Министерство науки и высшего образования Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ»

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

РАЗРАБОТКА ПЛАТФОРМЫ ДЛЯ УПРАВЛЕНИЯ ТАРГЕТИРОВАННОЙ РЕКЛАМОЙ

Автор: Чекменев Александр Романович _____

Направление подготовки: 01.04.02 Прикладная математика и информатика

Квалификация: Магистр

Руководитель: Фильченков А. А., к.ф.-м.н. _____

К защите допустить

Руководитель ОП Парфенов В.Г., проф., д.т.н. _____

«____» _____ 20___ г.

Санкт-Петербург, 2020 г.

Студент Чекменев А.Р. Группа М42381 Факультет ИТиП

Направленность (профиль), специализация
Технологии разработки программного обеспечения

Консультанты:

а) Шевченко Е.А., ведущий разработчик, Fulldive corp. _____

ВКР принята «____» _____ 20___ г.

Оригинальность ВКР _____%

ВКР выполнена с оценкой _____

Дата защиты «____» _____ 20___ г.

Секретарь ГЭК Павлова О.Н. _____

Листов хранения _____

Демонстрационных материалов/Чертежей хранения _____

Министерство науки и высшего образования Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ»

УТВЕРЖДАЮ

Руководитель ОП

проф., д.т.н. Парфенов В.Г. _____

«____» _____ 20___ г.

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ

Студент Чекменев А.Р. **Группа** М42381 **Факультет** ИТиП

Руководитель Фильченков А. А., к.ф.-м.н., доцент ФИТиП

1 Наименование темы: Разработка платформы для управления таргетированной рекламой

Направление подготовки (специальность): 01.04.02 Прикладная математика и информатика

Направленность (профиль): Технологии разработки программного обеспечения

Квалификация: Магистр

2 Срок сдачи студентом законченной работы: «01» июня 2020 г.

3 Техническое задание и исходные данные к работе

Требуется разработать рекламный сервер, обеспечивающий функционирование рекламной сети согласно установленным техническим и бизнес-требованиям.

4 Содержание выпускной работы (перечень подлежащих разработке вопросов)

Описание архитектуры рекламного сервера, устройство отдельных модулей бизнес-логики, устройство хранения данных в MongoDB, Redis, ClickHouse и процесс развертывания микросервиса в кластере Google Kubernetes Engine.

5 Перечень графического материала (с указанием обязательного материала)

Графические материалы и чертежи работой не предусмотрены

6 Исходные материалы и пособия

нет

7 Дата выдачи задания «01» сентября 2019 г.

Руководитель ВКР _____

Задание принял к исполнению _____

«01» сентября 2019 г.

Министерство науки и высшего образования Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ»

АННОТАЦИЯ
ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ

Студент: Чекменев Александр Романович

Наименование темы ВКР: Разработка платформы для управления таргетированной рекламой

Наименование организации, в которой выполнена ВКР: Университет ИТМО

ХАРАКТЕРИСТИКА ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ

1 Цель исследования: Разработка рекламного сервера для функционирования тергетированной рекламной сети

2 Задачи, решаемые в ВКР:

а) разработка контракта клиент-серверного взаимодействия;
б) разработка архитектуры серверного приложения;
в) разработка модуля для создания и обновления рекламных объявлений на основе товаров Amazon;
г) разработка механизма получения списка объявлений с сохранением консистентности;
д) разработка алгоритмов работы стратегий таргетирования объявлений;
е) разработка системы трекинга просмотров и переходов по объявлениям с возможностью просмотра статистики по объявлениям и стратегиям таргетинга.

3 Число источников, использованных при составлении обзора: 7

4 Полное число источников, использованных в работе: 28

5 В том числе источников по годам:

Отечественных			**Иностранных**		
Последние 5 лет	От 5 до 10 лет	Более 10 лет	Последние 5 лет	От 5 до 10 лет	Более 10 лет
0	0	0	28	0	0

6 Использование информационных ресурсов Internet: да, число ресурсов: 28

7 Использование современных пакетов компьютерных программ и технологий: нет

8 Краткая характеристика полученных результатов

Рекламный сервер запущен в production-окружении в кластере Google Kubernetes Engine. Примеры рекламных объявлений различного формата доступны в приложении Fulldive Browser на платформах iOS и Android.

9 Гранты, полученные при выполнении работы

нет

10 Наличие публикаций и выступлений на конференциях по теме работы

нет

Студент Чекменев А.Р. _____

Руководитель Фильченков А. А. _____

«____» _____ 20__ г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Прогресс в сфере технологий и интеллектуальных систем неизбежно влечет за собой активное развитие всех смежных областей, а достижения вследствие такого развития внедряются и в другие сферы. Сфера digital-рекламы в этом плане не является исключением.

Результатом применения современных подходов в digital-рекламе является, например, алгоритмическая закупка рекламы (англ. programmatic buying) [22] — способ закупки целевого трафика с оплатой за совершенные целевые действия. Данный способ активно вытесняет традиционные способы закупки рекламы, так как обладает рядом преимуществ. Например, позволяет в режиме реального времени за долю секунды подобрать специальное рекламное объявление для каждого конкретного пользователя с учетом его демографических данных, места жительства, интересов и прочих параметров. По прогнозам маркетинговых агентств в 2020 году 69 % digital-рекламы будет закуплено программно [10].

Одной из причин столь активного вытеснения ранее существующих подхов стало повышение эффективности алгоритмов таргетинга.

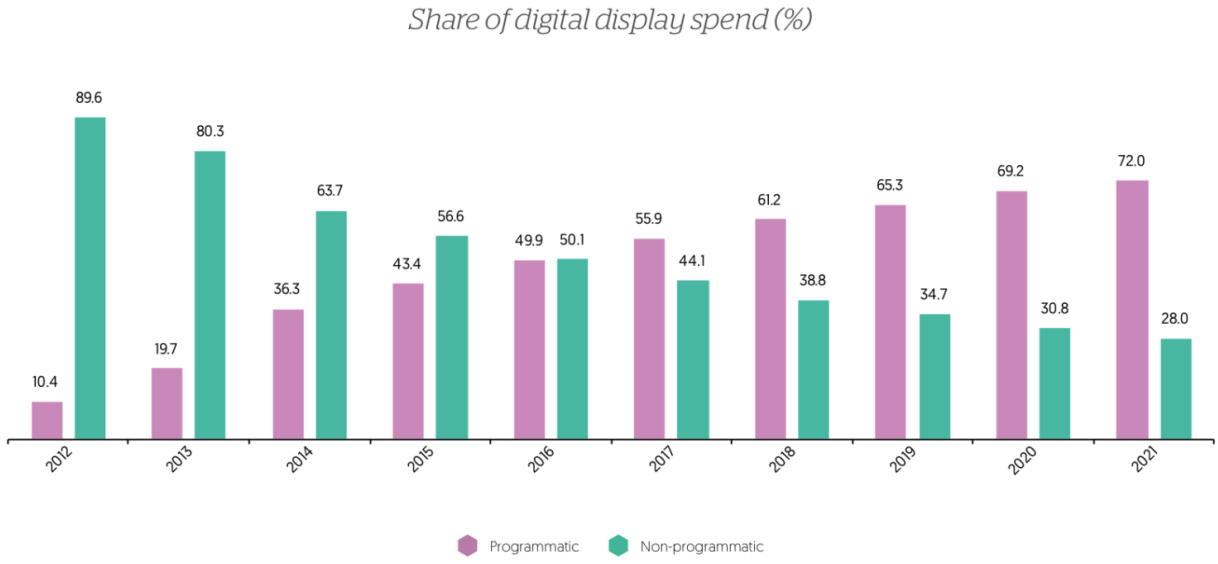

Рисунок 1 – Рост доли алгоритмической закупки рекламы

В данный момент таргетированная реклама одна из наиболее развивающихся областей digital-рекламы, так как позволяет максимально гибко и точно задать необходимую целевую аудиторию для каждого рекламного объявления. Существует ряд компаний, предоставляющих услуги по размещению таргети-

рованной рекламы в своих рекламных сетях. Но все они имеют такие существенные недостатки, как отсутствие прямого доступа к целевой аудитории, а также строгая зависимость от условий, устанавливаемых рекламными площадками. Все это создает трудности в воплощении нестандартных подходов для компаний, которые по каким-либо причинам не могут выполнить установленные требования, и дает все основания задуматься о создании собственной рекламной сети.

Компания Fulldive разработала приложение Fulldive VR, которое входит в топ-5 приложений для Android в категории VR. Fulldive Browser — второй активно развивающийся продукт компании. Одним из его выжнейших и эффективно влияющих на привлечение новых пользователей преимуществом является возможность пользователя монетизировать свое время, проведенное за чтением новостей, общением в социальных сетях, просмотром роликов и другими действиями, типично совершаемыми в любом другом браузере. Достигается это за счет того, что компания отдает пользователю часть прибыли, полученной от рекламных сетей за просмотры рекламы, совершенные пользователем приложении.

Далеко не все рекламные площадки приветствуют такой способ мотивации пользователей, но реализация рекламы внутри своих приложений с использованием собственной рекламной сети поможет избежать данной проблемы. Это позволит компании в полной мере следовать своей стратегии развития, не боясь сокращения или потери доходов от рекламы в будущем, а также позволит не отдавать часть прибыли сторонним рекламным площадкам.

Таким образом, можно выделить следующие преимущества наличия у компании собственной рекламной сети:

а) диверсификация риска сокращения прибыли от рекламы из-за прекращения сотрудничества с основными рекламными сетями;
б) возможность добавлять собственные рекламные форматы;
в) возможность более тонко настраивать где и в каком контексте будет показана реклама;
г) возможность сбора данных непосредственного взаимодействия пользователей с рекламой (First-Party Data), которые представляют наивысшую ценность, так как является наиболее точными, качественными и принадлежат компании;

д) возможность использовать собственную рекламную сеть для привлечения сторонних рекламодателей.

Целью данной работы является создание собственной рекламной сети UniAds, для функционирования которой необходимо разработать рекламный сервер на основе установленных бизнес-требований.

В первой главе приводится список используемых терминов, формулируется цель и постановка решаемой задачи, рассматриваются существующие решения. В второй главе приводится архитектура предлагаемого решения поставленной задачи. В третьей главе рассматриваются аспекты, связанные с технической реализацией приведенного во второй главе решения. В заключении приводятся полученные результаты, подводятся итоги проделанной работы и предлагаются направления для дальнейшей деятельности.

ГЛАВА 1. ПОСТАНОВКА ЗАДАЧИ И ОБЗОР СУЩЕСТВУЮЩИХ РЕАЛИЗАЦИЙ РЕКЛАМНОГО СЕРВЕРА

В начале данной главы (п. 1.1) приведены определения для основных терминов и понятий, используемых в работе. Затем формулируются цель работы, постановка задачи с разбиением на необходимые подзадачи и требования к результатам проделанной работы (п. 1.2). В конце главы проводится обзор существующих решений, удовлетворяющих сформулированным требованиям (п. 1.3).

1.1. Используемые термины и понятия

1.1.1. Amazon

Amazon — это американская организация, одна из крупнейших в мире среди компаний, занимающихся продажей всевозможных товаров и услуг через сеть интернет. Также это лидер в области продаж товаром массового спроса через систему интернет-сервисов.

У Amazon есть собственная партнерская программа, в рамках которой всем парнерам начисляется от 1 до 5 % от суммы совершенных привлеченными пользователями покупок, в зависимости от уровня партнера и категории покупаемого товара.

1.1.2. Рекламная сеть

Рекламная сеть (англ. ad network) — это технологическая платформа, которая обеспечивает продажи рекламных ресурсов между издателями и рекламодателями. Ключевой функцией рекламной сети является агрегация рекламного предложения от издателей и его соответствие спросу рекламодателя. Принципиальное различие между традиционными рекламными сетями в средствах массовой информации и рекламными сетями в интернете заключается в том, что рекламные сети в интернете используют рекламные серверы для таргетирования и доставки рекламы потребителям, отслеживания и составления отчетов о показах способами, невозможными при использовании аналоговых средств массовой информации.

Таргетированная рекламная сеть (англ. targeted ad network) — наиболее продвинутый вид рекламных сетей, которые специализируются на технологиях точного таргетирования по поведенческим или контекстным признакам и активно анализируют данные о пользователях в целях повышения стоимости инвентаря, который они продают.

1.1.3. Рекламодатель

Рекламодатель (англ. advertiser) — участник рекламной сети, который создает, обновляет и оплачивает рекламу продукта или услуги и, фактически, представляющий сторону спроса.

Рекламодатель как пользователь рекламной сети:

- создает рекламные материалы различных форматов;
- выбирает модель оплаты и согласно ей оплачивает приобретенные у издателя ресурсы;
- конфигурирует целевую аудиторию для каждого управляемого им рекламного объявления;
- управляет категориями подходящих рекламных площадок и издателей.

1.1.4. Издатель

Издатель (англ. publisher) — владелец рекламной площадки, на которой рекламодатель размещает свою рекламу. За данную услугу издатель взимает с рекламодателя определенную сумму в соответствии с одной из ценовых моделей размещения рекламы. Чаще всего рекламной площадкой является веб-сайт или мобильное приложение.

Издатель как пользователь рекламной сети:

- регистрирует рекламную площадку и указывает ее параметры: тематику, тип и объем доступной аудитории и т.д.;
- создает элементы рекламного инвентаря, которые будут использованы для продажи рекламодателям доступные для показа рекламы места;
- управляет категориями допустимых рекламодателей.

1.1.5. Рекламный сервер

Рекламный сервер (англ. ad server) — это технологическая платформа (обычно веб-ориентированная), отвечающая за размещение, оптимизацию и распространение рекламного контента на различных веб-сайтах и в мобильных приложениях. Рекламные серверы также отвечают за управление и отслеживание рекламы, составление отчетов и выставление счетов рекламодателям.

Показ рекламы (англ. ad serving) — это технологический цикл, в котором рекламный сервер используется для размещения рекламы на различных веб-сайтах и в мобильных приложениях. Механизм показа рекламы является ключевым элементом каждого рекламного сервера. Он использует сложные

алгоритмы и передовые инструменты принятия решений для выбора наиболее релевантной рекламы для показа. Процесс выбора рекламы строго ограничен правилами, которые определены издателями, рекламодателями и самим рекламным сервером. Эти правила включают критерии таргетинга, частоту просмотра, место размещения рекламы, приоритет рекламы, потенциальный доход и другие параметры.

Рекламный сервер должен учитывать все эти факторы в режиме реального времени и возвращать результат в течение миллисекунд. Любая задержка приведет к дополнительному времени ожидания загрузки страницы, что приведет к меньшему количеству просмотров страниц и показов рекламы.

Существует два типа рекламных серверов: один — для рекламодателей, другой — для издателей (рекламных площадок). Их отличие заключается в организации данных и удобстве работы для каждой группы пользователей.

Рекламодатели используют единый сервер объявлений, где хранится рекламный контент и предоставляется функционал для отчетов по показам, кликам и другим метрикам. Такой централизованный сервис, контролирующий распространение контента на сайтах, делает удобным отслеживание и управление рекламными материалами, повышает точность таргетинга.

Издатели — владельцы площадок — имеют отдельные рекламные серверы (только для своих доменов). Это удобно, поскольку у них есть доступ только к тому рекламному контенту, который требуется для публикации, и не нужно фильтровать все рекламные материалы на централизованном сервере.

1.1.6. Таргетинг

Таргетинг (англ. ad targeting) — рекламный механизм, позволяющий выделить из всей имеющейся аудитории только ту часть, которая удовлетворяет заданным критериям (целевую аудиторию), и показать конкретное объявление именно ей.

Цель таргетинга — создать рекламно-информационное сообщение, по своей форме и содержанию максимально ориентированное на заинтересованную в конкретном товаре/услуге часть аудитории, а также повысить эффективность взаимодействия с этой аудиторией и получить как можно большей отдачи от неё.

Преимущества таргетинга:

— возможность выделить ту аудиторию, заинтересованную в покупке товара или услуги;
— возможность снизить расходы на рекламу за счёт правильного выбора таргетинга для целевой аудитории;
— возможность разделить всю целевую аудиторию на группы и для каждой группы сформировать уникальное предложение.

В зависимости от того, на основе какой информации принимается решение о выборе объявления, выделяют такие виды таргетинга, как контекстный и поведенческий.

Контекстный таргетинг (англ. contextual targeting) [4] — это практика показа рекламы на основе содержания веб-сайта. Существует контекстный таргетинг на категории, где объявления ориентированы на страницы, которые попадают в предварительно назначенные категории, и контекстный таргетинг на ключевые слова, где объявления ориентированы на страницы, соответствующие определенным ключевым словам.

Семантический таргетинг является наиболее эффективной формой контекстного таргетинга. Он предполагает использование машинного обучения для распознавания контекста и семантики каждой страницы, содержащей определенный контент, а не просто определение подходящих ключевых слов на странице. Когда пользователь заходит на страницу, информация о содержимом этой веб-страницы попадает на рекламный сервер, который затем сопоставляет ее с релевантными объявлениями по ключевым словам и содержанию. Чем лучше система понимает контекст страницы, тем релевантнее будет реклама.

Поведенческий таргетинг (англ. behavioral targeting) — это практика сегментирования клиентов на основе поведения при просмотре веб-страниц или действий в мобильном приложении.

Данные о поведении пользователя служат для:
— показа релевантной, персонализированной рекламы в тот момент, когда покупатель, скорее всего, совершит покупку;
— изучения интересов, предпочтений и планов потенциальных покупателей;
— формирования портретов разных сегментов целевой аудитории
— повышения лояльности к продукту и отклика на рекламу;

— расширения целевой аудитории.

Поведенческий таргетинг позволяет показывать объявления пользователям, которые действительно заинтересованы в рекламируемом товаре или услуге. Обратиться к целевой аудитории можно даже на тех страницах, содержание которых не соответствует тематике объявления.

1.1.7. Трекинг

Трекинг (англ. ad tracking) или отслеживание рекламы — фиксация различных событий взаимодействия пользователя с рекламой. Данными событиями, например, являются показы, клики и конверсии. Все они записываются в базу данных для дальнейшей обработки и анализа. Процесс отслеживания рекламы важен, так как помогает измерить и оценить эффективность каждого объявления.

Конверсионная воронка — это последовательность совершенных на сайте пользователем действий (микроконверсий), ведущих его к конечной цели (макроконверсии), визуализированная в виде воронки, т.к. достижение каждой следующей микроконверсии в количественном выражении меньше, чем достижения предыдущей. В зависимости от того, влияют ли результаты предыдущих стадий на последующие, воронки могут быть одного из двух типов: открытые или закрытые.

Метрики — параметры, рассчитываемые на основе данных трекинга. На основе таких параметров проводится прогнозирование и анализ эффективности рекламных кампаний.

Метрики, используемые в данной работе:

— OR (англ. open rate) — отношение числа показов рекламы к числу запросов рекламных объявлений;
— CTR (англ. click-through rate) — отношение числа кликов на рекламное объявление к числу показов;
— CPA (англ. cost per action) — отношение общей стоимости рекламы к количеству совершенных целевых действий; стоимость целевого действия.

1.1.8. Клиент-серверная архитектура

«Клиент — сервер» (англ. client–server) — вычислительная или сетевая архитектура, в которой задания или сетевая нагрузка распределены между поставщиками услуг, называемыми серверами, и заказчиками услуг, называемыми клиентами. Фактически, клиент и сервер — это программное обеспечение.

Обычно эти программы расположены на разных вычислительных машинах и взаимодействуют между собой через вычислительную сеть посредством сетевых протоколов, но они могут быть расположены также и на одной машине. Программы-серверы ожидают от клиентских программ запросы и предоставляют им свои ресурсы в виде данных или в виде сервисных функций.

API — это вычислительный интерфейс, который является контрактом взаимодействия нескольких программных комплексов. API определяет функциональность, которую предоставляет программа (модуль, библиотека), при этом API позволяет абстрагироваться от того, как именно эта функциональность реализована.

1.1.9. Микросервисная архитектура

Микросервисная архитектура — вариант сервис-ориентированной архитектуры программного обеспечения, направленный на взаимодействие насколько это возможно небольших, слабо связанных и легко изменяемых модулей — микросервисов, получивший распространение в середине 2010-х годов в связи с развитием практик гибкой разработки и DevOps.

Свойства, характерные для микросервисной архитектуры:

— модули можно легко заменить в любое время: акцент на простоту, независимость развёртывания и обновления каждого из микросервисов;
— модули организованы вокруг функций: микросервис по возможности выполняет только одну достаточно элементарную функцию;
— модули могут быть реализованы с использованием различных языков программирования, фреймворков, связующего программного обеспечения, выполняться в различных средах контейнеризации, виртуализации, под управлением различных операционных систем на различных аппаратных платформах: приоритет отдаётся в пользу наибольшей эффективности для каждой конкретной функции, нежели стандартизации средств разработки и исполнения;
— архитектура симметричная, а не иерархическая: зависимости между микросервисами одноранговые.

1.1.10. Трехуровневая архитектура

Трёхуровневая архитектура (англ. three-tier) — архитектурная модель программного комплекса, предполагающая наличие в нём трёх компонентов:

клиента, сервера приложений (к которому подключено клиентское приложение) и сервера баз данных (с которым работает сервер приложений).

Клиент (слой клиента) — это интерфейсный компонент комплекса, предоставляемый конечному пользователю. Этот уровень не должен иметь прямых связей с базой данных, быть нагруженным основной бизнес-логикой и хранить состояние приложения. На этот уровень обычно выносится только простейшая бизнес-логика.

Сервер приложений (средний слой, связующий слой) располагается на втором уровне, на нём сосредоточена большая часть бизнес-логики. Серверы приложений проектируются таким образом, чтобы добавление к ним дополнительных экземпляров обеспечивало горизонтальное масштабирование производительности программного комплекса и не требовало внесения изменений в программный код приложения.

Сервер баз данных (слой данных) обеспечивает хранение данных и выносится на отдельный уровень, реализуется, как правило, средствами систем управления базами данных, подключение к этому компоненту обеспечивается только с уровня сервера приложений.

1.1.11. Платформа Node.js

Node.js — программная платформа, основанная на движке V8 (транслирующем JavaScript в машинный код), превращающая JavaScript из узкоспециализированного языка в язык общего назначения. Node.js добавляет возможность JavaScript взаимодействовать с устройствами ввода-вывода через свой API, подключать другие внешние библиотеки, написанные на разных языках, обеспечивая вызовы к ним из JavaScript-кода. Node.js применяется преимущественно на сервере, выполняя роль веб-сервера. В основе Node.js лежит событийно-ориентированное и асинхронное (или реактивное) программирование с неблокирующим вводом/выводом.

NPM (изначально сокращение от Node Package Manager) — менеджер пакетов для языка программирования JavaScript. Этот менеджер пакетов используется по умолчанию для исполняющей среды Node.js.

1.1.12. TypeScript

TypeScript — язык программирования, представленный компанией Microsoft в 2012 году и позиционируемый как средство разработки веб-

приложений, расширяющее возможности JavaScript. TypeScript является обратно совместимым с JavaScript и компилируется в последний. Фактически, после компиляции программу на TypeScript можно выполнять в любом современном браузере или использовать совместно с серверной платформой Node.js.

1.1.13. SOLID

SOLID — это аббревиатура пяти основных принципов проектирования в объектно-ориентированном программировании.

Расшифровка аббревиатуры:

— Single responsibility — принцип единственной ответственности;
— Open-closed — принцип открытости / закрытости;
— Liskov substitution — принцип подстановки Барбары Лисков;
— Interface segregation — принцип разделения интерфейса;
— Dependency inversion — принцип инверсии зависимостей.

Эти принципы позволяют строить на базе ООП масштабируемые и сопровождаемые программные продукты с понятной бизнес-логикой.

1.1.14. Базы данных

MongoDB — документоориентированная система управления базами данных с открытым исходным кодом, не требующая описания схемы таблиц. Классифицирована как NoSQL, использует JSON-подобные документы и схему базы данных. Написана на языке C++. Используется в веб-разработке, в частности, в рамках JavaScript-ориентированного стека MEAN.

Redis — быстрое хранилище в памяти с открытым исходным кодом для структур данных «ключ-значение». Redis поставляется с набором разнообразных структур данных в памяти, что упрощает создание различных специальных приложений. Самые распространенные примеры использования Redis включают кэширование, управление сессиями, системы «издатель-подписчик» и таблицы лидеров. Это самое популярное на текущий момент хранилище пар «ключ-значение».

Благодаря высокой скорости и простоте Redis часто используется для мобильных и интернет-приложений, игр, рекламных платформ, «Интернета вещей», т. е. в тех случаях, когда необходима максимально возможная производительность.

ClickHouse — это колоночная аналитическая СУБД с открытым кодом, позволяющая выполнять аналитические запросы в режиме реального времени на структурированных больших данных, разрабатываемая компанией Яндекс.

1.1.15. Развертывание

Docker — программное обеспечение для автоматизации развёртывания и управления приложениями в средах с поддержкой контейнеризации. Позволяет «упаковать» приложение со всем его окружением и зависимостями в контейнер, который может быть перенесён на любую Linux-систему с поддержкой cgroups в ядре, а также предоставляет среду по управлению контейнерами.

Kubernetes — это портативная расширяемая платформа с открытым исходным кодом для управления контейнеризованными рабочими нагрузками и сервисами, которая облегчает как декларативную настройку, так и автоматизацию. У платформы есть большая, быстро растущая экосистема. Сервисы, поддержка и инструменты Kubernetes широко доступны. Поддерживает основные технологии контейнеризации, включая Docker, rkt, также возможна поддержка технологий аппаратной виртуализации.

Google Cloud Platform — предоставляемый компанией Google набор облачных служб, которые выполняются на той же самой инфраструктуре, которую Google использует для своих продуктов, предназначенных для конечных потребителей, таких как Google Search и YouTube. Кроме инструментов для управления, также предоставляется ряд модульных облачных служб, таких как облачные вычисления, хранение данных, анализ данных, машинное обучение и многие другие.

Google Kubernetes Engine — мощная система управления и оркестрации контейнеров Docker, выполняемых в облачной инфраструктуре Google. GKE составляет контейнеры в кластеры и управляет ими автоматически, опираясь на определенные вами требования. Google Kubernetes Engine основан на системе с открытым исходным кодом Kubernetes.

1.2. Постановка задачи

Fulldive Browser — браузер, который, помимо втроенных возможностей социальных сетей, позволяет пользователям получать часть дохода компании от просмотренной рекламы. В приложении используется таргетированная реклама, закупаемая у сторонних рекламных сетей, таких как Google AdMob,

Facebook Audience Network, MoPub, PubMatic и других. Практически каждая из приведенных рекламных сетей либо запрещает возвращать часть доходов пользователям приложения, либо накладывает существенные ограничения на объем и качество поставляемой рекламы. Подобные ограничения являются вполне обоснованным решением для создания собственной рекламной сети.

Целью данной работы является создание собственной рекламной сети UniAds, для функционирования которой необходимо разработать рекламный сервер на основе установленных бизнес-требований.

Разработка рекламного сервера (п. 1.1.5) сводится к решению следующих подзадач:

— разработка контракта клиент-серверного взаимодействия;
— разработка архитектуры серверного приложения;
— разработка модуля для создания и обновления рекламных объявлений на основе товаров Amazon (п. 1.1.1);
— разработка механизма получения списка объявлений с сохранением консистентности;
— разработка алгоритмов работы стратегий таргетирования объявлений;
— разработка системы трекинга просмотров и переходов по объявлениям с возможностью просмотра статистики по объявлениям и стратегиям таргетинга.

Каждая из позадач подразумевает составление требований, разработку архитектуры необходимых модулей и их взаимодействия, программную реализацию, соответствующее тестирование полученных модулей и внедрение.

Основными требованиями к рекламному серверу являются возможность непосредственного доступа сотрудников компании для модификации исходного кода сервера, возможность развертывания в контролируемом окружении в облачной инфраструктуре Google Cloud Platform (п. 1.1.15), а также желательно использование платформы Node.js (п. 1.1.11) при реализации сервера. Требования к каждому отдельному модулю приводятся в в описании соответствующего модуля.

1.3. Обзор существующих решений

Для применения наиболее эффективных и актуальных подходов при решении приведенных подзадач рассмотрим наиболее популярные (по рейтингу Capterra [3]) существующие реализации функционала рекламного сервера.

Revive Adserver [8] — единственная open-source реализация на языке PHP, разработанная много лет назад в компании OpenX. Данный продукт хорошо подходит компаниям, которые:

— не хотят платить ежемесечную абонентскую плату за использование продукта;
— обладают собственными инженерными ресурсами;
— хотят существенно сэкономить ресурсы на разработку собственного решения.

И в то же время данный продукт:

— согласно документации не позволяет использовать собственные алгоритмы таргетинга;
— обладает слабой гибкостью в использовании собственных рекламных форматов;
— обладает очень скудной и местами устаревшей документаций, что затрудняет процесс интеграции данного решения в экосистему компании.

В июне 2018 года Google объявила о новом брендинге для ряда своих рекламных продуктов. Благодаря этой инициативе они объединили свой рекламный сервер DoubleClick For Publishers со своей биржей рекламы Google Ad Exchange в единую платформу под названием Google Ad Manager (GAM) [9]. Этот инструмент призван помочь как рекламодателям, так и издателям оптимизировать процесс показа рекламы, позволяя брендам управлять рекламой и доставлять ее в различные аудитории с одной платформы.

Данный инструмент обладает такими преимуществами, как высокая степень отказоустойчивости, тесная взаимосвязь с другими сервисами компании Google и подробная документация. Но в то же время его существенными недостатками являются:

— размещение на серверах компании Google без доступа к исходному коду;
— низкий уровень гибкости форматов рекламных объявлений;
— существенная ограниченность в применении собственных алгоритмов таргетинга;
— пользовательские данные хранятся на серверах Google.

Adzerk [1] — это облачная платформа для разработки рекламных серверов. Платформа предлагает различные API для облегчения проектирования, разработки и развертывания всех видов рекламных серверов. Adzerk вклю-

чает в себя несколько систем управления базами данных, таких как UserDB и ContentDB, чтобы помочь пользователям управлять ограничением частоты, поведенческим таргетингом, таргетингом на страницы, динамическим текстом объявлений и хранением информации о веб-страницах. Модуль принятия решений Adzerk предоставляет алгоритм принятия решений, который помогает пользователям управлять таргетингом кампаний и оптимизировать доходы от рекламы.

Существенными недостатками, приводящими к невозможности использования данного сервиса компанией для достижения поставленной цели, являются:

— интеграция при помощи API, что не позволяет получить прямой доступ к исходному коду даже в случае размещения на собственных серверах;
— и как следствие стандартизации программных компомент, существенно ограниченный уровень их гибкости.

EPOM Ad Server [7] — это рекламный сервер, который обслуживает рекламные сети. Ключевыми особенностями данного сервера являются возможность создания white-label рекламной сети, широкий спектр форматов рекламных объявлений, расширенные возможности отслеживания показателей рекламных кампаний. Также имеется возможность индивидуальной разработки необходимого функционала. Платформа размещается на серверах компании EPOM, что ограничивает доступ к исходному коду для самостоятельного внесения изменений.

MoPub [19] — платформа для монетизации мобильных приложений, принадлежащая компании Twitter. Платформа обладает всеми стандартными для рекламного сервера функциями и широко используется многими крупными компаниями. Имеются ограничения с гибкостью настройки таргетинга, а также полное отсутствие поведенческого таргетинга и таргетинга на основе содержания веб-страницы, которые на данный момент являются одними из наиболее эффективных. Также не предоставляется доступ к исходному коду платформы.

В качестве итогов проведенного обзора существующих реализаций можно привести следующие выводы:

— только одна из приведенных реализаций (Revive Adserver) является open-source решением с полноценным доступом к исходному коду, но исполь-

зует язык программирования PHP плоохо совместимый с экосистемой микросервисов компании Fulldive;

— максимально подходящим решением является Adzerk, которое не предоставляет доступ к исходному коду и не позволяет развернуть рекламный сервер в собственной инфраструктуре;
— ни один из рассмотренных вариантов реализации не позволяет получать список рекламных объявлений с сохранением консистентности.

Выводы по главе 1

В ходе первой главы приведены определения для основных терминов и понятий, сформулированы цель, постановка задачи и подзадачи, которые необходимо решить для достижения цели. Перечислены основные требования, которым должен соответствовать рекламный сервер. Произведен обзор существующих реализаций рекламного сервера с указанием основных преимуществ и недостатков, и подведены итоги по соответствию каждого из рассмотренных решений сформулированным требованиям.

ГЛАВА 2. ОПИСАНИЕ ПРЕДЛОЖЕННОГО РЕШЕНИЯ

В данной главе описываются основные компоненты рекламного сервера, клиент-серверное взаимодействие, модели данных, модули бизнес-логики и сценарии использования.

2.1. Клиент-серверное взаимодействие

Взаимодействие клиента и сервера происходит по заранее установленному контракту, реализованному поверх HTTP протокола и представляет из себя несколько сценариев.

Основные сценарии взаимодействия:

— издатель запрашивает одно рекламное объявление;
— издатель запрашивает список объявлений;
— издатель сообщает о совершении просмотра или перехода по объявлению пользователем;
— рекламодатель запрашивает статистику по объявлениям;
— администратор запрашивает статистику по стратегиям таргетинга.

Совокупность приведенных сценариев взаимодействия формирует API, данные по которому передаются в формате JSON.

Разделение ответственности между клиентом и сервером происходит по модели распределенного представления данных.

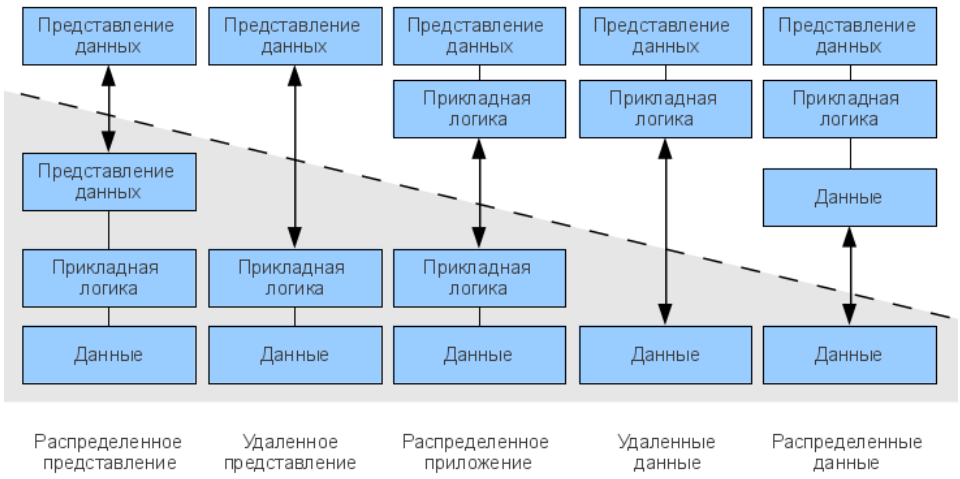

Рисунок 2 – Разделение ответственности между клиентом и сервером

Сервер отвечает за управление рекламным инвентарем и объявлениями, сбор статистики показов, переходов и конверсий, поддержание внутренних инвариантов для сохранения актуальности и консистентности данных. Клиент

отвечает за формирование запросов и корректное отображение полученных от сервера рекламных материалов и статистики.

2.2. Архитектура серверного приложения

Реализацией рекламного сервера является серверное приложение. В качестве архитектуры серверного приложения используется трёхуровневая архитектура (п. 1.1.10), в которой все функции разделяются на слои абстракции: слой представления, слой обработки и слой хранения данных. Разделяя приложение на слои абстракции, появляется возможность внесения изменений в отдельный слой, вместо того, чтобы перерабатывать всё приложение целиком. Трёхуровневая архитектура обычно состоит из слоя представления, слоя бизнес-логики и слоя доступа к данным.

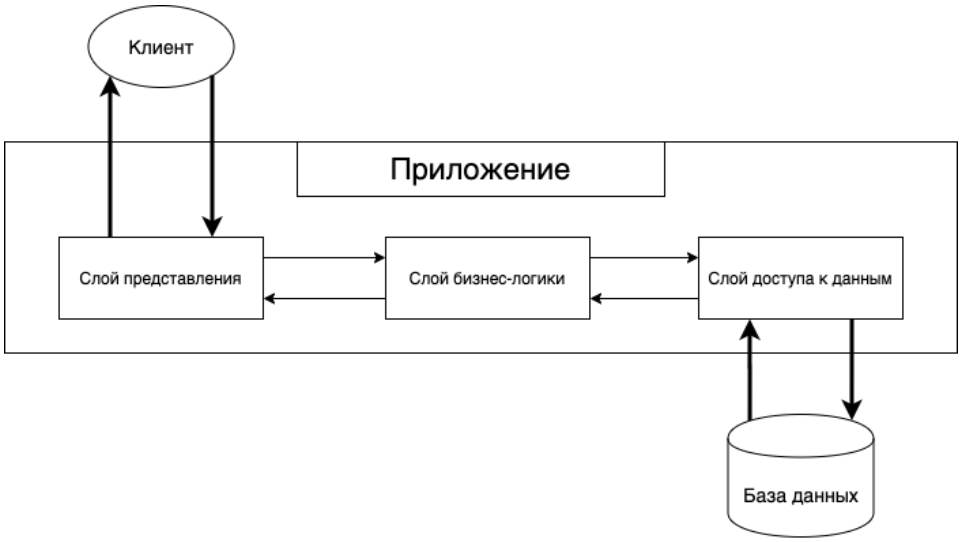

Рисунок 3 – Трёхуровневая архитектура

2.3. Модели

Перед тем как приступить к детальному описанию выбранной архитектуры, перечислим основные модели, которые представляют объекты предметной области, а затем последовательно рассмотрим каждый из слоев абстракции серверного приложения.

2.3.1. Модель AdProvider

Модель AdProvider представляет поставщика товаров, из которых впоследствии формируются рекламные объявления. Модель состоит из идентификатора провайдера, названия и ссылки на логотип для отображения ее в качестве небольшой иконки на объявлении, чтобы пользователь понимал с какой

площадки представлен конкретный товар. В контексте данной работы Amazon единственный поставщик товаров.

2.3.2. Модель Ad

Модель Ad представляет рекламное объявление и состоит из следующих полей:

— уникальный идентификатор объявления;
— идентификатор продукта;
— формат объявления;
— контент объявления, представленный ссылками на изображения и видео, используемыми для отображения объявления на клиенте;
— ссылка для перехода на рекламируемый продукт;
— идентификатор провайдера продукта;
— цена продукта;
— категория продукта.

Поддерживаемые форматы объявлений:

— нативный — настраиваемый формат, соответствующий дизайну приложения;
— баннер — платформонезависимый формат, представленный в формате HTML.

Каждый из вышепреведенных форматов также поддерживает объявления с видеоконтентом.

2.3.3. Модель StrategyState

Модель StrategyState представляет состояние стратегии таргетирования и состоит из следующих полей:

— идентификатор стратегии (одна из стратегий, описанных в п. 2.5.5);
— идентификтор потока (подробнее описан в п. 2.5.4);
— номер версии;
— набор пар (идентификатор объявления, скоринг объявления).

Ключ стратегии формируется из идентификатора стратегии, идентификатора потока и номера версии. Состояние стратегии предсталяет из себя набор пар идентификатор объявления и скоринг объявления, которые используются для задания порядка при запросе списка объявлений.

2.3.4. Модель Session

Модель Session представляет сессию пользователя, запрашивающего рекламу из приложения. Данная модель связывает ключ сессии с ключем состояния стратегии.

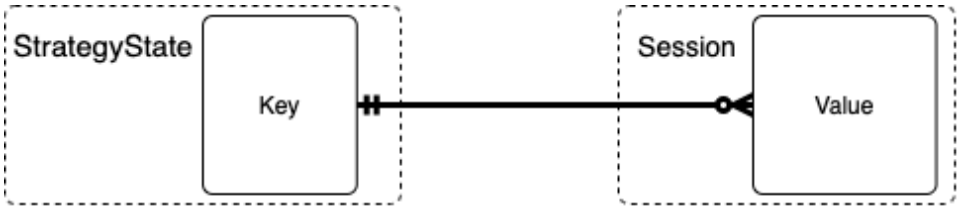

Рисунок 4 – Связь моделей StrategyState и Session

Сохранение ключа состояния стратегии в сессии пользователя позволяет достичь консистентности списка объявлений при пагинации, как минимум, на время равное длительности одной сессии.

2.3.5. Модели Track, Impression, Click

Модели Track, Impression и Click необходимы для представления данных статистики рекламных объявлений. Они хранят состояние на разных стадиях жизненного цикла объявления:

— запрошено клиентом — создается экземпляр Track;
— пользователь совершил просмотр объявления — создается экземпляр Impression, связанный с уже созданным экземпляром Track;
— пользователь совершил переход по ссылке в объявлении — создается экземпляр Click, связанный с уже созданным экземпляром Track.

Фиксирование события показа происходит на стороне клиента по истечении определенного времени (2-3 секунды), когда не менее 50 % площади объявления видны пользователю.

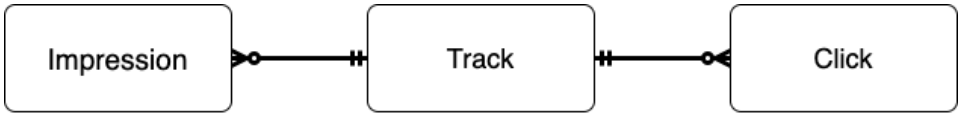

Рисунок 5 – Связь моделей Impression, Track и Click

Все три модели имеют схожую структуру полей, используемых для расчета статистики по различным срезам:

— идентификатор трека;
— идентификатор рекламного объявления;

— идентификатор элемента рекламного инвентаря;
— идентификатор стратегии таргетинга;
— время создания.

Модель Track также дополнительно хранит обезличенные пользовательские данные, отправляемые клиентским приложением во время запроса рекламы: тип и версия операционной системы, страна, язык, часовой пояс, версия клиентского приложения и др. Они используются в стратегиях таргетинга для увеличения вероятности клика на объявление.

2.4. Слой доступа к данным

Слой доступа к данным (англ. data access layer) [6] взаимодействует только с вышестоящим слоем бизнес-логики и состоит из репозиториев. Репозиторий (англ. repository pattern) [24] представляет абстракцию для доступа к данным так, что приложение может работать с данными с помощью интерфейса, который приближен к интерфейсу коллекции. Каждая модель обладает собственным репозиторием.

Добавление, удаление, обновление и получение элементов из этой коллекции выполняется с помощью ряда простых методов без необходимости решать проблемы с базами данных, такие как соединения, команды, курсоры. Использование этого шаблона позволяет добиться слабой связности и отсутствия зависимости от способа хранения данных. Таким образом вся логика необходимая для функционирования моделей содержится в репозиториях.

Исходя из представленных моделей 2.3 и требований к рекламному серверу, репозитории делятся на три группы:

— репозиторий модели рекламного объявления ($AdRepository$, $AdMediaRepository$, $PriceRepository$);
— репозитории моделей таргетинга ($StrategyStateRepository$, $SessionRepository$);
— репозитории моделей трекинга ($TrackRepository$, $ImpressionRepository$, $ClickRepository$).

Также слой доступа к данным состоит из кэша рекламных объявлений (AdCache). Он необходим для сокращения времени ответа на запросы рекламы, а также для снижения количества обращений к базе данных.

2.5. Слой бизнес-логики

Слой бизнес-логики взаимодействует с низлежащим слоем доступа к данным и с вышестоящим слоем представления. Состоит из модулей, которые должны быть слабо связаны между собой для того, чтобы внесение изменений в один или несколько модулей не приводило к изменениям в коде всех модулей слоя.

Модуль — это высокоуровневый компонент системы, взаимодействующий с другими модулями, а также с репозиториями . При разделении бизнес-логики на модули используется подход SOLID (п. 1.1.13), который обеспечивает высокую гибкость, изменяемость и поддерживаемость кода.

Далее рассмотрим каждый модуль по отедельности с указанием его назначения, алгоритма работы и диаграмм для наглядности.

2.5.1. Обновление объявлений

Обновление рекламных объявлений является автоматическим и в контексте данной работы выполняет функцию рекламодателя, т.к. генерирует объявления.

Рисунок 6 – Схема процесса обновления объявлений

Стадии данного процеса описываются следующим алгоритмом:

а) отбор актуальных товаров на Amazon по заранее заданным критериям;
б) генерация рекламных объявлений на основе полученных товаров;
в) сохранение полученных объявлений в базе данных;
г) деактивация уже существующих в системе объявлений и активация только что созданных;
д) обновление объявлений в кэше.

При обновлении объявлений в основной базе данных необходимо также обновить их и в кэше, т.к. кэш не является долгосрочным хранилищем, а лишь ускоряет доступ к актуальной версии объявлений. Выполнение обновления объявлений происходит по расписанию, установленному для соблюдения требуемого уровня актуальности.

2.5.2. Обновление категорий интереса пользователя

У каждого пользователя есть категории, которые его интересуют более остальных. Выделив подобные категории можно показывать объявления, наиболее релевантные интересам пользователя. Данный подход позволит увеличить средний CTR, что является позитивным фактором как для пользователя, который видит только интересующую его рекламу, так и для рекламодателя с издателем, которые извлекают дополнительную прибыль.

Выделение категорий интереса пользователя происходит на основе анализа посещенных веб-страниц. Каждой странице на основе алгоритмов машинного обучения сопоставляется список категорий, наиболее релевантных данной странице. Для каждого пользователя хранится вектор пар категорий и соответствующее количество посещений веб-страниц по данной категории за 30 дней. Схожим образом для каждого пользователя подсчитывается еще один вектор пар категорий рекламных объявлений и количество переходов по объявлениям данной категории за 30 дней.

Отсортировав каждый из двух полученных векторов по невозрастанию количества посещений/переходов и выбрав первые 5 категорий в каждом, получим два профиля краткосрочных интересов пользователя, которые теперь можно использовать в таргетинге.

2.5.3. Таргетинг

Ранжирование — механизм подбора рекламного объявления для показа пользователю таким образом, чтобы вероятность перехода по нему была наиболее высокой.

До тех пор пока данные о совершенных пользователями покупках недоступны, ранжирование производится по показателю CTR вместо более эффективного CPA.

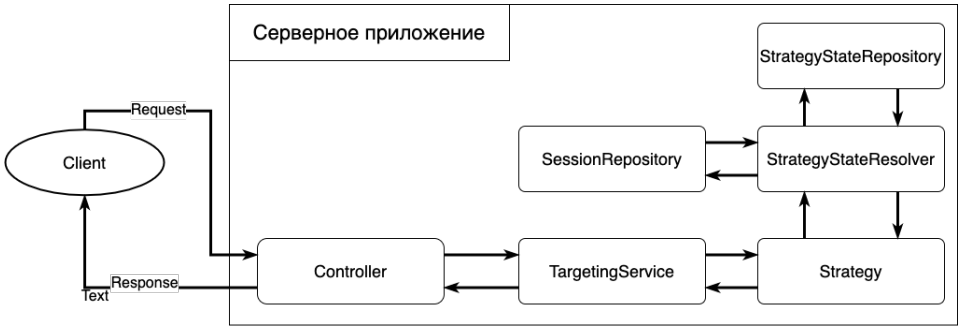

Рисунок 7 – Схема процесса таргетинга

В рамках данной работы алгоритм таргетинга состоит из следующих этапов:

— фильтрация всех доступных объявлений на основе параметров запроса и профиля пользователя;
— выбор для каждого товара одного объявления с наивысшим CTR;
— ранжирование объявлений по невозрастанию CTR;
— получение одного или нескольких объявлений из итогового списка.

Стратегия таргетинга — реализация вышепреведенного алгоритма с применением конкретных подходов для фильтрации, ранжирования и выбора рекламных объявлений. Для каждой стратегии формируется свой набор отранжированных списков объявлений (состояний стратегии) и сохраняется в кэш (репозиторий состояний стратегии), чтобы ускорить процесс получения объявлений. Выбор того или иного списка при запросе объявлений определяется логикой стратегии, а также параметрами пользователя, запрашивающего объявления.

Каждая стратегия таргетирования назначает веса рекламным объявлениям [5], чтобы управлять их частотой показа. Рандомизация применяется для равномерного выбора объявлений.

Выбор оптимальной стратегии на основе данных, представленных в запросе. Если рекламное объявление должно зависеть от контекста, в котором оно будет показано, выбирается одна из стратегий, которая таргетирует на основании параметров переданного контекста. Иначе выбирается стратегия, которая таргетирует на основании накопленных данных о категориях интереса текущего пользователя.

2.5.4. Состояния стратегий

Ранее было преведено определение понятию ”состояние стратегии”(п. 2.3.3). Далее будет рассмотрена концепция разделения состояний стратегий на ”потоки”.

Поток (англ. thread) — совокупность состояний стратегии, объединенных общим признаком (фильтром) и отсортированных в хронологическом порядке. Иными словами, поток представляет последовательность слепков (англ. snapshot) ранжирований объявлений, объединенных общим признаком. Идентификатор потока является строкой. Такой подход является компромиссом между ресурсоемким ранжированием объявлений при каждом запросе объяв-

лений и устареванием показателей каждого объявления, зависящих от количества просмотров и кликов.

Каждое новое ранжирование обяъвлений для конкретного потока, становится состоянием стратегии последней версии и обновляет индекс последней версии данного потока для того, чтобы быстро получать доступ к последней версии в кэше. Каждое состояние получает адрес в кэше (ключ состояния стратегии) согласно идентификатору стратегии, потоку и номеру версии состояния. По этому адресу состояние сохраняется и запрашивается. Состояние не может быть изменено.

Основной поток ($general$) каждой стратегии содержит в себе все возможные объявления, доступные для данной стратегии. Также существуют потоки для категорий объявлений и для пользователей. Они позволяют ранжировать объявления в рамках одной категории или ранжировать только те объявления, которые потенциально будут интересны конкретно взятому пользователю. Это позволяет избегать показов объявлений неинтересных пользователям.

При запросе рекламных объявлений используется существующая или создается новая сессия. Как уже упоминалось ранее (п. 2.3.4), сессия хранит в себе ключ состояния стратегии для доступа к одному и тому же состоянию стратегии при запросах в рамках одной сессии, что позволяет сохранить консистентность при пагинации списка объявлений. Связь сессии с состоянием стратегии реализована при помощи структурного шаблона проектирование $Flyweight$ для уменьшения потребления оперативной памяти при хранении большого количества сессий. За сопоставление состояния стратегии каждой новой сессии отвечает компонент $StrategyStateResolver$.

2.5.5. Стратегии таргетинга

В данной работе рассматриваются 4 стратегии таргетинга объявлений. Каждая из рассматриваемых далее стратегий гарантирует:

— распределение частот показа объявлений пропорционально распределению их CTR;
— консистентность при запросе списка объявлений с пагинацией в рамках одной сессии.

Для выполенения этих требования у каждой стратегии (наследники класса StatefulStrategy) имеются:

— метод $getState(stateKey, offset, limit)$ — для получения из состояния с ключем $stateKey$ списка объявлений длины limit, начиная с позиции offset;
— метод $updateState()$ — для генерации нового списка отранжированных объявлений.

Реализации стратегии по сути сводится к реализации этих двух методов.

Стратегия, возвращающая случайные объявления, разработана первой по счету и использована для ускоренного запуска первой версии рекламного сервера. Логика данной стратегии тривиальна: метод $updateState$ генерирует новое состояние стратегии в основном потоке с ранжированием в случайном порядке, которое затем используется методом $getState$.

Стратегия, возвращающая отранжированные по CTR объявления, описывается следующим образом:

— в данной стратегии используется только основной поток;
— метод $updateState$ генерирует новое состояние стратегии в основном потоке с ранжированием всех объявлений по невозрастанию CTR.

Последующие две стратегии используют множество потоков: для каждой категории объявлений свой поток. Обе эти стратегии возвращают отранжированные по CTR объявления в рамках заданной категории и описываются следующим образом:

— у объявлений каждой категории свой отдельный поток, содержащий состояния стратегии, рассчитанные только на основе объявлений данной категории;
— метод $updateState$ генерирует новое состояние стратегии для каждой категории в соответствующем потоке с ранжированием объявлений данной категории по невозврастанию CTR;
— ключ состояния стратегии, передаваемый в метод $getState$, формируется на основе случайного выбора одной из пяти категорий интереса пользователя.

Отличаются они лишь методом определения категорий интереса пользователя. В одной из стратегий категории интереса пользователя определяются на основе просмотренных веб-страниц при помощи сервиса IBM Watson Natural Language Understanding [18]:

— каждые 24 часа агрегируется список посещенных пользователем веб-страниц, заданных с помощью URL;
— на основе каждого URL из списка формируется запрос к IBM Watson API;
— из ответа на каждый запрос извлекается список категорий, отсортированных по невозрастанию релевантности данной веб-странице;
— каждая категория из списка сопоставляется категории товаров на Amazon;
— для каждой полученной категории увеличивается соответствующий счетчик числа посещений пользователем веб-страниц этой категории;
— счетчики числа посещений категории хранят данные за последние 30 дней.

В второй стратегии категории интереса пользователя определяются на основе переходов по рекламным объявлениям:

— каждые 24 часа агрегируется список посещенных пользователем рекламных объявлений;
— каждое объявление имеет единственную категорию, соответствующую категории товара;
— увеличивается счетчик числа посещений пользователем товаров данной категории;
— счетчики числа посещений категории хранят данные за последние 30 дней.

2.5.6. Трекинг показов и переходов

Трекинг показов и переходов сводится к фиксации изменений стадий жизненного цикла показа объявления пользователю. Жизненный цикл показа объявления состоит из следующих стадий:

— запрос объявления для показа — событие Track фиксируется сервером;
— объявление просмотрено пользователем — событие Impression фиксируется клиентом и отправляется на сервер;
— пользователь совершил переход по объявлению — событие Click фиксируется сервером в момент перехода с помощью редиректа.

Последовательные изменения стадий жизненного цикла, формируют закрытую конверсионную воронку (п. 1.1.7).

Подобные воронки используются для анализа эффективности рекламных объявлений. По данным стадиям воронки можно выделить следующие показатели эффективности:

— Open rate (OR) — отношение числа событий Impression к числу событий Track;
— Click-through rate (CTR) — отношение числа событий Click к числу событий Impression.

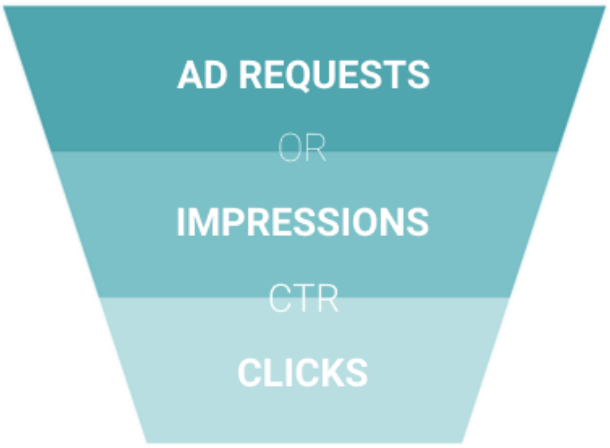

Рисунок 8 – Закрытая конверсионная воронка

Данные показатели рассчитываются как в статистике рекламных объявлений для учета при ранжировании в статегиях таргетинга, так и в статистике стратегий таргетинга для оценки эффективности работы стратегии.

Выводы по главе 2

В ходе второй главы рассмотрен формат клиент-серверного взаимодействия, приведена архитектура серверного приложения, перечислены основные модели и компоненты бизнес-логики, а также описан процесс их работы. Рассмотрен алгоритм таргетинга объявлений и 4 его реализации в виде стратегий таргетинга с описанием хранения и обновления их состояний. В заключении описан процесс трекинга событий для расчета статистики рекламных объявлений и статистики стратегий таргетинга.

ГЛАВА 3. ТЕХНИЧЕСКАЯ РЕАЛИЗАЦИЯ И ВНЕДРЕНИЕ ПРЕДЛОЖЕННОЙ АРХИТЕКТУРЫ

В данной главе подробно рассматриваются детали технической реализации рекламного сервера, а также описание необходимых для её осуществления процессов разработки, тестирования и развертывания.

Текущая инфраструктура компании представлена несколькими Kubernetes (п. 1.1.15) кластерами (по одному на каждое окружение). Серверные приложения являются микросервисами (п. 1.1.9), которые развертываются в кластере и общаются между собой по протоколу AMQP. Разделение всей серверной бизнес-логики на микросервисы производится по функциональному принципу.

Для реализации рекламного сервера также выбрана микросервисная архитектура, которая позволяет обеспечить автономность, хорошую масштабируемость и слабую связность компонент рекламного сервера с компонентами других микросервисов.

В качестве программной платформы, исполняющей исходный код сервера, выбрана платформа Node.js (п. 1.1.11). Она предоставляет необходимый функционал для реализации стандартных возможностей веб-сервера, для взаимодействия с устройствами ввода-вывода (работа с файловой системой, сетевые запросы), для подключения сторонних модулей с помощью NPM (п. 1.1.11) и многое другое. Это позволяет реализовать большую часть функционала посредством стандартных или сторонних модулей.

В качестве основного языка программирования используется TypeScript (п. 1.1.12). Основные преимущества данного языка:

— широко используется в различных проектах компании, что уменьшает порог вхождения членов команды;
— современный язык, разработанный и поддерживаемый компанией Microsoft — гарантия надежности и развития языка;
— предоставляет возможность написания кода как в императивном стиле, так и декларативном, что позволяет повысить читаемость кода и сократить его объем за счет использования подходов из функционального программирования;
— обладает строгой типизацией — существенное сокращение вероятности появления дефектов в процессе разработки и сопровождения кода.

Весь исходный код серверного приложения представляется в виде пакета NPM. Основные параметры пакета (название, версия, ссылка на репозиторий и прочие), скрипты и подключенные сторонние модули подробно описываются в конфигурационном файле пакета — $package.json$. Скрипты представляют из себя команды для проверки синтаксиса, сборки, тестирования и запуска приложения.

В качестве решения для контейнеризации среды исполнения кода приложения используется Docker. Расположенный в корневой директории проекта файл Dockerfile описывает процесс создания образа, который впоследствии будет отправлен в облачный реестр образов и использован для развертывания микросервиса в Kubernetes кластере.

Развернутый в кластере микросервис взаимодействует с другими микросервисами, обмениваясь сообщениями по протоколу AMQP, и предоставляет API (п. 1.1.8) для получения внешних запросов по протоколу HTTP. Для обработки поступающих запросов используется Node.js фреймворк Koa [15].

В качестве DNS серверов используется сервис Cloudflare, который также фильтрует подозрительный трафик и кеширует статические файлы такие как, например, изображения и видео, используемые в рекламе.

В следующих параграфах более детально описывается устройство моделей БД, детали реализации отдельных модулей бизнес-логики, устройство методов API и алгоритм развертывания в Kubernetes кластере.

3.1. Работа с моделями

Каждая модель представлена в базе данных одной или несколькими сущностями в зависимости от сложности структуры модели. Каждой сущности соответствует репозиторий, предоставляющий методы для их создания, получения, изменения и удаления из БД. Далее подробно рассматриваются модели и соответствующие им сущности.

3.1.1. Модели рекламного объявления

Для хранения товаров, полученных с помощью Amazon Product Advertising API, используется сущность $AmazonItem$. Модель Ad представляет рекламное объявление и состоит из 3 сущностей: Ad, $AdMedia$, $Price$. Модель разделяется на сущности для удобства представления отношений ”один-к-одному”и ”один-ко-многим”:

— одной сущности $AmazonItem$ соответствует одна сущность Price;
— одной сущности $Price$ соответствует много сущностей Ad;
— одной сущности Ad соответствует много сущностей $AdMedia$.

Для хранения вышеперечесиленных сущностей используется NoSQL база данных MongoDB (п. 1.1.14). Выбор произведен в пользу данной базы, поскольку:

— MongoDB — масштабируемая база данных общего назначения;
— хорошо сочетается с экосистемой Node.js;
— в терминах теоремы CAP [2] обеспечивает строгую консистентность данных и позволяет разделять их на шарды;
— позволяет достаточно быстро и просто расширять схему данных без миграции всех данных в коллекции;
— является основной БД компании.

Каждая сущность имеет собственный репозиторий, который реализует функционал создания, получения, изменения и удаления сущностей в соответствующей коллекции MongoDB. Перечисленный функционал репозиториев реализуется при помощи стандартного MongoDB драйвера для Node.js [17].

3.1.2. Модели таргетинга

Для обеспечения работы таргетинга необходимы модели:

— $Session$ — для хранения состояния сессии пользователя;
— $StrategyState$ — для хранения состояния стратегии.

Также для каждого пользователя хранятся такие модели, как

— $PageViewsProfile$ — вектор количества посещений сайтов каждой из 316 категорий второго уровня [12];
— $AdClicksProfile$ — вектор количества кликов по товарам каждой из 50 допустимых категорий товаров Amazon (Приложение Б).

Оба профиля также содержат поля $userId$ для связи профиля с пользователем и $createdAt$ — время создания профиля.

Исходя из частоты и характера сценариев использования этих моделей (частые чтения, редкие записи), необходима максимальная скорость получения их из БД. Для решения такой задачи идеально подходит база данных Redis (п. 1.1.14), поскольку:

— Redis — это нереляционная высокопроизводительная СУБД;
— подходит для управлением состояний сессий [25];

— позволяет эффективно кэшировать данные [11];
— в случае существенного преобладания количества чтений данных над количеством записей показывает себя более эффективным инструментом, чем MongoDB [23].

Каждая сущность имеет собственный репозиторий, который реализует функционал создания, получения, этих моделей. Данный функционал репозиториев реализуется при помощи модуля $ioredis$ [13].

3.1.3. Модели трекинга и счетчиков категорий

В трекинге участвуют такие модели, как $Track$, $Impression$, $Click$. Они соответствуют событиям, фиксируемым в процессе показа рекламных объявлений.

Указанные модели являются ключевыми данными, на основе которых рассчитывается статистика рекламных объявлений и стратегий таргетинга, по которым производится анализ эффективности объявлений и рекламного сервера в целом. Примером подобной технологии является OLAP (онлайн обработка аналитических запросов) — технология обработки данных, заключающаяся в подготовке суммарной (агрегированной) информации на основе больших массивов данных, структурированных по многомерному принципу.

В качестве хранилища для вышеперечисленных моделей выбрана база данных ClickHouse (п. 1.1.14), поскольку:
— ClickHouse — СУБД для OLAP;
— ClickHouse является колоночной базой данных, которая отлично подходит для хранения структур представленных векторами: векторы категорий веб-страниц и товаров, профили пользователей.

Счетчики категорий представлены векторами состоящими из 0 и 1. Для категорий веб-страниц вектор имеет размерность 316, для категорий товаров — 50. Вектора каждого из двух типов хранятся в соответствующих таблицах семейства MergeTree базы данных ClickHouse.

Для создания и получения этих моделей в БД также используются репозитории на основе драйвера для Node.js [28]. Каждому элементу вектора соответствует собственная колонка в таблице базы данных.

3.2. Обновление объявлений

Класс $AdUpdater$ отвечает за периодическое автоматическое обновление рекламных объявлений. Обновление объявлений описывается следующим алгоритмом:

— получить товары Amazon согласно установленным параметрам фильтрации: товар доступен для приобретения в США, наличие скидки, категория товара входит в список допустимых (Приложение Б);
— генерация рекламных объявлений на основе полученных товаров Amazon;
— обновление списка активных рекламных объявлений;
— обновление рекламных объявлений в кэше.

3.2.1. Получение товаров Amazon

На данный момент официальным способом доступа к товарам, размещаемым на площадке Amazon является интерфейс Product Advertising API 5.0 [20]. Доступ к API осуществляется с помощью $accesstoken$, который выдается после успешного прохождения авторизации на площадке. Взаимодействие происходит по HTTP протоколу, формат запросов и ответов — JSON. Для осуществления авторизованных запросов используется Node.js SDK [21].

В ссылку на товар подставляется $PartnerTag$ для связывания покупок, совершаемых по данной ссылке, с партнером, который привел покупателя.

3.2.2. Создание объявления

Класс $AdService$ отвечает за получение и обновление текущих объявлений и деактивацию неактуальных. $AdService$ имеет в качестве зависимостей репозитории $AdRepository$, $AdMediaRepository$, $PriceRepository$ для ощуствления операций над внутренними сущностями модели Ad.

Класс $AdUpdater$ на основе полученного списка товаров с Amazon генерирует список моделей Ad, которые затем передаются в метод $update$ класса $AdService$. Данный метод в свою очередь разделяет экземпляр модели Ad на сущности Ad, $AdMedia$, $Price$ и передает их в методы репозиториев, которые создают или обновляют сущности в MongoDB.

При создании нового объявления все связанные с ним медиаресурсы сохраняются в Google Bucket, который также предоставляет функции CDN. Медиаресурсы уникализируются по ссылке для того, чтобы не создавать копии уже сущетвующих файлов в Google Bucket.

При создании или обновлении объявлений необходимо также обновить их данные и в кэше. За это отвечает класс $AdCache$, который использует Redis для кэша и стандартный модуль $classtransformer$ для автоматической сериализации и десериализации модели Ad.

3.3. Обновление категорий интереса пользователя

Каждому просмотру веб-страницы пользователем сопоставляется вектор размера 316 (количество категорий веб-страниц), состоящий из 0 и 1, где 1 на позиции категории означает, что веб-страницу можно отнести к этой категории, 0 — иначе. Список категорий, соответствующих веб-странице, получается путем HTTP запроса к IBM Watson Natural Language Understanding API. Список категорий, полученный в результате запроса, трансформируется в вектор по вышеупомянутому правилу.

Аналогичным образом каждому клику по рекламному объявлению сопоставляется вектор длины 50 (количество категорий товаров), состоящий из 0 и 1, где 1 на позиции категории означает, что товар относится к этой категории, в противном случае — 0.

Запускаемый раз в сутки модуль $ProfileUpdater$ для каждого пользователя рассчитывает 2 профиля:

— $PageViewsProfile$ — на основе векторов категорий веб-страниц, посещенных за последние 30 дней;

— $AdClicksProfile$ — на основе векторов категорий товаров Amazon, просмотренных за последние 30 дней.

Оба профиля рассчитываются группировкой векторов по $userId$, которая производится посредством базы данных ClickHouse. Колонки категорий при группировке суммируются.

Рассчитанные профили в формате JSON сохраняются в $hash$ Redis. Они используются в соответствующих стратегиях таргетинга для выбора состояний стратегий, которые содержат наиболее релевантные текущим интересам пользователя объявления. Новые профили замещают имеющиеся в Redis при сохранении.

3.4. Обновление состояний стратегий

Для обновления состояний стратегий существует класс $StrategyStateUpdater$. Он запускается раз в сутки по расписанию и вызывает метод $updateState$ каждой из 4-х стратегий таргетинга. Вызов данного

метода генериует новое состояние стратегии для каждого потока, который она использует. Как уже упоминалось ранее, состояние стратегии содержит в себе отранжированный список некоторого подмножества рекламных объявлений, объединенных общим признаком (категория товара, релевантность для пользователя).

Метод $updateState$ использует репозиторий $StrategyStateRepository$ для сохранения нового состояния в Redis. Время жизни каждого состояния — 1 сутки, после истечения данного срока Redis автоматически их удаляет. Периодическое сохранение актуального состояния с одной стороны позволяет поддерживать актуальность ранжирования объявлений, а с другой стороны избавляет от неоправданного использования серверных ресурсов, в случае если бы ранжирование производилось при каждом запросе.

Детали реализации метода $updateState$ зависят от логики конкретной стратегии. Стратегия, ранжирующая объявления на основе CTR, при обновлении своего состояния раз в 15 минут запрашивает актуальную статистику объявлений с помощью класса $TrackService$. Стратегии, учитывающие при ранжировании категории интереса пользователей, получают профиль категорий интереса при помощи класса $ProfileService$.

Метод $getAdStats$ класса $TrackService$ производит группировку просмотров и кликов по $adId$ для всех активных объявлений. Результат вызова данного метода используется при обновлении состояния стратегии, ранжирующей по CTR.

3.5. Методы API

Контроллеры, соответствующие методам API, спроектированы так, чтобы содержать минимальное количество бизнес-логики. Такой подход обеспечивает слабую связность контроллеров и бизнес-логики, что упрощает процесс их тестирования.

В контексте платформы Node.js процесс обработки запросов является примером поведенческого шаблона проектирования Chain of Responsibility. Для каждого запроса последовательно выполняются все обработчики цепи (middlewares). Реализация метода API сводится к созданию обработчика и помещению его в цепочку вывозов, соответствующую этому методу API.

Для аутентификации запросов к API и передачи данных сессии используется стандарт JSON Web Token (JWT) [14]. Для автоматической генерации

документации методов API используется OpenAPI Specification (OAS) [26] — спецификация, определяющая стандартное, независимое от языка программирования описание для API. Файл openapi.yml, расположенный в корне проекта, содержит описание методов API согласно данной спецификации. Для доступа к сгенерированной документации неоьходимо сделать HTTP запрос GET/doc.

3.5.1. Методы для получения объявлений

Класс $TargetingService$ предназначен для получения объявлений с наиболее высокой вероятностью перехода по ним. В зависимости от параметров пользователя и контекста запроса выбирается наиболее подходящая стратегия таргетинга. Класс имеет метод $getNextAd$ для получения одного объявления, и метод $getAdsRange$ для получения части списка длины $limit$, начиная с позиции $offset$.

Для получения рекламных объявлений в API предусмотрено 2 метода:

— $POST/get\_ad$ — вызывает метод $getNextAd$ с параметрами запроса (страна, язык, операционная система, идентификатор пользователя, ключ сессии и пр.), возвращает одно объявление;

— $POST/get\_ads$ — вызывает метод $getAdsRange$ с параметрами запроса, возвращает список объявлений с пагинацией, который используется как лента товаров (товары из топ-5 категорий интереса пользователя) в приложении. Для того, чтобы при пагинации товары в ленте не повторялись, этот список формируется заранее на стороне сервера.

Ссылки в возвращаемых объявлениях ”оборачиваются”с помощью класса $RedirectEncoder$ для отслеживания переходов.

3.5.2. Методы для учета просмотра и перехода по ссылке объявления

Класс $TrackService$ отвечает за фиксирование событий запроса рекламы (метод $createTrack$), просмотра (метод $trackImpression$) и перехода по ссылке в объявлении (метод $trackClick$).

Фиксирование просмотра объявления просходит с помощью метода $POST/track$, в его теле запроса передаются:

— $trackId$ — идентификатор трека;

— $timestamp$ — время совершения просмотра объявления на клиенте.

Далее эти параметры передаются в метод $trackImpression$ класса $TrackService$.

Фиксирование перехода по ссылке в объявлении просходит с помощью метода $GET/redirect$ Все ссылки в объявлениях ”оборачиваются”вызовом этого метода, что позволяет отследить переход и передать его параметры (url, $trackId$) в метод $trackClick$ для сохранения в базе данных. Далее происходит перенаправление на URL указанный в параметрах запроса.

3.5.3. Метод для получения статистики по стратегиям таргетинга

Метод $GET/analytics$ использует метод $getAnalytics$ класса $TargetingService$ для получения статистики по количеству запросов объявлений, просмотров и переходов по объявлениям с разбивкой по стратегиям таргетинга за указанный интервал дат (интервал задается параметрами $from$ и to). Также теле ответа этого метода присутствуют показатели OR и CTR по каждой стратегии.

Метод $getAnalytics$ класса $TargetingService$ рассчитывает все вышеперечисленные параметры группировкой треков, просмотров и кликов по $strategyId$. Группировка производится базой данных ClickHouse.

3.6. Развертывание

Для локального развертывания рекламного сервера используется $docker\text{-}compose$, по умолчанию выполняющий алгоритм развертывания, прописанный в файле docker-compose.yml в корне проекта.

Для полноценного развертывания серверного приложения используется кластер в Google Kubernetes Engine. Для автоматизированного управления ресурсами кластера используется Terraform. Приложение может быть развернуто в одном из двух окружений staging или production, которые являются похожими с точностью до количества выделенных вычислительных ресурсов и доменных имен.

Production окружение является основным окружением, в котором разворачиваются серверные приложения, обрабатывающие реальные запросы пользователей. Staging окружение сконструировано так, чтобы быть максимально похожим на production и используется для финального этапа тестирования серверных приложений. Исходный код организован таким образом, чтобы при старте приложения все параметры необходимого окружения передавались в переменных окружения (англ. environment variables) [27].

Для описаний конфигурации таких экземпляров абстакций Kubernetes, как Pod, Service и Deployment используются .yml файлы в папке deploy. Pod —

одна из основных абстракций Kubernetes, позволяющая запускать экземпляры Docker образов в изолированном окружении. Сетевая абстракция $Service$ связывает несколько экзепляров Pod и является входным интерфейсом для них. $Ingress$ позволяет $Service$ принимать запросы из внешней сети и переадресовывать их для обработки в связанные с ним Pod'ы. Также $Ingress$ используется в качестве балансировщика нагрузки.

Процесс развертывания для staging и production окружений описывается в файле bitbucket-pipelines.yml и состоит из следующих стадий:

— сборка текущего кода в master;
— запуск unit тестов;
— сборка новой версии Docker образа и отправка в Google Container Registry;
— вызов утилиты $skaffold$ для развертывания полученного образа.

Для запуска развертывания в staging окружении необходимо сделать коммит в ветку master. Если в коммите присутсвует тег с номером версии развертывание происходит в production окружении.

3.7. Результаты

Рекламный сервис развернут в кластере Google Kubernetes Engine, который представленен несколькими серверами общего назначения типа N1 [16] — $n1highmem4$. Каждый сервер имеет 4 CPU и 24 Гб RAM.

На графиках 9 и 10 представлены использование CPU и RAM двумя репликами рекламного сервера за последние 30 дней.

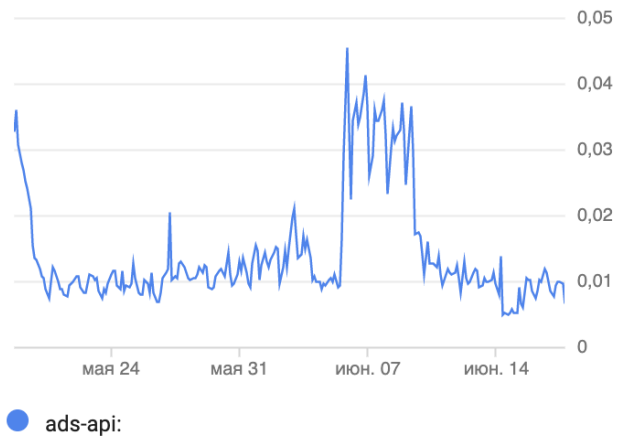

Рисунок 9 – Загрузка ЦП репликами рекламного сервера

Ось ординат на графике 9 выражена в Millicores — метрика Kubernetes, которая используется для измерения загрузки процессора (ядро процессора,

разделенное на 1000 единиц). Исходя из данных графика видно, что процессор слабо задействован в операциях совершаемых рекламным сервером.

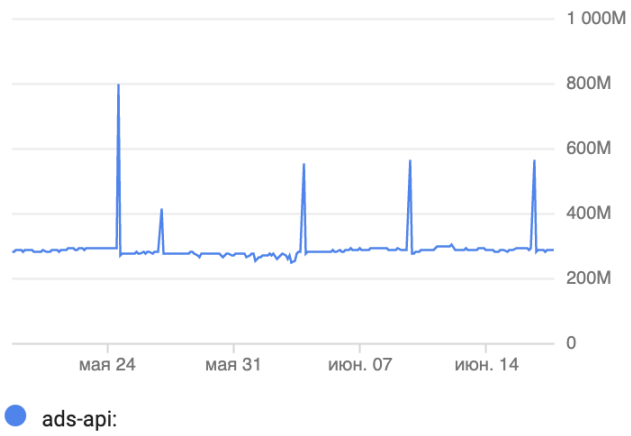

Рисунок 10 – Использование RAM репликами рекламного сервера

График 10 показывает, что каждая реплика использует порядка 150 Мб RAM в среднем. Пиковые значения на графике обусловлены периодическим обновлением образа из репозитория Google Container Registry.

Выводы по главе 3

В ходе третьей главы подробно рассмотрен принцип работы моделей: разбиение моделей на сущности, выбранные для их хранения базы данных. Приведено обоснование выбора той или иной базы данных в каждом конкретном случае. Описаны реализации процессов обновления, кэширования рекламных объявлений и обновления категорий интереса пользователей. Приведено описание методов API и контроллеров, обрабатывающих поступающие запросы. Подробно рассмотрен процесс развертывания рекламного сервера в различных окружениях.

ЗАКЛЮЧЕНИЕ

В ходе проделанной работы достигнута поставленная цель и выполнены все необходимые для этого подзадачи:

- разработан контракт клиент-серверного взаимодействия;
- разработана архитектура серверного приложения;
- разработан модуль для создания и обновления рекламных объявлений на основе товаров Amazon;
- разработан механизм получения списка объявлений с сохранением консистентности;
- разработаны алгоритмы работы 4-х стратегий таргетинга объявлений;
- разработана система трекинга просмотров и переходов по объявлениям с возможностью просмотра статистики.

Рекламный сервер в виде микросервиса успешно развернут в production-окружении и используется клиентами приложения Fulldive Browser на платформах iOS и Android. Результаты работы рекламного сервера в виде изображений доступны в приложении А.

В случае успешных договоренностей с компанией Amazon по поводу предоставления возможности отслеживать совершаемые пользователями покупки, станет возможным оптимизировать таргетинг по показателю CPA, что должно увеличить число конверсий.

Дальнейшие разработки и улучшения направлены на:

- добавление стратегий на основе контекстного таргетинга и улучшение качества работы таргетинга объявлений в целом;
- улучшение алгоритма фильтрации наиболее привлекательных товаров для целевой аудитории приложений компании;
- расширение списка поддерживаемых площадок с товарами: Ebay, AliExpress, Wildberries и другие;
- разработку необходимого функционала для привлечения сторонних рекламодателей;
- оптимизацию количества и качества рекламного инвентаря в приложениях компании.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Build an Ad Server in Weeks | Adzerk [Электронный ресурс]. — 2020. — URL: https://adzerk.com/ (visited on 05/26/2020).

2 CAP theorem [Электронный ресурс]. — 2020. — URL: https://en.wikipedia.org/wiki/CAP_theorem (visited on 05/26/2020).

3 Capterra - find the best Ad Server Software for your business [Электронный ресурс]. — 2020. — URL: https://www.capterra.com/ad-server-software/ (visited on 05/26/2020).

4 Contextual advertising [Электронный ресурс]. — 2020. — URL: https://en.wikipedia.org/wiki/Contextual_advertising (visited on 05/26/2020).

5 Creative sequencing [Электронный ресурс]. — 2020. — URL: https://en.wikipedia.org/wiki/Creative_sequencing (visited on 05/26/2020).

6 Data access layer [Электронный ресурс]. — 2020. — URL: https://en.wikipedia.org/wiki/Data_access_layer (visited on 05/26/2020).

7 Epom Ad Server | Leading Ad Serving Platform for Networks [Электронный ресурс]. — 2020. — URL: https://epom.com/ (visited on 05/26/2020).

8 Free Open Source Ad Server - Revive Adserver [Электронный ресурс]. — 2020. — URL: https://www.revive-adserver.net/ (visited on 05/26/2020).

9 Google Ad Manager [Электронный ресурс]. — 2020. — URL: https://admanager.google.com/home/ (visited on 05/26/2020).

10 How is programmatic marketing and digital display advertising defined [Электронный ресурс]. — 2020. — URL: https://www.smartinsights.com/internet-advertising/internet-advertising-targeting/what-is-programmatic-marketing/ (visited on 05/26/2020).

11 How To: Redis Caching | Redis Labs [Электронный ресурс]. — 2020. — URL: https://redislabs.com/redis-enterprise/use-cases/caching/ (visited on 05/26/2020).

12 IBM Cloud Docs - Categories hierarchy [Электронный ресурс]. — 2020. — URL: https://cloud.ibm.com/docs/natural-language-understanding?topic=natural-language-understanding-categories-hierarchy (visited on 05/26/2020).

13 ioredis - a robust, performance-focused and full-featured Redis client for Node.js. [Электронный ресурс]. — 2020. — URL: https://github.com/luin/ioredis (visited on 05/26/2020).

14 JSON Web Tokens - jwt.io [Электронный ресурс]. — 2020. — URL: https://jwt.io/ (visited on 05/26/2020).

15 Koa - next generation web framework for node.js [Электронный ресурс]. — 2020. — URL: https://koajs.com/ (visited on 05/26/2020).

16 Machine types|Compute Engine Documentation|Google Cloud [Электронный ресурс]. — 2020. — URL: https://cloud.google.com/compute/docs/machine-types (visited on 06/17/2020).

17 MongoDB Node.js Driver [Электронный ресурс]. — 2020. — URL: https://mongodb.github.io/node-mongodb-native/ (visited on 05/26/2020).

18 Natural Language Understanding - IBM Cloud API Docs [Электронный ресурс]. — 2020. — URL: https://cloud.ibm.com/apidocs/natural-language-understanding#categories (visited on 05/26/2020).

19 Powerful app monetization | MoPub [Электронный ресурс]. — 2020. — URL: https://www.mopub.com/en (visited on 05/26/2020).

20 Product Advertising API 5.0 - documentation [Электронный ресурс]. — 2020. — URL: https://webservices.amazon.com/paapi5/documentation/ (visited on 05/26/2020).

21 Product Advertising API 5.0 - Node.js SDK [Электронный ресурс]. — 2020. — URL: https://webservices.amazon.com/paapi5/documentation/assets/archives/paapi5-nodejs-sdk-example.zip (visited on 05/26/2020).

22 Programmatic Buying - Definition and Demarcation [Электронный ресурс]. — 2020. — URL: https://en.ryte.com/wiki/Programmatic_Buying (visited on 05/26/2020).

23 Redis vs. MongoDB: Comparing In-Memory Databases with Percona Memory Engine [Электронный ресурс]. — 2017. — URL: https://scalegrid.io/blog/comparing-in-memory-databases-redis-vs-mongodb-percona-memory-engine/ (visited on 05/26/2020).

24 Repository Pattern [Электронный ресурс]. — 2020. — URL: https://deviq.com/repository-pattern/ (visited on 05/26/2020).

25 Session Management | Redis Labs [Электронный ресурс]. — 2020. — URL: https://redislabs.com/redis-enterprise/use-cases/session-management/ (visited on 05/26/2020).

26 The OpenAPI Specification Repository [Электронный ресурс]. — 2020. — URL: https://github.com/OAI/OpenAPI-Specification (visited on 05/26/2020).

27 The Twelve-Factor App [Электронный ресурс]. — 2020. — URL: https://12factor.net/ (visited on 05/26/2020).

28 Yandex ClickHouse driver for nodejs [Электронный ресурс]. — 2020. — URL: https://github.com/apla/node-clickhouse (visited on 05/26/2020).

ПРИЛОЖЕНИЕ А. РЕКЛАМА В ПРИЛОЖЕНИИ FULLDIVE BROWSER

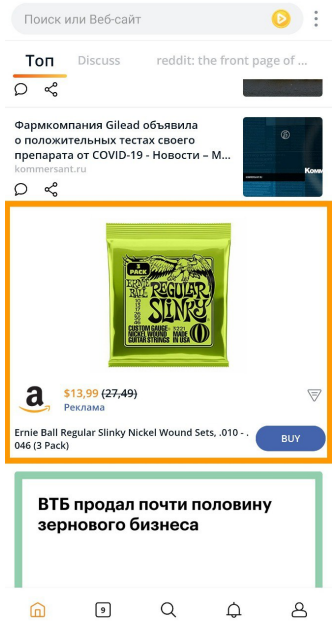

Рисунок А.1 – Пример объявления в основной ленте новостей

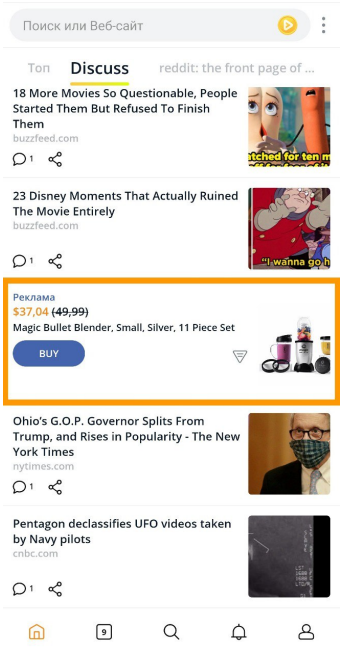

Рисунок А.2 – Пример объявления в ленте обсуждаемых новостей

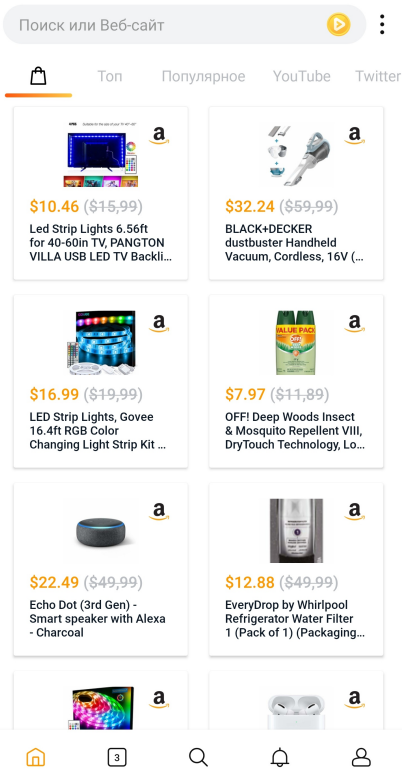

Рисунок А.3 – Пример списка рекламных объявлений на Android

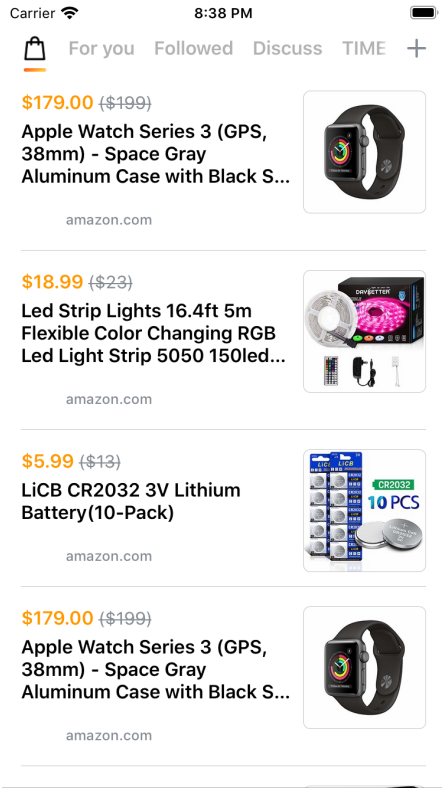

Рисунок А.4 – Пример списка рекламных объявлений на iOS

ПРИЛОЖЕНИЕ Б. КАТЕГОРИИ ТОВАРОВ НА AMAZON

```
const AMAZON_CATEGORIES = [
    'AmazonVideo',
    'Apparel',
    'Appliances',
    'ArtsAndCrafts',
    'Automotive',
    'Baby',
    'Beauty',
    'Books',
    'Classical',
    'Collectibles',
    'Computers',
    'DigitalMusic',
    'DigitalEducationalResources',
    'Electronics',
    'EverythingElse',
    'Fashion',
    'FashionBaby',
    'FashionBoys',
    'FashionGirls',
    'FashionMen',
    'FashionWomen',
    'GardenAndOutdoor',
    'GiftCards',
    'GroceryAndGourmetFood',
    'Handmade',
    'HealthPersonalCare',
    'HomeAndKitchen',
    'Industrial',
    'Jewelry',
    'KindleStore',
    'LocalServices',
    'Luggage',
```

```
    'LuxuryBeauty',
    'Magazines',
    'MobileAndAccessories',
    'MobileApps',
    'MoviesAndTV',
    'Music',
    'MusicalInstruments',
    'OfficeProducts',
    'PetSupplies',
    'Photo',
    'Shoes',
    'Software',
    'SportsAndOutdoors',
    'ToolsAndHomeImprovement',
    'ToysAndGames',
    'VHS',
    'VideoGames',
    'Watches'
]
```

Листинг Б.1 – Список допустимых категорий товаров на Amazon