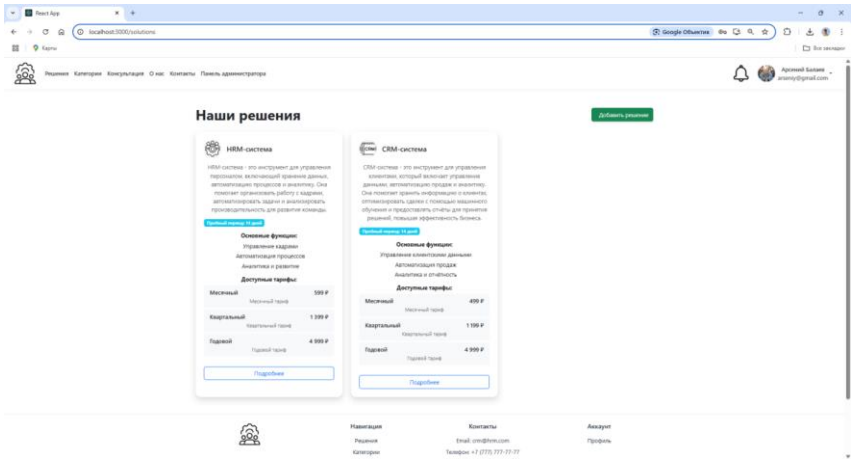


Рисунок 10 – Интерфейс сайта ИТ-компании с компонентом Solutions

Серверная часть сайта реализована на платформе ASP.NET Core и обеспечивает обработку запросов, интеграцию с базой данных, механизм

аутентификации и авторизации на основе JWT-токена [5]. Аутентификация и авторизация на основе JWT-токена на сайте реализована с помощью метода расширения AddAuthenticationServices и сервиса TokenService [21]. Метод расширения AddAuthenticationServices добавляет в DI-контейнер сервис аутентификации с помощью метода AddAuthentication, сервис авторизации с помощью метода AddAuthorization и сервис TokenService для работы с JWT-токеном [1]. Код метода расширения AddAuthenticationServices представлен на рисунке 11.

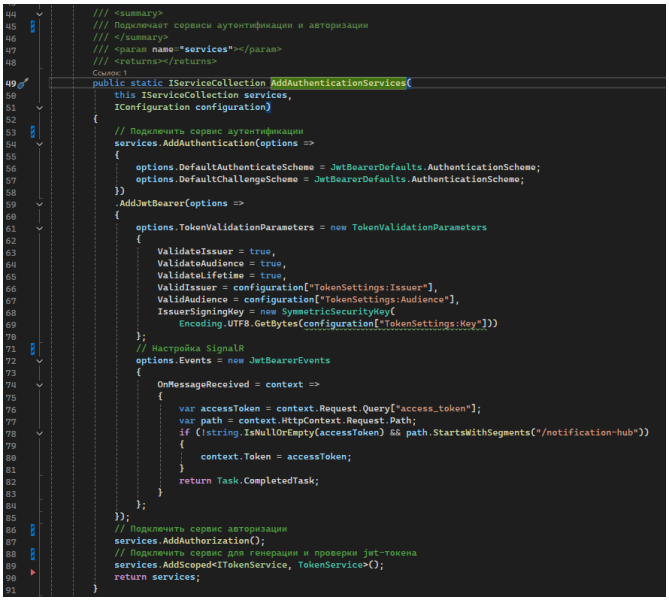


Рисунок 11 – Код метода расширения AddAuthenticationServices

Генерация токена выполняется методом GenerateJwtToken в сервисе TokenService, отвечающий за создание, обновление и проверку JWT-токена. Метод в качестве параметра принимает класс User и возвращает строку, содержащую JWT-токен [1]. Токен включает в себя Claims, содержащий идентификатор, почту и роль пользователя [21]. Также в сервисе представлены методы для генерации Refresh-токена, обновления JWT-токена и получения данных клиента из истёкшего JWT-токена [1].

Интеграция с базой данных MongoDB реализована с использованием драйвера MongoDB.Driver. Для взаимодействия с MongoDB используется сервис IMongoClient, который реализуется через класс MongoClient и регистрируется как singleton в DI-контейнере [18]. Регистрация осуществляется с использованием строки подключения, получаемой из файла конфигурации appsettings.json и представлена на рисунке 12.

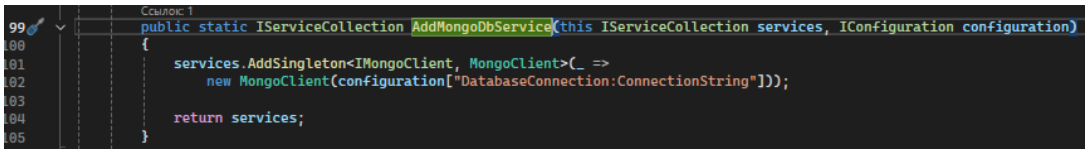


Рисунок 12 – Подключение сервиса MongoDB

Базовый сервис BaseService является базовым для всех остальных сервисов, обеспечивая унифицированный доступ, работу с данными в MongoDB и упрощения разработки наследуемых сервисов [27]. Он реализован как обобщённый класс, принимающий тип T, где T должен наследоваться от абстрактного класса AbstractEntity. Абстрактный класс AbstractEntity представляет базовый класс, определяющий 3 общих свойства для всех сущностей, обеспечивая единый формат данных в базе [25].

В качестве примера рассмотрен класс Solution и сервис SolutionService. Solution является наследником AbstractEntity и описывает сущность решения. Сервис SolutionService реализует логику для работы с объектами Solution, используя базовые методы, предоставляемые классом BaseService. На рисунке 13 приведён код сервиса SolutionService.

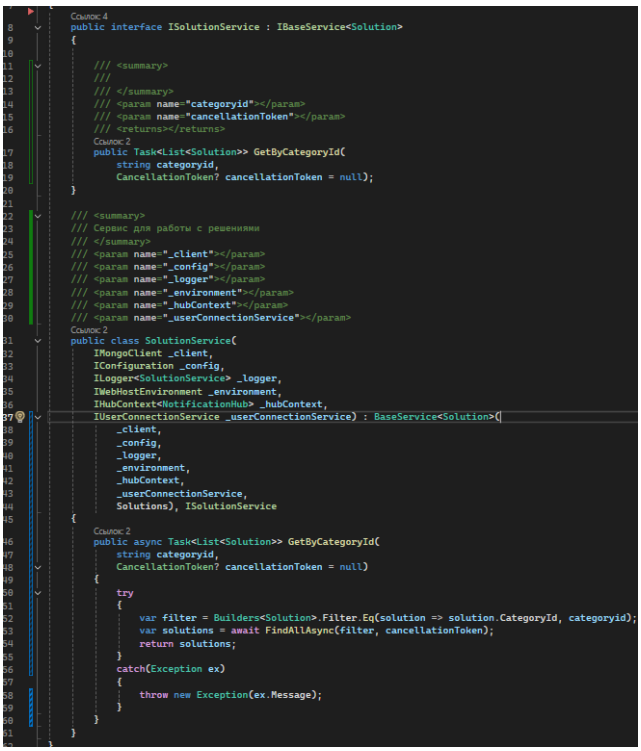


Рисунок 13 – Код сервиса SolutionService

В ASP.NET Core обработка HTTP-запросов осуществляется с использованием контроллеров – классов, отвечающих за приём, обработку и формирование ответов на запросы [20]. Контроллеры наследуются от базового класса Controller и взаимодействуют с сервисами бизнес-логики. В качестве примера использован контроллер SolutionController, обрабатывающий CRUD-операции над сущностью Solution с помощью сервиса SolutionService. Он помечен аннотацией [ApiController], которая указывает, что класс является API-контроллером, активирует автоматическую валидацию модели и привязка параметров [20]. Аннотация [Route] определяет базовый маршрут для всех действий данного контроллера. Для обработки запросов в контроллере применяются следующие аннотации:

- [HttpGet] – обработка GET-запросов;
- [HttpPost] – обработка POST-запросов;
- [HttpPut] – обработка PUT-запросов;
- [HttpDelete] – обработка DELETE-запросов;

– [Route(«{id}»)] – указание маршрута с параметром;

– [FromForm] – получение данных из формы.

На рисунке 14 приведён пример реализации метода GetAllAsync из SolutionController, демонстрирующий получение списка решений с помощью сервиса SolutionService.

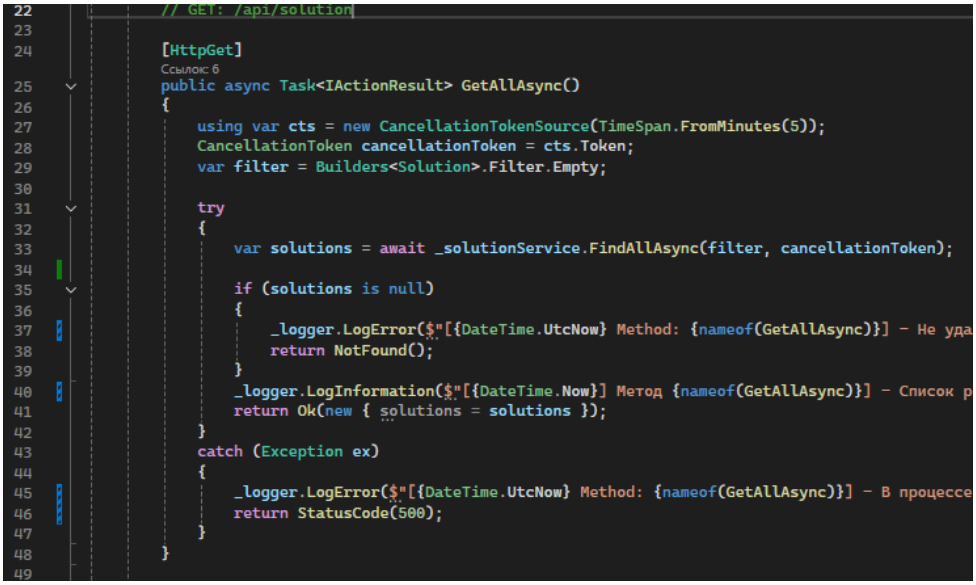


Рисунок 14 – Код метода GetAllAsync для получения списка решений

Для сбора пользовательской информации в клиентском приложении React и API-сервере были разработаны механизм отслеживания активности клиентов. Механизм отслеживания собирает данные о взаимодействиях клиентов с сайтом ИТ-компании и передаёт на API-сервер через HTTP POST-запросы [26]. На серверной стороне данные сохраняются в соответствующие коллекции базы данных MongoDB [6]. Подробная информация о реализации частей механизма отслеживания активности клиентов представлена в таблице 3.

Таблица 3 – Реализация частей механизма отслеживание активности пользователей

Тип взаимодействия клиента с системой	Механизм клиентского приложения	Механизм API-сервера	Конечная точка API-сервера для обработки запросов
Информация о заявках на консультации	Компонент ConsultationCreateForm, представляющий форму для создания заявки на консультацию. Дополнительно используется хук useActionTracking для отслеживания действия создания заявки на консультацию.	Контроллер ConsultationController, сервис ConsultationService, класс Consultation.	/api/consultation
Информация о страницах, которые посетил пользователь на сайте	React-хук usePageTracking, который отслеживает переходы клиента по страницам сайта и отправляет данную информацию на API-сервер.	Контроллер PageController, сервис PageService, класс Page.	/api/page
Информация об отзывах на решения	Компонент с формой создания отзыва ReviewCreateForm.	Контроллер ReviewController, сервис ReviewService, класс Review.	/api/review
Информация о сессиях пользователя	Компонент SessionManager, отвечающий за управление и сбор информации о сессиях клиента.	Контроллер SessionController сервис SessionService, класс Session.	/api/session
Информация о подписках пользователя.	Компонент Payments, представляющий форму для оформления подписки на решение. Дополнительно используется React-хук useActionTracking для отслеживания действия оформления подписки на решение и начала ввода данных для оплаты.	Контроллер UserSubscription Controller, сервис, UserSubscriptionService, класс UserSubscription.	/api/subscription
Информация о действиях пользователя на сайте.	React-хук useActionTracking, который отслеживает действия клиента на сайте и отправляет данную информацию на API-сервер.	Контроллер UserActionController, сервис UserActionService, класс UserAction.	/api/action

Разработанные механизм отслеживания клиентской активности обеспечивают эффективный сбор данных о взаимодействиях клиентов с

сайтом ИТ-компании. Собранные данные являются основой для формирования признаков оттока в классе MLModelInput.

3.2 Формирование признаков для модели машинного обучения и определение критериев оттока

Для разработки модели машинного обучения, прогнозирующей отток клиентов в системе, осуществляется сбор и обработка данных клиентов, включающих поведенческие и атрибутивные характеристики. Эти данные формируют основу для анализа активности клиентов и предсказания вероятности их ухода с сайта ИТ-компании. Для описание клиентских метрик взаимодействия с сайтом был разработан класс MLModelInput, включающий 21 свойство [22]. Для удобства вычислений класс MLModelInput был разделён на 6 подклассов по типу взаимодействия клиента с сайтом.

Название подклассов, их свойства, предназначение свойств, соответствующие свойствам класса MLModelInput представлены в таблице 4.

Таблица 4 – Подклассы класса MLModelInput

Класс системы	Свойство класса MLModelInput
Consultation Interaction	– ConsultationRequestsCount – количество запросов на консультации; – MinutesSinceLastConsultation – количество дней с момента последней консультации.
PageInteraction	– TotalPageViews – общее количество посещений страниц сайта; – UniquePages – количество посещений уникальных страниц сайта; – AvgTimeBetweenViews – среднее время между просмотрами страниц сайта; – TimeSinceLastView – время с последнего просмотра страницы сайта.
ReviewInteraction	– ReviewCount – количество оставленных отзывов; – AverageReviewGrade – средняя оценка отзывов; – TimeSinceLastReview – время с последнего отзыва.
SessionInteraction	– TotalSessions – общее количество сессий; – AvgSessionDuration – средняя длительность сессии; – AvgTimeBetweenSessions – средний интервал между сессиями; – TimeSinceLastSession – время с последнего посещения сайта.

Продолжение таблицы 4

Класс системы	Свойство класса MLModelInput
SubscriptionInteraction	– SubscriptionAttempts – количество попыток оформлений подписки; – ReachedPaymentStage – дошёл ли пользователь до этапа оплаты; – SubscriptionPlan – текущий тариф; – SubscriptionTime – количество месяцев с начала оформления подписки; – HasTrial – использовал ли пробный период.
UserActionInteraction	– AvgTimeBetweenActions – средний интервал между действиями; – TimeSinceLastAction – время с последнего действия; – DetailsClicks – количество просмотров детальной информации о решениях.

Определены ключевые признаки для модели машинного обучения, включающие поведенческие и атрибутивные характеристики, структурированные в классе MLModelInput и разделённые на 6 подклассов. Признаки, представленные в таблице 4, обеспечивают основу для точного анализа активности клиентов и прогнозирования их оттока.

3.3 Разработка сервиса машинного обучения

Для обучения модели машинного обучения в задаче прогнозирования оттока клиентов была сформирована выборка объектов типа MLModelInputs, содержащих детализированные метрики взаимодействий клиентов с системой и целевую переменную IsLikelyToChurn. Формирование выборки включает три основных этапа.

На первом этапе извлекаются данные из коллекции MLModelInputs базы данных MongoDB, содержащие детализированные метрики взаимодействий клиентов с системой. Из-за недостатка реальных данных выборка дополняется синтетическими данными, сгенерированными сервисом SyntheticDataGeneratorService [8]. Синтетические данные генерируются с добавлением 5% шумовых выбросов для имитации аномалий. Сервис создаёт

заданный число записей для 4 типов клиентов системы, представленных в таблице 5:

Таблица 5 – Распределение типов клиентов в синтетически сгенерированных данных

Тип клиента	Процент от сгенерированных данных
Лояльный клиента	20%
Постоянный клиента	40%
Редкий клиента	20%
Клиент с высокой вероятностью оттока	20%

Следующим этапом выборка дынных разделяется на обучающую и тестовую подвыборки в пропорции 80% к 20% с использованием метода TrainTestSplit [11]. Обучающая подвыборка используется для построения pipeline ML-модели и обучения в методе TrainModel, тестовая для оценки метрик AUC-ROC, F1-Score, Accuracy [11]. Процесс разделение выборки представлен на рисунке 15.

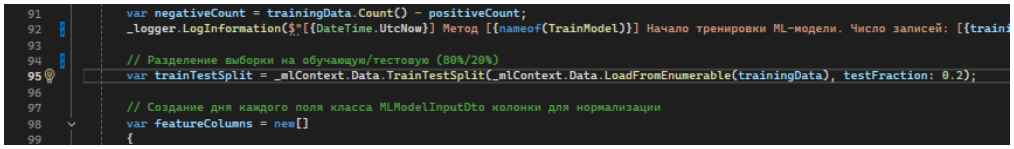


Рисунок 15 – Код метод TrainTestSplit

Заключительным этапом проводится балансировка классов из-за несбалансированности классов в выборке. Записи с ключевым полем IsLikelyToChurn, равные true, составляют 26,36%, а записи с ключевым полем, равные false, – 73,64%. Она необходима для минимизации смещения ML-модели в сторону мажоритарного класса и достижения высоких показателей метрик AUC-ROC, F1-Score. Пропорции классов, которые определялись перед обучением представлена на рисунке 16.

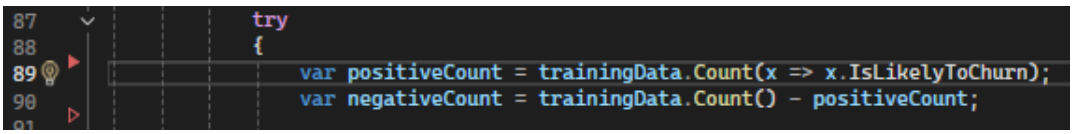


Рисунок 16 – Балансировка классов

Сформированная выборка данных для обучения ML-модели подготовлена с учётом всех необходимых требований. Она включает реальные данные клиентов, а также дополнительно синтетически сгенерированные данные, обеспечивая достаточный объём для процесса обучения. Проведена балансировка классов, что гарантирует объективность модели. Добавление шумов повышает её устойчивость к возможным аномалиям.

Для обработки данных и обучения модели машинного обучения разработан pipeline с использованием библиотеки Microsoft.ML [11]. Pipeline предназначен для подготовки входных данных, обучения ML-модели, оценки её качества и кросс-валидации. Pipeline используется фоновым сервисом TrainingMLModelBackgroundService для обучения модели каждые 4 часа в фоновом режиме. Pipeline в ML.NET состоит из 4 этапов, реализованных в методе TrainModel.

Первым этапом поле IsLikelyToChurn, показывающее склонен ли клиент к оттоку или нет, копируется в колонку Label для использования в качестве целевой переменной бинарной классификации с помощью метода CopyColumns, представленном на рисунке 17 [11].

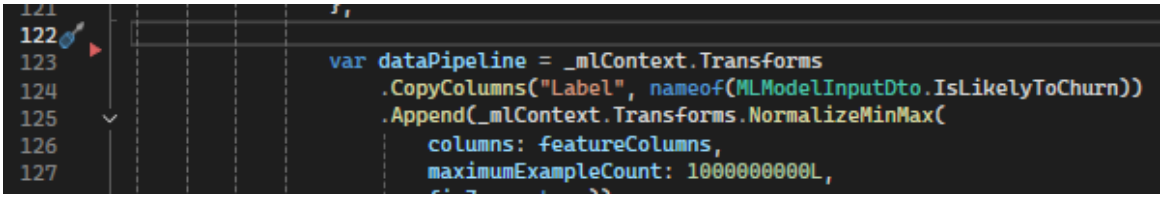


Рисунок 17 – Код метода CopyColumns

Вторым этапом все числовые признаки класса MLModelInput нормализуются с использованием метода NormalizeMinMax, представленном на рисунке 18 [11]. Нормализация применяется для приведения значений к диапазону от 0 до 1. Параметр fixZero обеспечивает корректную обработку нулевых значений, а maximumExampleCount позволяет обрабатывать большие наборы данных.

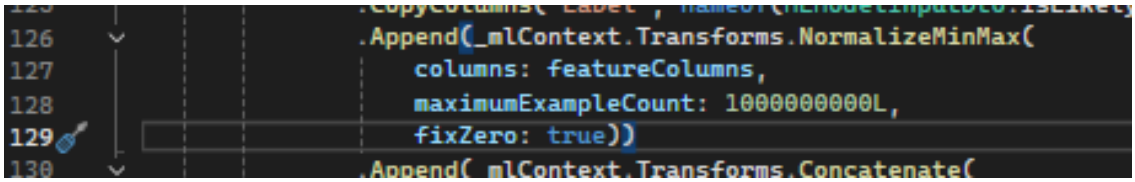


Рисунок 18 – Код метода NormalizeMinMax

Третьим этапом нормализованные признаки объединяются в единый вектор Features с помощью метода Concatenate, представленном на рисунке 19 [11]. Конкатенация обеспечивает создание компактного представления данных, оптимизированного для последующего обучения модели.

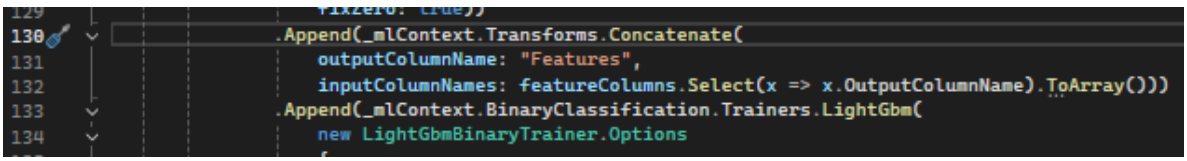


Рисунок 19 – Код метода Concatenate

На заключительном этапе для решения задачи бинарной классификации используется алгоритм LightGbm, реализованный с помощью метода LightGbm [12]. В процессе создания модели задаются параметры NumberOfLeaves – количество листьев в дереве, MinimumExampleCountPerLeaf – минимальное число примеров в листе, LearningRate – скорость обучения, UnbalancedSets – параметр, учитывающий

несбалансированность классов. Код настройки модели представлен на рисунке 20 [11]. После настройки модель обучается на обучающей выборке с помощью метода Fit, а затем оценивается на тестовой выборке с использованием метода Evaluate. Оценка производится по метрикам AUC-ROC, F1-Score и Accuracy. Код обучения и оценки модели представлен на рисунке 21.

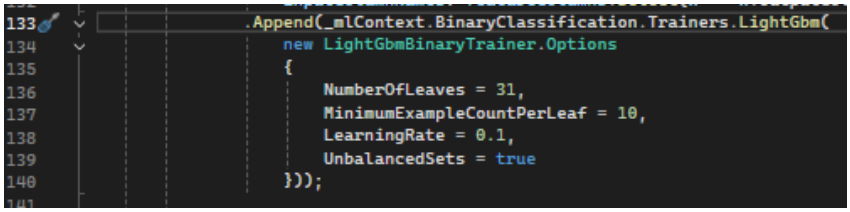


Рисунок 20 – Код метода LightGBM

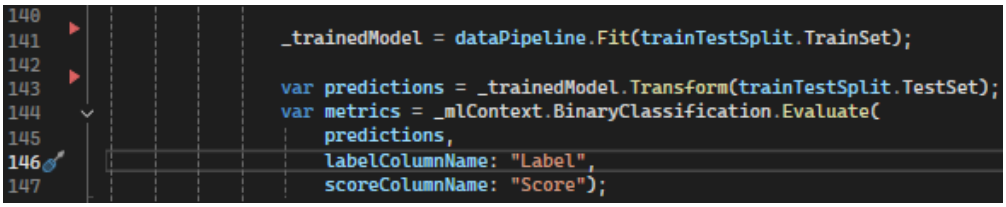


Рисунок 21 – Код методов Fit и Evaluate

Построенный pipeline для обучения модели машинного обучения успешно реализует все необходимые этапы для прогнозирования оттока клиентов. Он включает подготовку данных через копирование целевой переменной, нормализацию и конкатенацию признаков, а также обучение ML-модели с использованием алгоритма LightGBM, адаптированного для работы с несбалансированными наборами данных [12]. Оценка качества ML-модели проводится с применением ключевых метрик, обеспечивая точность её предсказаний.

Для оптимального решения задачи прогнозирования оттока были выбраны и протестированы ML-модели LightGBM, FastForest и GAM из библиотеки Microsoft.ML [11]. Выбор обусловлен их высокой производительностью, устойчивостью к несбалансированным данным и

объяснимостью результатов, что важно для анализа поведения клиентов. Перечень выбранных моделей, а также область их применения представлена в таблице 6.

Таблица 6 – Выбранные модели машинного обучения из библиотеки Microsoft.ML и их области применения

ML-Модель	Класс Microsoft.ML	Область применения
LightGBM	LightGbmBinaryTrainer	Задачи классификации с акцентом на достижение высокой точности на несбалансированных наборах данных. Быстрая обучаемость и поддержка нелинейных зависимостей.
FastForest	FastForestBinaryTrainer	Задачи классификации, где требуется стабильность и устойчивость к шумам и выбросам в данных. Работа с неупорядоченными или содержащими аномалии наборами данных.
GAM	GamBinaryTrainer	Задачи классификации, требующие высокой объяснимости результатов и выявления нелинейных закономерностей и, где важно интерпретировать влияние отдельных факторов на исход.

Для обучения и оценки качества ML-модели был разработан метод TrainAndCompare. Работа TrainAndCompare повторяет логику работы метода TrainModel, добавляя возможность обучения и оценки метрик 3 моделей. Результаты оценки качества ML-моделей представлены в таблице 7 [11].

Таблица 7 – Результаты оценки метрик ML-моделей

ML-Модель	Баланс классов	AUC-ROC	Accurancy	F1-Score	Средний AUC-ROC
LightGBM	26,36%	98,33%	93,28%	86,71%	98,13%
FastForest	26,36%	98,00%	93,48%	86,01%	98,12
GAM	26,36%	97,45%	92,99%	84,62%	97,85

На рисунке 22 представлены логи с метриками оценки ML-моделей, включая результаты кросс-валидации.

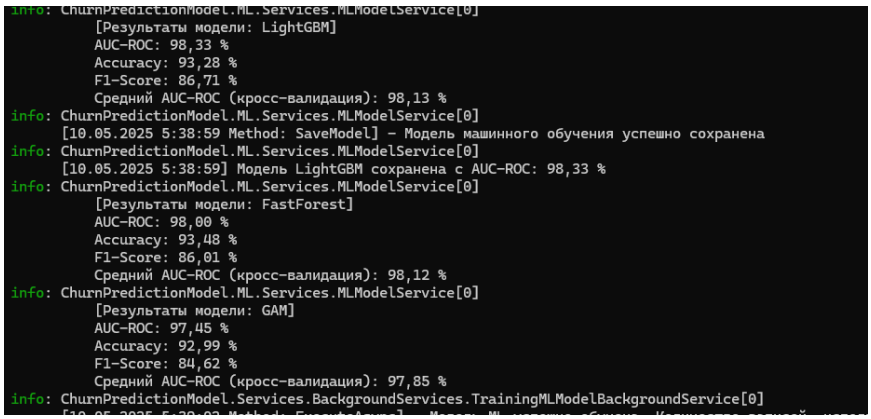


Рисунок 22 – Логи с информацией о метриках ML-моделей LightGBM, FastForest, GAM

Объяснение используемых метрик для оценки качества ML-моделей:

– AUC-ROC измеряет площадь под кривой ROC, которая отражает зависимость между долей истинно положительных предсказаний и долей ложно положительных предсказаний при различных порогах классификации. Значение AUC-ROC варьируется от 0 до 1. Где 1.0 – это идеальная модель, безошибочно различающая классы, 0.5 – ML-модель не лучше случайного угадывания, 0,9 – высокая производительность;

– F1-Score представляет гармоническое среднее значение между числом клиентов, верно классифицированных как склонные к оттоку, среди всех отнесённых к оттоку и числом клиентов, склонных к оттоку, которых ML-модель верно выявила из всех действительно склонных к оттоку.

– Accuracy представляет собой долю правильных предсказаний модели машинного обучения от общего числа предсказаний. Метрика показывает, насколько часто ML-модель верно классифицирует клиентов.

На основе анализа метрик моделей машинного обучения ML-модель LightGBM показала наилучшие результаты метрики AUC-ROC – 98, 33%, что

делает её предпочтительной для прогнозирования оттока [12]. ML-модель LightGBM выбрана как основная модель благодаря высокой точности, поддержке несбалансированных данных и эффективности на больших выборках данных, что соответствует требованиям разрабатываемой системы. AUC-ROC является ключевой метрикой, так как она наиболее точно отражает способность ML-модели различать классы в условиях несбалансированного распределения данных – 26,36%.

Для автоматической обработки клиентской информации в фоновом режиме был разработан фоновый сервис DataProcessingBackgroundService. DataProcessingBackgroundService выполняется раз в 1 час и обеспечивает подготовку данных для тренировки ML-модели [20]. Для получения клиентских данных отправляется HTTP GET-запрос к API-серверу. Полученные данные обрабатываются, записываются в объекты типа MLModelInput с использованием соответствующих методов и сохраняются в базе данных [25]. Обработка клиентских данных осуществляется с помощью следующих 6 методов:

- Метод CalculateConsultationInteraction – вычисляет все поля класса ConsultationInteraction;
- Метод CalculateUserPageInteraction –вычисляет все поля класса PageInteraction;
- Метод CalculateReviewInteraction – вычисляет все поля класса ReviewInteraction;
- Метод CalculateUserSessionInteraction – вычисляет все поля класса SessionInteraction;
- Метод CalculateSubscriptionInteraction – вычисляет все поля класса SubscriptionInteraction;
- Метод CalculateUserAction – вычисляет все поля класса UserAction.

Для поддержания актуальности и точности предсказаний оттока клиентов был разработан фоновый сервис TrainingMLModelBackgroundService, обеспечивающий периодическую тренировку ML-модели раз в 4 часа на

основе актуальных клиентских данных, дополненных синтетически сгенерированными данными [27]. Алгоритм работы сервиса:

- Осуществляется загрузка детализированных метрик взаимодействия клиентов с системой из базы данных MongoDB с помощью метода GetMLModelInputData [6]. Полученные данные проверяются на наличие. Если список пуст, процесс обучения не запускается;
- Каждый загруженный объект типа MLModelInput преобразуется в класс MLModelInputDto для совместимости с pipeline ML-модели;
- Вызывается метод Generate для генерации 10000 синтетических записей. При генерации используется фиксированный сид – 999, обеспечивающий воспроизводимость результатов [8]. Синтетически сгенерированные данные используются в дополнение к реальным данным клиентов для тренировки модели, что позволяет увеличить объём выборки и смоделировать дополнительно разнообразные сценарии поведения пользователей;
- Вызывается метод TrainModel для тренировки модели машинного обучения, передавая в качестве параметра детализированные метрики взаимодействия клиентов с системой [11].

Для обеспечения актуальности предсказаний оттока клиентов разработан фоновый сервис CalculatingPredictionBackgroundService, который раз в 30 минут вычисляет актуальную вероятность оттока на основе данных, подготовленных сервисом DataProcessingBackgroundService, и сохраняет результаты в базе данных [20]. Алгоритм работы сервиса:

- Сервис вызывает метод GetMLModelInputData, который выполняет загрузку всех объектов типа MLModelInput из базы данных MongoDB. Полученные данные проверяются на наличие. Если список пуст, процесс предсказания не запускается;
- Для каждого объекта MLModelInput создаётся объект типа MLModelInputDto. Выполняется предсказание с помощью метода

Predict. Если предсказание успешно – Prediction не равно null, оно добавляется в список churnPredictionsList [11];

– Список churnPredictionsList с объектами типа Prediction сохраняется в MongoDB с помощью метода SaveChurnPredictionsAsync [6].

Разработанные фоновые сервисы обеспечивают автоматизацию обработки клиентских данных, периодическую. тренировку модели машинного обучения на актуальных данных, а также получения актуальных предсказаний оттока клиентов. Данные сервисы обеспечивают эффективность и надёжность процесса прогнозирования оттока, соответствуя функциональным требованиям разрабатываемой системы.

3.4 Интеграция сервиса машинного обучения

Для обеспечения взаимодействия сервиса машинного обучения с сайтом ИТ-компании разработан контроллер DataController, реализованный на API-сервере. Контроллер размещён по маршруту /api/data и предназначен для передачи данных клиентов из базы данных MongoDB в сервис машинного обучения. DataController включает конечную точку, обрабатывающую HTTP GET-запросы для получения данных клиентов [20]. При обращении к ней контроллер взаимодействует с сервисами UserService, ConsultationService, PageService, ReviewService, SessionService, UserSubscriptionService, UserActionService для загрузки соответствующих данных из базы данных MongoDB [27]. В случае, если метрики взаимодействия клиента равны null, возвращается HTTP-статус 404. Для каждого клиента формируется объект UserData, содержащий ранее полученные данные. Список объектов UserData возвращается в качестве ответа сервису машинного обучения в формате JSON через HTTP-статус 200 [15]. Код конечной точки представлен на рисунке 23.

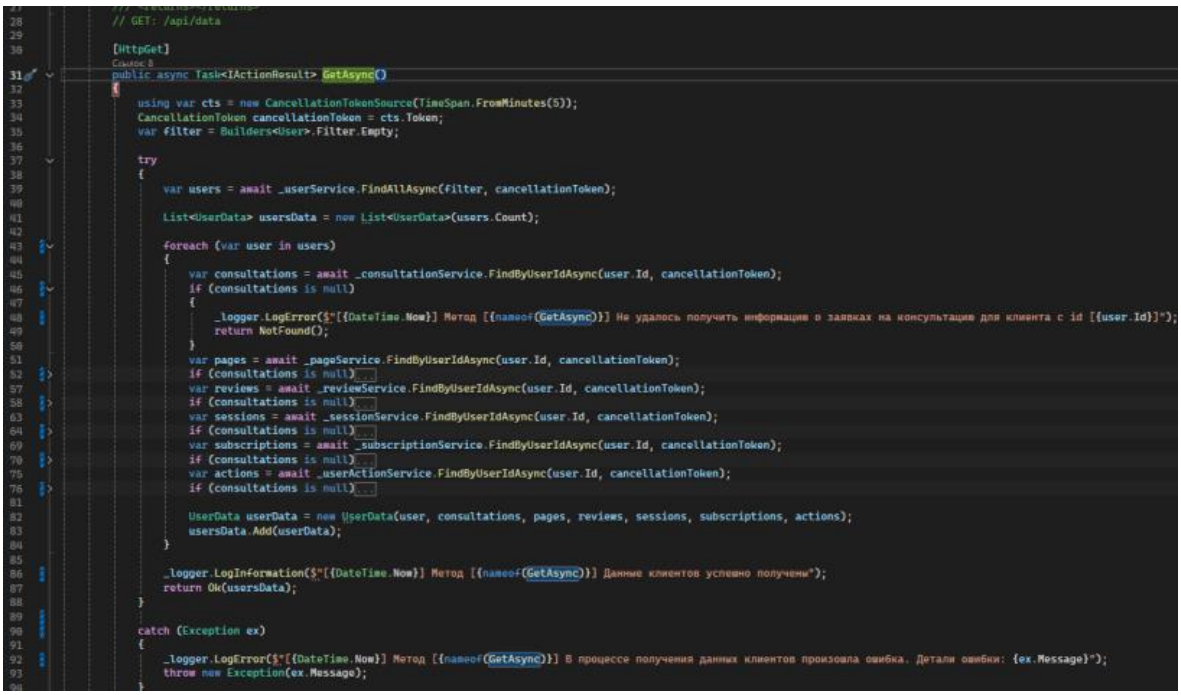


Рисунок 23 – Код конечной точки контроллера DataController для получения данных клиентов

Разработанный контроллер DataController с конечной точкой на API-сервере обеспечивают передачу данных клиентов из базы данных в сервис машинного обучения.

3.5 Разработка панели администратора и системы уведомлений

Для анализа клиентов с высокой вероятностью оттока и отправки им персонализированных уведомлений были разработаны компоненты клиентского приложения, реализующие отправку HTTP-запросов, а также соответствующая логика на стороне API-сервера.

На клиентской стороне был разработан компонент ChurnPredictions, представляющий собой панель администратора. Компонент предоставляет возможность просмотра предсказаний оттока, анализа детализированных метрик взаимодействия клиентов с системой и отправки им персонализированных уведомлений.

Компонент ChurnPredictions реализует следующий функционал:

– При монтировании компонента выполняется отправка HTTP GET-запроса к API-серверу для загрузки списка предсказаний [7]. Полученный в ответе список визуализируется на компоненте. Предсказание представляют собой имя и фамилию клиента, информацию, склонен ли к оттоку и процентную вероятность оттока [9]. Компонент администратора представлен на рисунке 24;
– Для получения детализированных метрик взаимодействия клиента с системой на компоненте реализована кнопка «Получить подробную информацию», при нажатии на которую отправляется HTTP GET-запрос на сервер для получения данных. Компонент администратора с отображением детализированных метрик взаимодействия клиента с системой представлен на рисунке 25;
– С помощью модального окна, представленного на рисунке 26, открывающееся при нажатии на кнопку «Отправить уведомления», определена возможность отправки персональных уведомлений клиентам на основе их метрик взаимодействия с системой. Поддерживаются четыре типа уведомлений: напоминание о незавершённой подписке, приглашение на консультацию с экспертом, поощрение за оставление отзыва, напоминание о длительном бездействии. Выбранные уведомления отправляются через HTTP POST-запрос с указанием идентификатора клиента [7].

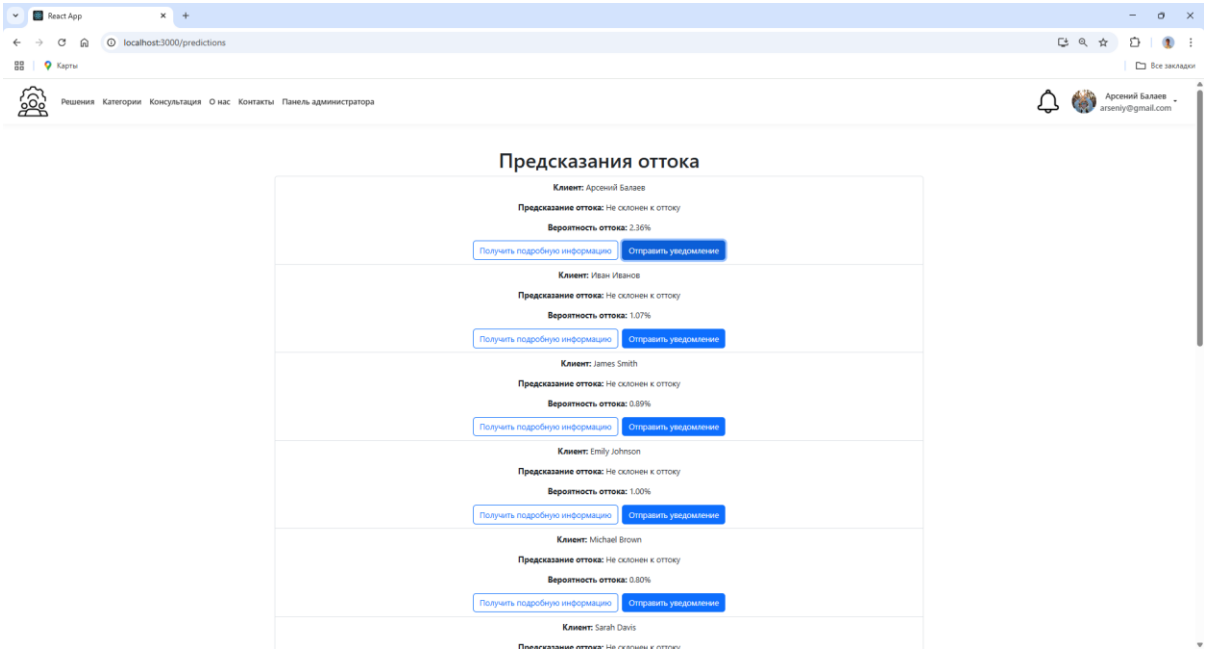


Рисунок 24 – Компонент панели администратора

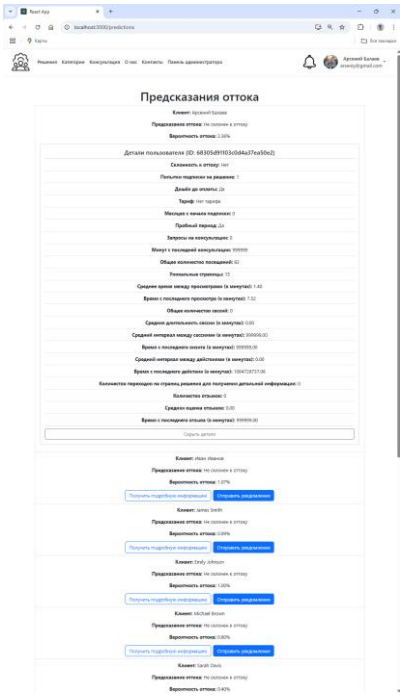


Рисунок 25 – Компонент панели администратора с отображением детализированных метрик взаимодействия клиента с системой

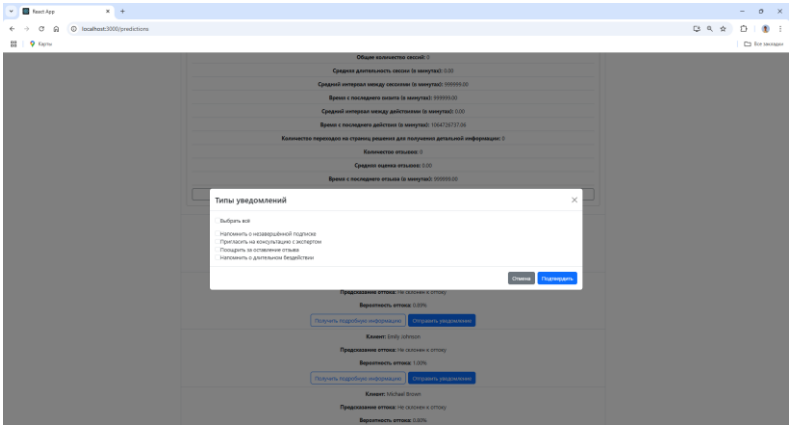


Рисунок 26 – Интерфейс модального окна для отправки персонализированных уведомлений

Также в клиентском приложении, в дополнение к компоненту ChurnPredictions, был разработан компонент Notifications, обеспечивающий отображение отправленных уведомлений. Компонент Notifications отвечает за загрузку и отображение списка уведомлений для клиентов. При монтировании компонента отправляется HTTP GET-запрос к API-серверу для получения списка уведомлений. Полученный список визуализируется в компоненте [9]. Компонент Notifications представлен на рисунке 27.

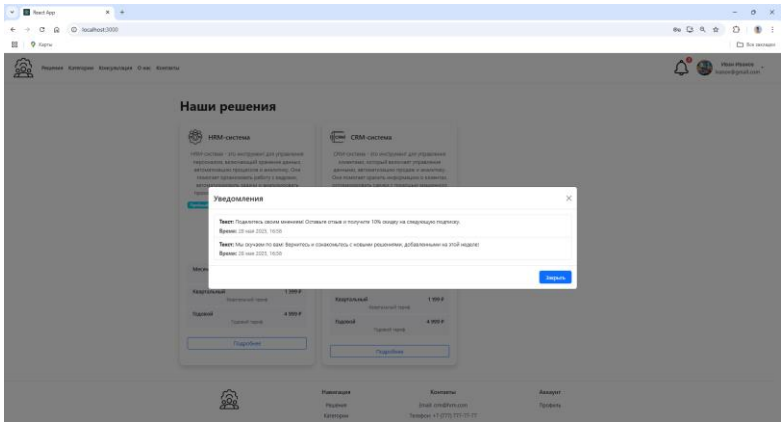


Рисунок 27 – Компонент Notifications

Для обеспечения полноценной работы клиентских компонентов ChurnPredictions и Notifications на стороне API-сервера реализованы

контроллеры ChurnPredictionController и NotificationController. Данные контроллеры обрабатывают запросы, связанные с получением предсказаний оттока, детализированных метрик взаимодействия клиентов с системой, а также с созданием и формированием персонализированных уведомлений [20].

Для обеспечения доступа к данным, обрабатываемым контроллерами ChurnPredictionController и NotificationController, исключительно пользователям с правами администратора, на сервере реализован механизм авторизации на основе JWT-токена. Доступ к защищённым ресурсам предоставляется только пользователям с ролью Admin. При этом для доступа к конечным точкам NotificationController, связанным с получением уведомлений, доступ разрешён также авторизованным клиентам [1].

Контроллер ChurnPredictionController отвечает за предоставление информации о вероятности оттока клиентов. Разработано 2 конечные точки для обработки запросов:

- Конечная точка GET /api/prediction предоставляет список объектов PredictionModel, содержащих данные клиентов и предсказания оттока;
- Конечная точка GET /api/prediction/details/{userId} предоставляет детализированные метрики взаимодействия клиента с системой на основе указанного идентификатора пользователя.

Контроллер NotificationController предназначен для создания и управления уведомлениями для клиентов. Разработано 3 конечные точки для обработки запросов:

- Конечная точка GET /api/notification/{userId} загружает список уведомлений на основе указанного идентификатора пользователя;
- Конечная точка GET /api/notification/count/{userId} загружает количество уведомлений на основе указанного идентификатора пользователя;
- Конечная точка POST /api/notification принимает объект типа RetentionAction, содержащий идентификатор клиента и список

действий и вызывает методы сервиса NotificationService для создания уведомлений [15].

Для создания персонализированных уведомлений на основе детализированных метрик взаимодействия клиента с системой разработан сервис NotificationService, вызываемый из контроллера NotificationController. С клиентского приложения поступает список типов уведомлений, рекомендуемых к отправке, после чего NotificationService извлекает из базы данных объект MLModelInput, анализирует метрики клиента и создаёт только корректные уведомления [27]. Также сервис выполняет CRUD-операции с уведомлениями. NotificationService предоставляет следующие методы для создания уведомлений:

- Метод SendIncompleteSubscriptionReminder создаёт уведомление для пользователей с незавершёнными попытками подписки на решения;
- Метод SendConsultationInvite анализирует наличие актуальных запросов на консультацию и создаёт уведомление, если с момента последнего запроса прошло менее 24 часов;
- Метод SendReviewEncouragement создаёт уведомление для клиентов, не оставлявших отзыв на решение, или если с последнего отзыва прошло более 3 дней;
- Метод SendInactivityReminder создаёт уведомления для клиентов, чья последняя сессия на сайте завершилась была более 7 дней назад;

Таким образом, клиентская сторона обеспечивает доступ к информации, позволяя просматривать прогнозы оттока клиентов и детализированные метрики взаимодействия клиентов с системой, а также отправлять персонализированные уведомления, с целью их удержания. Серверная часть включает контроллеры, обрабатывающие запросы для получения предсказаний оттока, детализированных метрик взаимодействия клиентов с системой, а также создания и получения уведомлениями. Для защиты ресурсов реализован механизм авторизации на основе JWT-токена, предоставляющий доступ к защищённым конечным точкам только пользователям с ролью

администратора, за исключением отдельных методов контроллера NotificationController, доступных также авторизованным клиентам.

3.6 Тестирование системы

Для проверки корректности работы ключевых компонентов системы, включая пользовательский интерфейс, API, фоновые сервисы было проведено функциональное тестирование. Тестирование охватило основные функции, связанные с регистрацией и аутентификацией, отображением списка предсказаний, отправкой запросов на консультацию, отслеживанием действий клиентов, работой фоновых сервисов, отображением предсказаний и детализированных метрик взаимодействия клиентов с системой, а также отправкой уведомлений. Ниже представлено описание каждого теста.

Тестирование регистрации и аутентификации пользователя. Процесс:

- Открыт компонент регистрации RegisterForm, где введены валидные данные пользователя. Форма регистрации с введёнными валидными данными представлена на рисунке 28;
- Нажата кнопка «Зарегистрироваться», отправляющая запрос к серверу для создания записи и получения JWT-токена [5]. Успешно создана запись в MongoDB содержащая данные клиента. Созданная запись представлена на рисунке 29;
- Выполнен выход из аккаунта на сайте и открыт компонент аутентификации AuthenticateForm. Введены данные зарегистрированного пользователя и нажата кнопка «Войти», отправляющая запрос к серверу для входа в аккаунт;

Результат: валидные данные клиента успешно отправлены и сохранены в базе данных. Вход в аккаунт выполнен успешно;

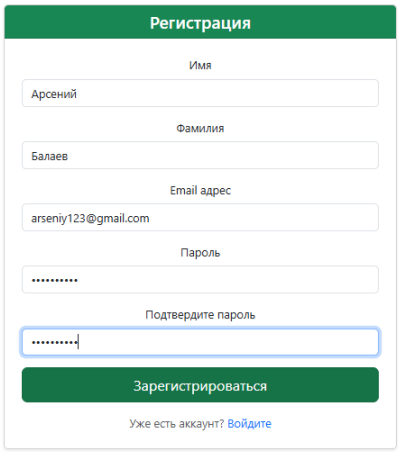


Рисунок 28 – Страница регистрации с введёнными валидными данными пользователя

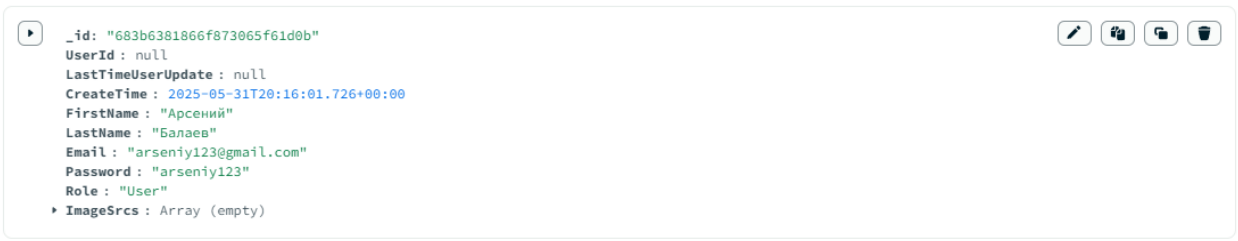


Рисунок 29 – Созданная запись в базе данных MongoDB

Тестирование отображения списка решений на странице сайта ИТ-компании. Процесс:

– Созданы 2 записи в базе дынных, содержащие информацию о решениях.
– Открыта страница сайта, содержащая компонент Solutions, который отправляет запрос к серверу при монтировании, получает и отображает список решений. Интерфейс сайта со списком решений показан на рисунке 30.

Результат: список решений корректно получен из базы данных и отображён на странице сайта.

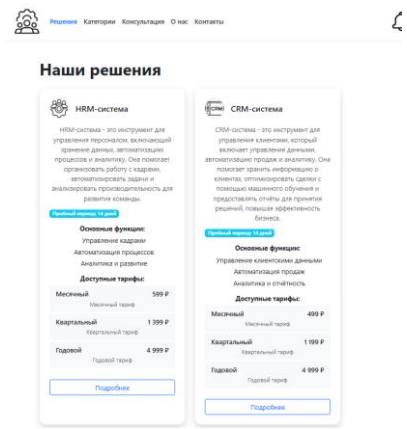


Рисунок 30 – Интерфейс страницы сайта со списком решений

Тест создания заявки на консультацию. Процесс:

– Открыта страница с формой для создания заявки на консультацию. Введены валидные данные. Страница с заполненными валидными данными представлена на рисунке 31;
– Нажата кнопка «Отправить», после которой была выполнена отправка запроса к серверу и сохранение в базе данных. Запись созданной заявки в базе данных MongoDB представлена на рисунке 32.

Результат: заявка на консультацию успешно создана и записана в базу данных.

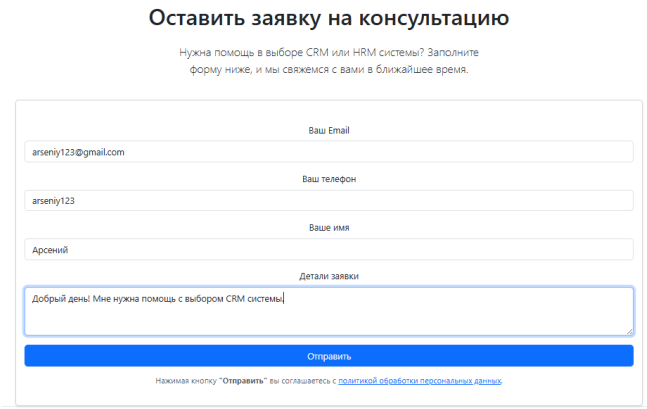


Рисунок 31 – Страница создания заявки на консультацию с заполненными валидными данными

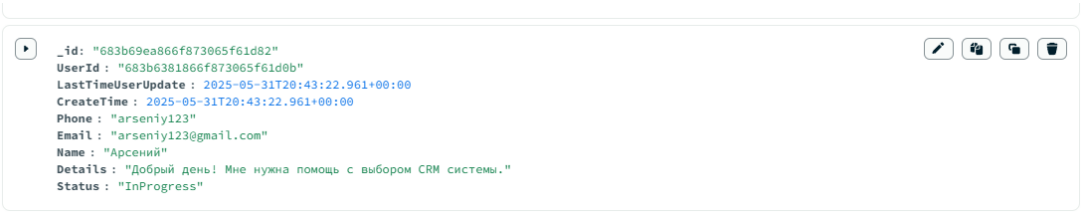


Рисунок 32 – Запись созданной заявки в базе данных MongoDB

Тест создания действия клиента при создании запроса на консультацию

Процесс:

– В процессе отправки заявки на консультацию React-хук useActionTracking зафиксировал действие создания заявки клиентом и отправил данные серверу [16];
– Данные действия клиента успешно сохранены в базе данных MongoDB. Запись действия клиента в базе данных MongoDB представлена на рисунке 33.

Результат: действие клиента при создании заявки корректно зафиксировано и сохранено в базе данных

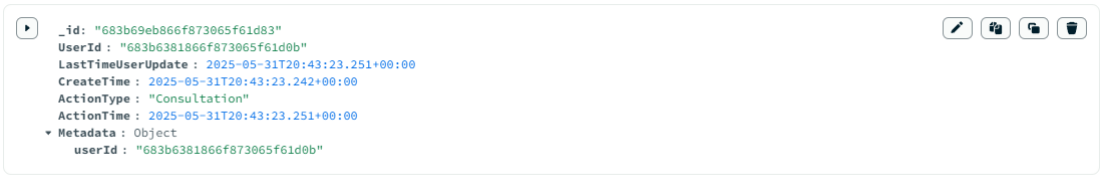


Рисунок 33 – Запись действия пользователя в базе данных MongoDB

Тестирование фонового сервиса DataProcessingBackgroundService с целью проверки корректности обработки клиентских данных и записи детализированных метрик взаимодействия клиентов с системой базу данных MongoDB.

Процесс тестирования:

– Создано 12 клиентов и выполнена имитация использования сайта для сбора необходимых данных;
– Фоновый сервис DataProcessingBackgroundService запросил клиентские данных с API-сервера и успешно получил их;
– Сервис обработал данные клиентов, выполнил обработку и записал детализированные метрики взаимодействия клиентов с системой в базу данных MongoDB.

Результат: все данные успешно обработаны и сохранены.

На рисунках 34, 35 представлены логи фонового сервиса DataProcessingBackgroundService, подтверждающие успешное получение данных с API-сервиса, выполнение обработки и сохранение, а также коллекция MLModelInputs, содержащая 12 записей метрик клиентов.

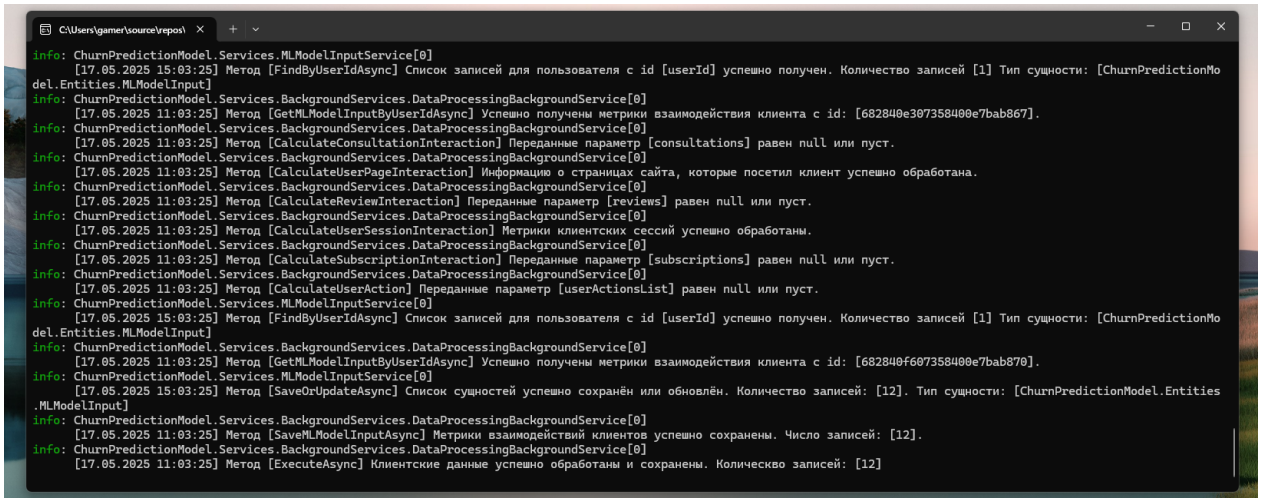


Рисунок 34 – Логи фонового сервиса DataProcessingBackgroundService

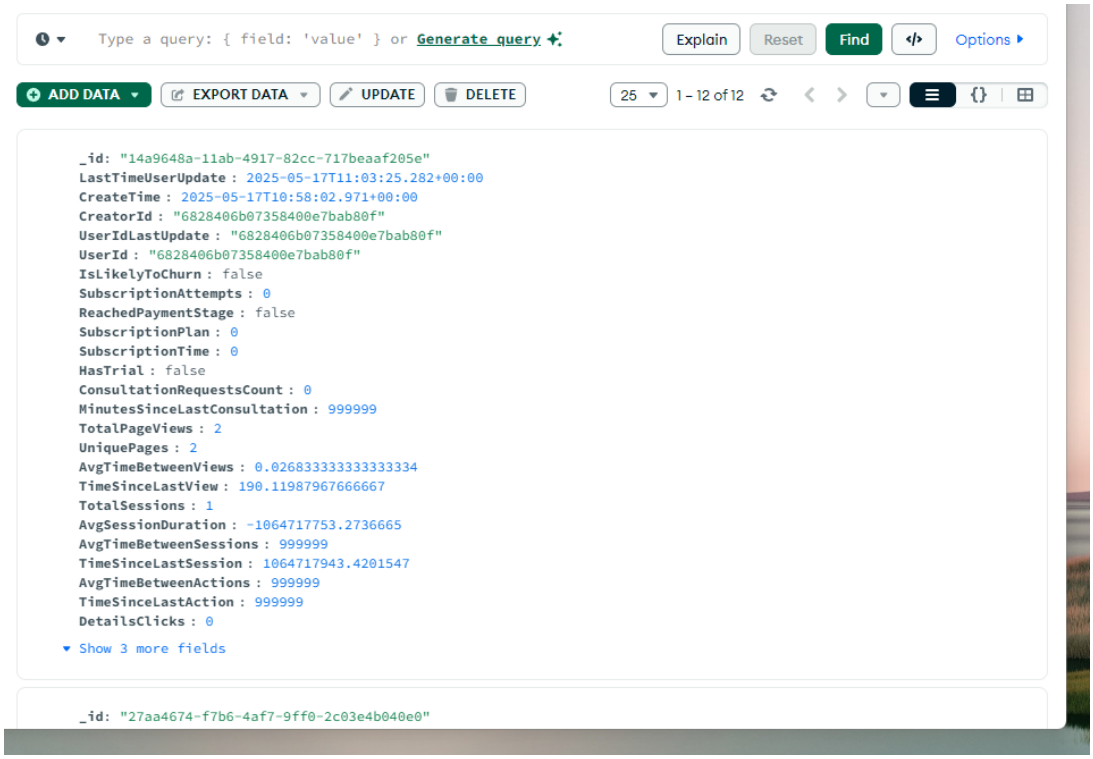


Рисунок 35 – Список созданных записей в коллекция MLModelInputs

Тестирование фонового сервиса TrainingMLModelBackgroundService с целью проверки корректности обучения ML-модели на основе метрик клиентов. Процесс:

– Для обучения из базы данных получено 12 ранее созданных записей метрик клиентов;
– Дополнительно сгенерировано 10000 синтетических записей типа MLModelInput [8];
– 10012 записей были переданы в ML-модель;
– Модель машинного обучения обучена и сохранена [11].

Результат: ML-модель успешно обучена, сохранена и готова к использованию.

На рисунке 36 представлены логи сервиса TrainingMLModelBackgroundService, подтверждающие успешную загрузку, получения данных, тренировку и сохранение модели машинного обучения.

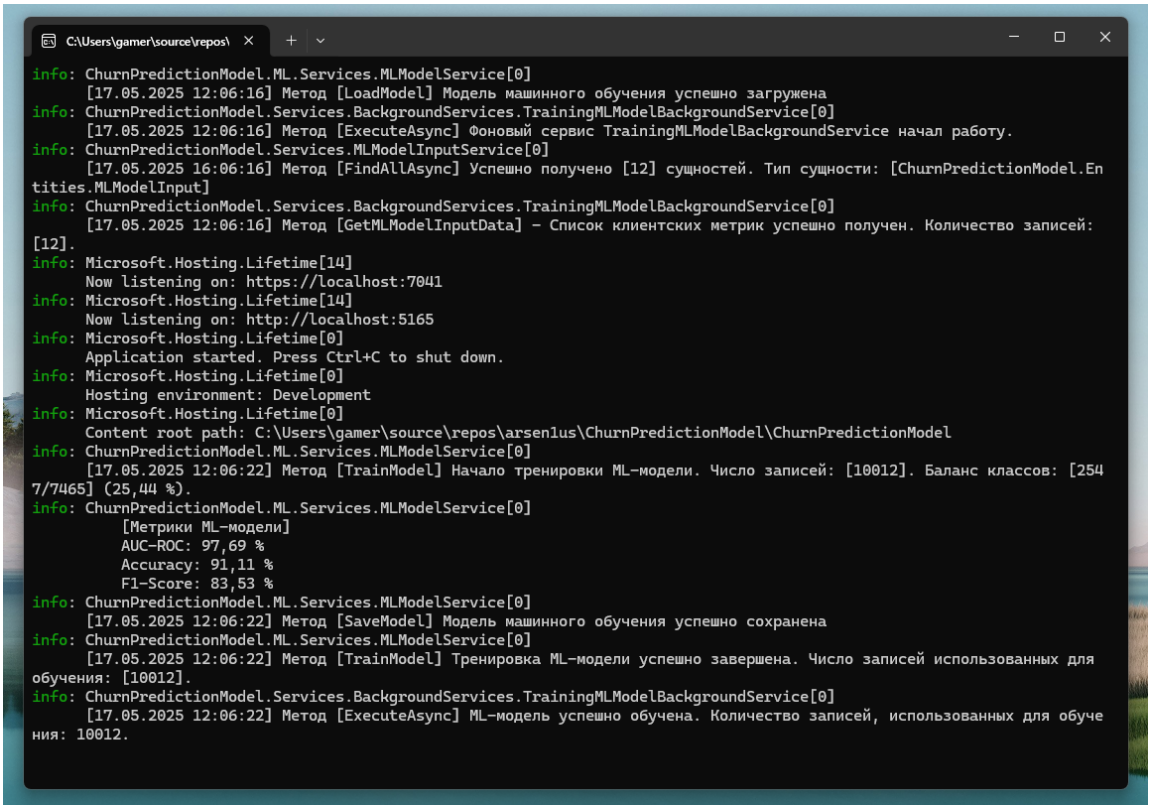


Рисунок 36 – Логи фонового сервиса TrainingMLModelBackgroundService

Тестирование фонового сервиса CalculatingPredictionBackgroundService с целью проверки корректности расчёта и сохранения вероятности оттока клиентов. Процесс:

– Для получения предсказаний получено 12 ранее созданных записей метрик клиентов;
– Сервис CalculatingPredictionBackgroundService рассчитал предсказания для 12 клиентов и сохранил объекты типа Prediction в базу данных MongoDB [6].

Результат: предсказания успешно рассчитаны и сохранены.

На рисунках 37, 38 представлены логи сервиса CalculatingPredictionBackgroundService, подтверждающий успешное получение данных, расчёт и сохранение предсказаний, а также коллекцию Predictions в базе данных MongoDB, содержащую 12 записей.

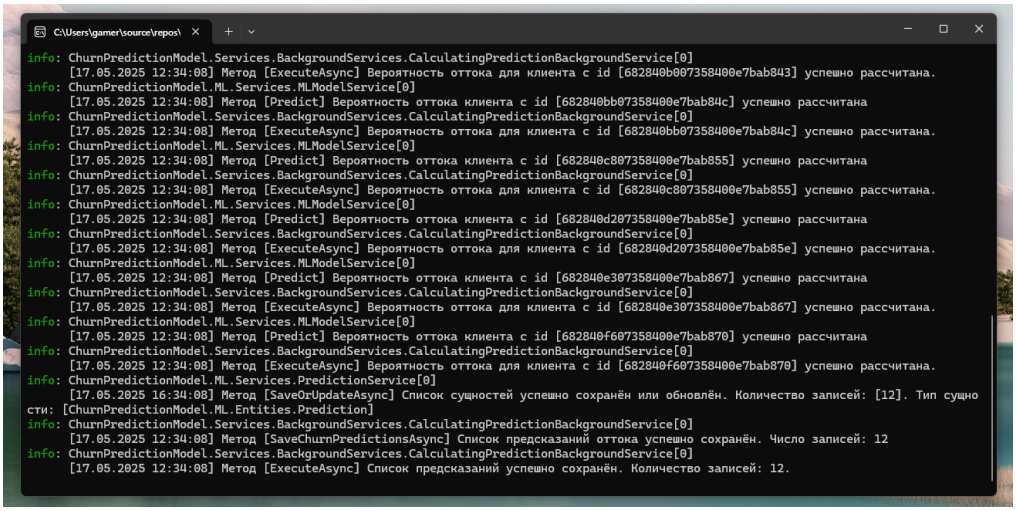


Рисунок 37 – Логи фонового сервиса CalculatingPredictionBackgroudService

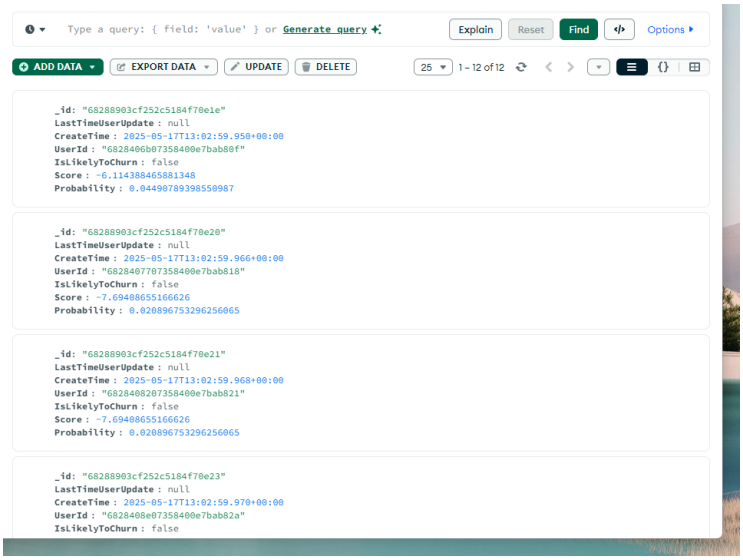


Рисунок 38 – Коллекция Predictions MongoDB

Тестирование визуализации предсказаний оттока клиентов на панели администратора. Процесс:

– Компонент ChurnPredictions отправил запрос к API-серверу для загрузки 12 ранее полученных предсказаний;
– Успешно получены данные от API-сервера и визуализированы на интерфейсе.

Результат: список из 12 предсказаний успешно визуализирован на интерфейсе.

На рисунке 39 представлен интерфейса компонента ChurnPredictions с корректно визуализированным списком из 12 предсказаний оттока клиентов.

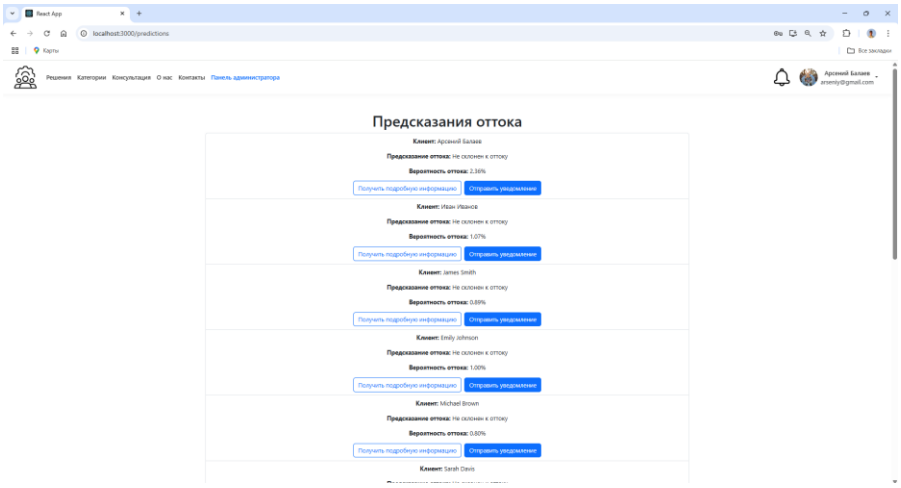


Рисунок 39 – Тестирование интерфейса страницы со списком предсказаний

Тестирование визуализации детализированных метрик взаимодействия клиента с системой. Процесс:

– В компоненте ChurnPredictions нажата кнопка «Получить подробную информацию» для выполнения запроса к серверу для получения детализированных метрик взаимодействия клиента с системой;
– Успешно получены данные визуализированы на интерфейсе [9].

Результат: детализированные метрики взаимодействия клиента с системой корректно визуализированы на интерфейсе.

На рисунках 40, 41 представлен интерфейса компонента ChurnPredictions с корректно визуализированными детализированными метриками взаимодействия клиента с системой, а также соответствующая запись в базе данных MongoDB.

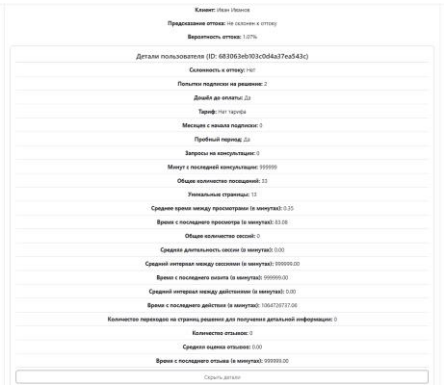


Рисунок 40 –Детализированные метрики взаимодействия клиента с системой

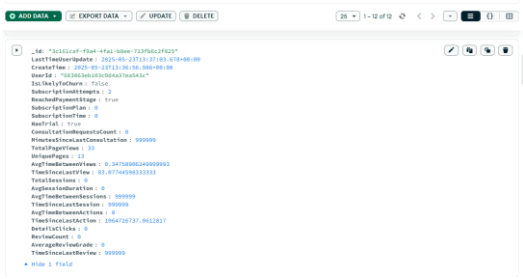


Рисунок 41 – Запись детализированных метрик взаимодействия клиента с системой в базе дынных MongoDB

Тестирование корректности отправки и получения персональных уведомлений на основе клиентских метрик. Процесс:

- Выполнен вход в аккаунт администратора, из списка пользователей выбран 1 клиент;
- Выбрано 4 возможных типа уведомления и отправлены клиенту;
- Выполнен вход в аккаунт клиента для проверки.
- Из 4 уведомлений 2 успешно доставлены и отображены в списке уведомлений клиента. На API-сервере с помощью сервиса Notification успешно произошёл анализ детализированных метрик взаимодействия клиента с системой и сформированы корректные уведомления [27].

Результат: уведомления отправлены, сформированы и получены корректно, сервис NotificationService работает как ожидается.

На рисунках 42, 43 представлен интерфейса панели администратора с модальным окном отправки уведомлений, а также интерфейс страницы клиента с отображением 2 полученных уведомлений.

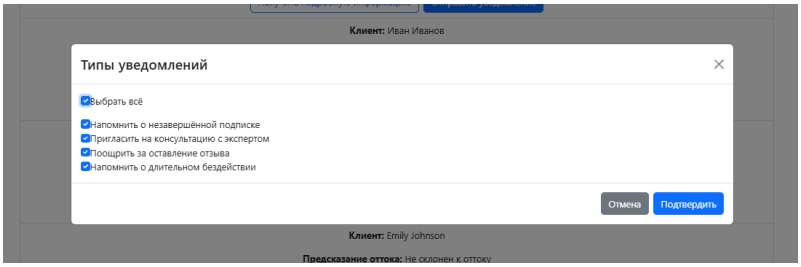


Рисунок 42 – Отправка уведомлений клиенту с панели администратора

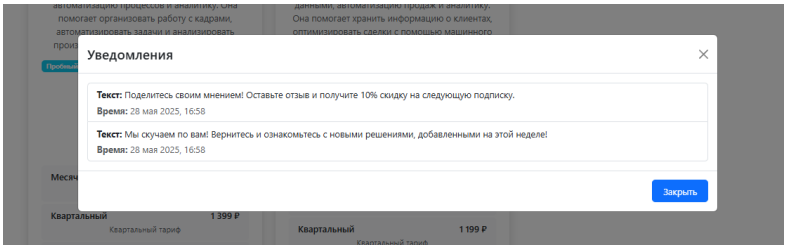


Рисунок 43 – Список уведомления, которые пришли клиенту

Функциональное тестирование подтвердило, что все основные компоненты системы функционируют в соответствии с заданными требованиями.

Было проведено интеграционное тестирование для проверки взаимодействия компонентов системы, обеспечивающих обучение ML-модели и предсказание вероятности оттока клиентов [3]. Тестирование охватило ключевые методы сервиса MLModelService, включая обработку валидных и некорректных входных данных, а также взаимодействие с базой данных MongoDB и файловой системой [13]. Использовались библиотеки FluentAssertions, Xunit, Moq, Microsoft.AspNetCore.Mvc.Testing, Mongo2Go,

MongoDB.Driver, а также фабрика CustomWebApplicationFactory для создания тестового окружения [23]. Ниже приведено описание проведённых тестов.

Тестирование метода TrainModel с валидными данными с целью проверки успешного обучения модели, её сохранения по указанному пути и использования для предсказания оттока с вероятностью в диапазоне от 0 до 1. Процесс:

- Сгенерировано 1000 записей, содержащие детализированные метрики взаимодействия клиентов с системой, с помощью сервиса SyntheticDataGeneratorService [14];
- Обучена ML-модели с помощью метода TrainModel, сохранена в файл;
- Загружена ранее обученная ML-модель и сгенерирована 1 синтетическая запись типа MLModelInputDto, для получения предсказания [14];
- Успешно получено предсказание;
- Прошла проверка наличия файла модели, успешной загрузки и корректности предсказания [10].

Результат: модель успешно обучена, сохранена и применена для получения предсказаний оттока.

Тестирование метода TrainModel с передачей null в качестве параметра метрик клиентов, с целью проверки обработки исключения ArgumentException. Процесс:

- Передано значение null в качестве параметра метрик клиентов в метод TrainModel;
- Ожидание исключения ArgumentException с сообщением, указывающим на null или пустой параметр [22].

Результат: метод корректно выбросил исключение с ожидаемым сообщением.

Тестирование метода TrainModel с передачей пустого списка метрик клиентов, с целью проверки обработки исключения ArgumentException. Процесс:

– Передан пустой список метрик клиентов в метод TrainModel;
– Ожидание исключения ArgumentException с сообщением, указывающим на null или пустой параметр.

Результат: метод корректно выбросил исключение с ожидаемым сообщением.

Тестирование метода Predict с валидными данными. Процесс:

– Сгенерировано 1000 записей, содержащие метрики клиентов, с помощью сервиса SyntheticDataGeneratorService;
– Обучена ML-модели с помощью метода TrainModel и сохранена в файл;
– Загружена ранее обученная ML-модель;
– Сгенерирована одна тестовая запись и выполнен метод Predict с уникальным идентификатором пользователя;
– Проверка соответствия идентификатора клиента, ненулевой результат и диапазон вероятности от 0 до 1 [10].

Результат: предсказание возвращено корректно.

Тестирование метода Predict с передачей в качестве параметра пустого идентификатора пользователя, с целью проверки обработки исключения ArgumentNullException. Процесс:

– Сгенерировано 1000 записей, содержащие метрики клиентов, с помощью сервиса SyntheticDataGeneratorService;
– Обучена ML-модели с помощью метода TrainModel и сохранена в файл;
– Загружена ранее обученная ML-модель;
– Сгенерирована одна тестовая запись и выполнен метод Predict с передачей пустого идентификатора клиента [14];
– Ожидание исключение ArgumentNullException с сообщением о некорректной передаче параметров [22].

Результат: метод корректно выбросил исключение.

Тестирование метода Predict с null с передачей в качестве параметра метрик клиентов, которые равны null, с целью проверки обработки исключения ArgumentNullException. Процесс:

– Сгенерировано 1000 записей, содержащие метрики клиентов, с помощью сервиса SyntheticDataGeneratorService;
– Обучена ML-модели с помощью метода TrainModel, сохранена в файл;
– Загружена ранее обученная ML-модель;
– Выполнен метод Predict с передачей null, в качестве метрик клиентов и созданного идентификатора пользователя;
– Ожидание исключение ArgumentNullException с сообщением о некорректной передаче параметров.

Результат: метод корректно выбросил исключение.

На рисунке 44 показано окно с обозревателем тестов в Microsoft Visual Studio со списком интеграционных тестов и успешным результатом их выполнения.

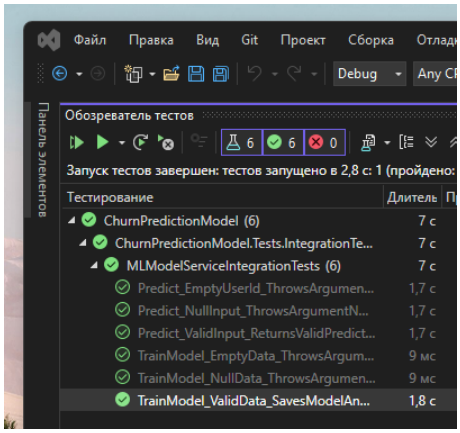


Рисунок 44 – Результат интеграционного тестирования

Интеграционное тестирование продемонстрировало стабильную и корректную работу компонентов сервиса машинного обучения при

тренировке, получении предсказаний и работе базой данных MongoDB. Все тестовые сценарии выполнены успешно.

Нагрузочное тестирование было проведено для оценки производительности API-сервера и методов TrainModel и Predict сервиса машинного обучения, с целью проверки их способности обрабатывать большие объёмы запросов и данных в рамках заданных временных ограничений. Тестирование API-сервера реализовано с помощью инструмента JMeter для эмуляции высокой нагрузки конечной точки обработки HTTP-запросов [19]. Тестирование методов TrainModel и Predict реализовано в классе MLModelServiceLoadTests с использованием библиотек Xunit, System.Diagnostics и System.Collections.Concurrent, а также фабрики CustomWebApplicationFactory для создания тестового окружения [23]. Ниже представлен процесс и результаты проведённых тестов.

Оценка производительности API-сервера при выполнении 10000 HTTP GET-запросов для получения списка решений, с ограничением среднего времени ответа не более 2 секунд. Процесс:

Запущен API-сервер;

В JMeter был создан тестовый план, включающий Thread Group с 10000 потоками, временным периодом в 2 секунды для запуска всех запросов, HTTP Request с указанной конечной точкой и типом HTTP-запроса, Summary Report для анализа результатов теста [19];

Выполнен запуск теста.

Результат теста, а также логи API-сервера представлены на рисунке 45. Успешно выполнено 10000 запросов, минимальное время ответа – 2 мс., среднее время ответа – 8 мс., максимальное время ответа – 294 мс., процент ошибок – 0 [19]. Среднее и максимальное время ответа значительно ниже установленного временного лимита в 2 секунды. Ошибки во время выполнения отсутствуют. Результаты теста полностью соответствуют заданным требованиям.

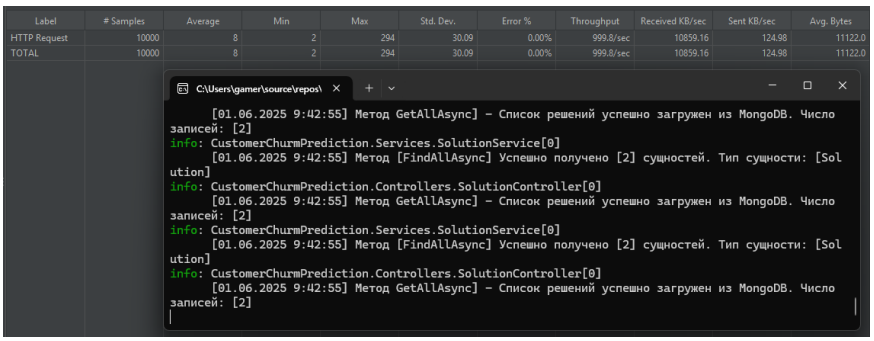


Рисунок 45 – Результат нагрузочного тестирования конечной точки API-сервера для загрузки списка решений

Оценка производительности метода TrainModel при обработке выборки из 10000 записей, с ограничением времени выполнения не более 10 минут.

Процесс:

– Сгенерировано 10000 синтетических записей;
– Выполнен метод TrainModel с использованием класса Stopwatch для измерения времени выполнения.

Результат: тестирование заняло 7 секунд, что полностью соответствует заявленным ограничениям в 10 минут.

Оценка производительности метода Predict при параллельной обработке 10000 предсказаний, с проверкой, с ограничением времени выполнения не более 3 минут.

Процесс:

– Модель предварительно обучена на 10000 синтетически сгенерированных записях [14];
– Сгенерировано 10000 записей для тестирования метода;
– Выполнен Predict параллельно с использованием Parallel.ForEach и Partitioner.Create, с измерением времени с помощью класса Stopwatch [22].

Результат: тестирование заняло 17,5 секунд, что полностью соответствует заявленным ограничениям в 3 минуты.

На рисунке 46 показано окно с обозревателем тестов в Microsoft Visual Studio со списком нагрузочных тестов и успешным результатом их выполнения.

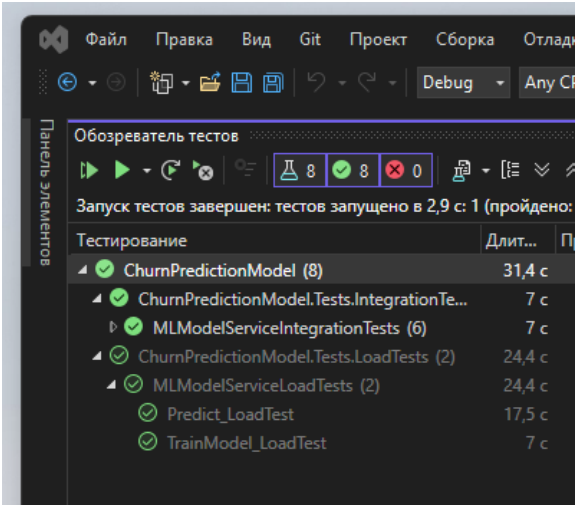


Рисунок 46 – Результаты выполнения нагрузочного тестирования методов TrainModel и Predict

Нагрузочное тестирование подтвердило высокую производительность API-сервера и методов TrainModel и Predict. API-сервер успешно выдерживает нагрузку в 10000 HTTP GET-запросов с средним временем ответа 8 мс., что значительно ниже установленного лимита в 2 секунды. Методы TrainModel и Predict эффективно обрабатывают заданные объёмы данных в 10000 записей в рамках заданных временных ограничений.

Выводы по главе 3

Реализована клиентская и серверная части сайта ИТ-компании, разработан механизм отслеживания активности пользователей. Реализован полный цикл разработки ML-модели. Сформированы признаки и критерии оттока клиентов, подготовлена выборка с учётом несбалансированности данных, построен pipeline в ML.NET, выбрана на основании метрик и обучена модель LightGBM с точностью более 85%, разработаны фоновые сервисы для обработки данных, обучения и получения предсказаний, а также проведена

интеграция сервиса машинного обучения с сайтом ИТ-компании. Разработана панель администратора и система уведомлений, предназначенные для анализа оттока клиентов, отправки и получения персонализированных уведомлений. Тестирование системы, включающее функциональное, интеграционное и нагрузочное, подтвердило её высокую надёжность и соответствие заданным требованиям. Функциональное тестирование успешно продемонстрировало корректную работу ключевых функций системы, включая регистрацию и аутентификацию, отображение списка предсказаний, отправку запроса на консультацию, отслеживание действий пользователя, работу фоновых сервисов, отображение предсказаний и метрик, а также отправку уведомлений. Интеграционное тестирование обеспечило проверку взаимодействия компонентов с моделью машинного обучения и базой данных MongoDB, включая обработку валидных и некорректных данных. Нагрузочное тестирование показало, что API-сервер и методы TrainModel и Predict способны эффективно справляться с высокой нагрузкой на систему.

Заключение

Результатом выполнения выпускной квалификационной работы (ВКР) является разработанный сайт ИТ-компании, предоставляющий услуги CRM и HRM систем, использующий алгоритмы машинного обучения для анализа активности пользователей, предсказания оттока клиентов и предоставления персонализированных уведомлений с целью их удержания.

В главе 1 в результате анализа предметной области поставлена задача на разработку собственного решения, изучены инструменты, используемые в процессе проектирования, разработки и тестирования, проведён анализ аналогов программного обеспечения и определены функциональные требования к системе.

В главе 2 спроектирована архитектура системы, посредством построения диаграммы развёртывания, диаграммы компонентов и диаграммы вариантов использования. Спроектирована и построена база данных MongoDB. Разработана физическая модель база данных, построена диаграмма классов, описывающая их связи, а также представлен поток данных в системе, описывающий процесс передачи, обработки и хранения данных.

В главе 3 реализована клиентская и серверная части сайта ИТ-компании, разработан механизм отслеживания активности пользователей. Реализован полный цикл разработки ML-модели. Сформированы признаки и критерии оттока клиентов, подготовлена выборка с учётом несбалансированности данных, построен pipeline в ML.NET, выбрана на основании метрик и обучена модель LightGBM с точностью более 85%, разработаны фоновые сервисы для обработки данных, обучения и получения предсказаний, а также проведена интеграция сервиса машинного обучения с сайтом ИТ-компании. Разработана панель администратора и система уведомлений, предназначенные для анализа оттока клиентов, отправки и получения персонализированных уведомлений. Проведено тестирование системы, включая функциональное, интеграционное и нагрузочное. Функциональное тестирование показало корректную работу

ключевых функций системы. Интеграционное тестирование протестировало методы для тренировки ML-модели и получения предсказаний. Нагрузочное тестирование продемонстрировало способность системы обрабатывать большие данные в заявленные временные лимиты.

Разработанный сайт ИТ-компании, использующий машинное обучение для прогнозирования оттока клиентов, является актуальным решением в условиях конкуренции и цифровизации. Он позволяет оперативно выявлять пользователей с высоким риском ухода, предоставлять подробную аналитику метрик пользователей через панель администратора и отправлять персонализированные уведомления, что повышает лояльность клиентов и увеличивает доходы компании за счёт удержания их в системе. Сайт решает проблемы ручного анализа данных и недостаточной автоматизации в процессе удержания клиентов, обеспечивая анализ активности пользователей с помощью машинного обучения и предоставляя механизм отправки персонализированных уведомлений.

Список используемой литературы и используемых источнков

1. Авторизация на основе ролей на платформе ASP.NET Core [Электронный ресурс]. URL: https://learn.microsoft.com/en-us/aspnet/core/security/authorization/roles?view=aspnetcore-9.0&source=recommendations (дата обращения: 26.02.2025).
2. Буч Г. Объектно-ориентированный анализ и проектирование с примерами приложений. Третье издание – изд.: М.: «Вильямс», 2010. - 720 с.
3. Интеграционные тесты в платформе ASP.NET Core [Электронный ресурс]. URL: https://learn.microsoft.com/en-us/aspnet/core/test/integration-tests?view=aspnetcore-9.0&pivots=xunit (дата обращения: 20.03.2025).
4. Мартин Р. Чистая архитектура. Искусство разработки программного обеспечения – СПб.: Питер, 2021. – 352 с.
5. Настройка аутентификации с использованием JWT в ASP.NET Core [Электронный ресурс]. URL: https://learn.microsoft.com/en-us/aspnet/core/security/authentication/configure-jwt-bearer-authentication?view=aspnetcore-9.0 (дата обращения: 25.02.2025).
6. Официальная документация базы данных MongoDB [Электронный ресурс]. URL: https://www.mongodb.com/docs/ (дата обращения: 11.01.2025).
7. Официальная документация библиотеки Axios [Электронный ресурс]. URL: https://axios-http.com/docs/intro (дата обращения: 10.03.2025).
8. Официальная документация библиотеки Bogus для .NET [Электронный ресурс]. URL: https://github.com/bchavez/Bogus (дата обращения: 27.02.2025).
9. Официальная документация библиотеки Bootstrap [Электронный ресурс]. URL: https://getbootstrap.com/docs/5.3/getting-started/introduction/ (дата обращения: 21.04.2025).

10. Официальная документация библиотеки FluentAssertions [Электронный ресурс]. URL: https://fluentassertions.com/introduction (дата обращения: 19.03.2025).

11. Официальная документация библиотеки Microsoft.ML (ML.NET) [Электронный ресурс]. URL: https://learn.microsoft.com/en-us/dotnet/machine-learning/ (дата обращения: 15.02.2025).

12. Официальная документация библиотеки Microsoft.ML.LightGbm [Электронный ресурс]. URL: https://www.nuget.org/packages/Microsoft.ML.LightGbm/ (дата обращения: 17.02.2025).

13. Официальная документация библиотеки Mongo2Go [Электронный ресурс]. URL: https://github.com/Mongo2Go/Mongo2Go (дата обращения: 20.03.2025).

14. Официальная документация библиотеки Moq [Электронный ресурс]. URL: https://github.com/devlooped/moq (дата обращения: 22.03.2025).

15. Официальная документация библиотеки Newtonsoft.Json для .NET [Электронный ресурс]. URL: https://www.newtonsoft.com/json/help/html/Introduction.htm (дата обращения: 12.03.2025).

16. Официальная документация библиотеки React [Электронный ресурс]. URL: https://react.dev/learn (дата обращения: 09.03.2025).

17. Официальная документация библиотеки React Router DOM [Электронный ресурс]. URL: https://reactrouter.com/home (дата обращения: 09.03.2025).

18. Официальная документация драйвера MongoDB для C#/.NET [Электронный ресурс]. URL: https://www.mongodb.com/docs/drivers/csharp/ (дата обращения: 12.01.2025).

19. Официальная документация инструмента Apache JMeter [Электронный ресурс]. URL: https://jmeter.apache.org/usermanual/index.html (дата обращения: 21.04.2025).

20. Официальная документация платформы ASP.NET Core [Электронный ресурс]. URL: https://learn.microsoft.com/en-us/aspnet/core/ (дата обращения: 08.01.2025).

21. Официальная документация по платформе идентификации Microsoft Identity Platform [Электронный ресурс]. URL: https://learn.microsoft.com/en-us/entra/identity-platform/ (дата обращения: 11.03.2025).

22. Официальная документация по C# [Электронный ресурс]. URL: https://learn.microsoft.com/en-us/dotnet/csharp/ (дата обращения: 07.01.2025).

23. Официальная документация по xUnit [Электронный ресурс]. URL: https://xunit.net/ (дата обращения: 19.03.2025).

24. Официальная стандартная документация по JavaScript (ECMAScript) [Электронный ресурс]. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript (дата обращения: 17.03.2025).

25. Рихтер Дж. CLR via C# Программирование на платформе Microsoft .NET Framework 4.5 на языке C#. 4-е изд. – СПб.: Питер, 2023 - 896 с.: ил - (Серия «Мастер-класс»).

26. Стефанов С. React. Быстрый старт, 2-е изд. – СПб.: Питер, 2023. – 304 с.: ил. – (Серия «Бестселлеры O’Reilly»).

27. Фримен А. ASP.NET Core 3 с примерами на C# для профессионалов, 8-е изд.: Пер. с англ. - СПб.: ООО «Диалектика», 2021. – 1184 с.