Оригинальная статья / Original article

УДК 004.94
http://doi.org/10.35854/1998-1627-2023-3-335-344

Модель модуля динамической генерации персональных предложений дополнительных услуг для пассажиров авиакомпаний

Александр Дмитриевич Столяров[1], Владимир Владимирович Гордеев[2], Виктор Иванович Абрамов[3]✉

[1, 3] *Национальный исследовательский ядерный университет МИФИ, Москва, Россия*

[2] *Общество с ограниченной ответственностью «АЭРОЛАБС», Москва, Россия*

[1] *mr.alexst@gmail.com, https://orcid.org/0000-0001-7259-9348*

[2] *v.gordeev@aerolabs.aero, https://orcid.org/0000-0001-8514-4705*

[3] *viabramov@mephi.ru✉, https://orcid.org/0000-0002-9471-9408*

Аннотация

Цель. Сформировать оптимальную модель программного модуля для генерации персональных предложений.

Задачи. Рассмотреть функциональные требования к модулю динамической генерации персональных предложений; проанализировать необходимые интеграции (вводы и выводы); выбрать оптимальную архитектуру модуля на основе поставленных перед ним задач и лучших практик; сформировать модель модуля для заданных типов генерируемых предложений.

Методология. При проведении исследования использованы общенаучные методы (анализ, синтез, монографический, группировка). Изучены лучшие практики построения рекомендательных систем, предназначенных для работы с большими данными с выделением наиболее перспективных подходов с точки зрения дальнейшего развития и потенциала к интеграции.

Результаты. Сформулированы функциональные требования к модулю динамической генерации персональных предложений и выполняемые им задачи. Проведен анализ лучших практик построения рекомендательных систем, предназначенных для работы с большими данными. Определена наиболее перспективная технологическая конфигурация модуля динамической генерации персональных предложений. Сформирована модель модуля генерации персональных предложений.

Выводы. Возможности обработки больших массивов данных позволяют существенно повышать эффективность маркетинговых инструментов, в частности путем обучения программного обеспечения в автоматическом режиме генерировать рекламные предложения дополнительных услуг, выбирая при этом оптимальный канал доставки, тип, класс услуги и этап жизненного цикла клиента. Работа модуля генерации персональных предложений возможна только в составе программного обеспечения, выполняющего ряд расширенных функций (таких как сбор, хранение и обработка данных о клиентах, кластеризация клиентов, сбор обратной связи о совершенных клиентами действиях, обучение на основе действий пользователей и многие другие). Ввиду этого архитектура модуля динамической генерации предложений должна позволять эффективно интегрироваться с программным обеспечением и работать с большими массивами данных, генерируемых авиакомпанией. В процессе исследования сформирована модель модуля генерации персональных предложений, удовлетворяющего описанным выше требованиям.

Ключевые слова: *рекомендательные системы, искусственный интеллект, таргетированная реклама, генерация предложений*

Для цитирования: Столяров А. Д., Гордеев В. В., Абрамов В. И. Модель модуля динамической генерации персональных предложений дополнительных услуг для пассажиров авиакомпаний // *Экономика и управление.* 2023. Т. 29. № 3. С. 335–344. http://doi.org/10.35854/1998-1627-2023-3-335-344

Model of a module for dynamic generation of personalized offers of additional services for airline passengers

Aleksandr D. Stolyarov[1], Vladimir V. Gordeev[2], Victor I. Abramov[3]✉

[1, 3] *National Research Nuclear University "MEPhI", Moscow, Russia*

[2] *Aerolabs LLC, Moscow, Russia*

[1] *mr.alexst@gmail.com, https://orcid.org/0000-0001-7259-9348*

[2] *v.gordeev@aerolabs.aero, https://orcid.org/0000-0001-8514-4705*

[3] *viabramov@mephi.ru✉, https://orcid.org/0000-0002-9471-9408*

Abstract

Aim. The presented study aims to develop an optimal model of a software module for generating personalized offers.

Tasks. The authors investigate the functional requirements of a module for dynamic generation of personalized offers; analyze the necessary integrations (inputs and outputs); choose the optimal architecture for the module based on the tasks assigned to it and best practices; develop a model of a module for the specified types of generated offers.

Methods. This study uses general scientific methods (analysis, synthesis, monographic method, grouping). Best practices of building recommendation systems designed to work with big data are investigated, and the most promising approaches in terms of further development and potential for integration are highlighted.

Results. The functional requirements of a module for dynamic generation of personalized offers and the tasks performed by it are formulated. Best practices of building recommendation systems designed to work with big data are analyzed. The most promising technological configuration of a module for dynamic generation of personalized offers is determined. A model of a module for generating personal offers is developed.

Conclusions. The capabilities of processing large amounts of data make it possible to significantly increase the effectiveness of marketing tools, particularly by training software to automatically generate advertising offers of additional services, choosing the optimal delivery channel, type, class of service, and stage of the customer lifecycle. A module for generating personalized offers can work only as part of software that performs several advanced functions (such as collecting, storing and processing customer data, clustering customers, collecting feedback on customer actions, training based on user actions, and many others). Therefore, the architecture of a dynamic offer generation module should be designed for effective integration with software and work with large amounts of data generated by the airline. During the study, a model of a module for generating personalized offers satisfying the requirements described above is developed.

Keywords: *recommendation systems, artificial intelligence, targeted advertising, offer generation*

For citation: Stolyarov A.D., Gordeev V.V., Abramov V.I. Model of a module for dynamic generation of personalized offers of additional services for airline passengers. *Ekonomika i upravlenie = Economics and Management.* 2023;29(3):335-344. (In Russ.). http://doi.org/10.35854/1998-1627-2023-3-335-344

Новая реальность нашего мира — четвертая промышленная революция и начало становления шестого технологического уклада. В современной экономической среде, которая характеризуется как *BANI*-мир (акроним от английских слов *brittle, anxious, non-linear, incomprehensible*, в переводе на русский язык означающих «хрупкий, тревожный, нелинейный, непонятный»), многие компании уже больше не смогут добиться успеха, лишь адаптируя методы управления, которые обеспечивали успех в прошлом. Задача цифровой трансформации экономики и повышения темпов экономического развития страны актуальна как никогда, поэтому требуются иные подходы к управлению с использованием инновационных цифровых технологий, обеспечивающих новые способы наращивания эффективности работы предприятий.

Важным фактором является использование предиктивной аналитики, объединяющей в себе множество методов машинного обучения, статистики, моделирования, теории игр, на основе которых анализируют исторические данные, составляют предска-

Рис. 1. Взаимодействие модуля динамической генерации персональных предложений с иными модулями разрабатываемого программного обеспечения

Fig. 1. Interaction between the module for dynamic generation of personalized offers with the other modules of the developed software

зания будущего состояния объекта, выявляют закономерности и определяют риски и возможности. Преимущества интеграции предиктивной аналитики в жизненный цикл организации заключаются в использовании интеллектуальной аналитики для принятия оптимальных решений, минимизации неопределенности, точного управления рисками и своевременного реагирования на изменения различных типов показателей эффективности бизнеса [1].

Темпы изменений конъюнктуры большинства рынков под влиянием различных обстоятельств неуклонно возрастают. В этих динамичных условиях повышается значимость умения адаптироваться в актуальной ситуации. Клиентоцентричность становится общепринятой стратегией, обеспечивающей выживание на конкурентных рынках, если соперничество компаний происходит в аспекте удобства и эмоциональности взаимодействия. При этом техническим средством, позволяющим управлять взаимоотношениями с клиентами, являются *CRM*-системы [2]. Управление отношениями с клиентами — это бизнес-стратегия, направленная на повышение рентабельности, дохода и удовлетворенности клиентов. Для достижения эффективных отношений с клиентами компания располагает широким спектром инструментов, методов и процедур, помогающих развивать отношения с клиентами для увеличения продаж [3].

Модуль динамической генерации персональных предложений предполагает функционирование в составе программного обеспечения рекомендательной системы, выполняющей сбор, обработку, структурирование и обогащение данных о клиентах авиакомпании; кластеризацию клиентов авиакомпании, расчет персональных предложений для клиентов, динамическую генерацию персональных предложений, доставку персональных предложений, обучение программного обеспечения на основе совершаемых пользователями действий. Результатом работы модуля динамической генерации персональных предложений служит конкретное предложение, готовое для доставки по требуемому каналу конечному пользователю. Таким образом, модуль динамической генерации персональных предложений выступает в качестве связки для модуля расчета рекомендаций сервисов дополнительных услуг и модуля доставки сгенерированных предложений, как видно на рисунке 1.

Главная задача модуля динамической генерации персональных предложений — формирование удобного для восприятия человеком рекламного объявления, адаптированного под тот или иной канал коммуникации, и содержащего информацию, подобранную с учетом решения модуля рекомендаций сервисов дополнительных услуг. Общий алгоритм работы модуля представлен на рисунке 2.

Для повышения скорости работы системы и обеспечения ее автономности все ресурсы, используемые при генерации персональных предложений, хранятся во внутренней базе данных. Тем самым при генерации объявлений исключены потенциальные задержки, связанные с запросами по *API* и загрузкой данных с серверов партнеров. Такие задержки могут приводить к тому, что баннер не будет сформирован либо будет сформирован частично. Претензии у клиента будут не к партнеру, а к нашему программному обеспечению. Для исключения подобных ситуаций все данные, необходимые для объявлений, как запрашиваемые по *API*, так и генерируемые внутри системы, решено хранить в собственной базе данных.

Задачей модуля служит формирование «на лету» оптимального предложения необходимого формата: при определенных действиях пользователя (просмотр страницы,

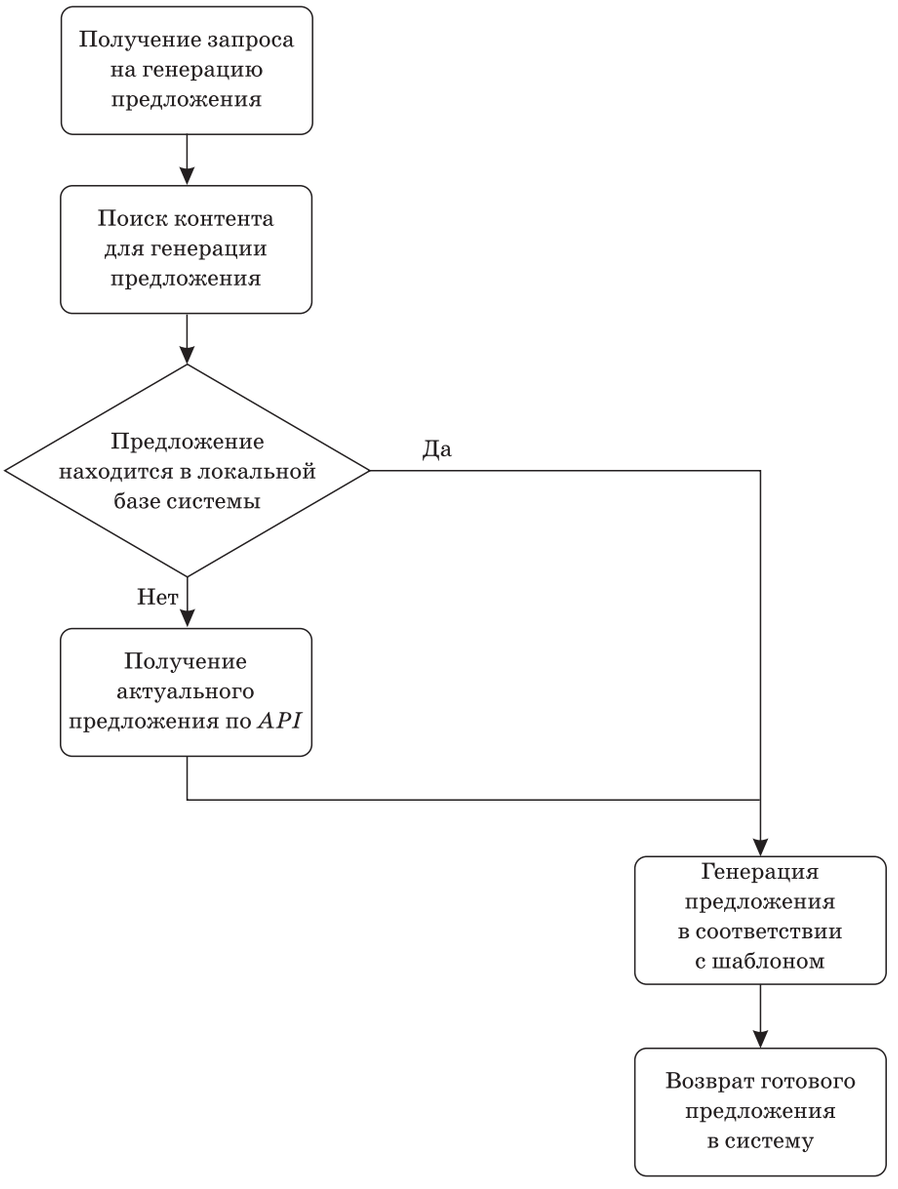

Рис. 2. Алгоритм работы модуля динамической генерации персональных предложений
Fig. 2. Algorithm of a module for dynamic generation of personalized offers

клик по кнопке); по расписанию (в случае встраивания баннера в запланированную рассылку либо в персонально генерируемые уведомления (приветственные), либо напоминания о рейсах, либо другие события, создаваемые программным обеспечением). Модуль при сохранении в базе данных блоков возможных объявлений (текстов/ссылок/изображений) должен на основе их формировать оптимальное объявление: подходящее по типу доставки (баннер необходимого размера на сайт или в *e-mail*, текстовое сообщение для *sms* или *push*-уведомления) или формату (что особенно важно для баннеров, которые встраиваются в блок определенного размера); оптимальное по предлагаемому типу дополнительной услуги (отель, такси, багаж, питание и т. д.), классу предлагаемой услуги (эконом, бизнес или определенный тип питания — например, халяль) или внешнему виду (цветовой схемы, расположения текста относительно изображения, разных изображений и выбора из данных конфигураций оптимальных).

В основе архитектуры лежит так называемый Кубернетис (*Kubernetes*) — система управления работающими на серверах приложениями. Основной структурной единицей этой системы являются ноды (*nodes*, что в переводе означает «узлы»). Нода — это виртуальный или физический сервер, выполняющий вычислительную

работу. В нашем случае используются две ноды, каждая из которых представляет собой физически выделенный сервер. Данное решение в текущей практике разработки является популярным, и его можно назвать отраслевым стандартом [4; 5]. Применение *Kubernetes* также дает возможность эффективно выстраивать балансировку потоков [6] и обеспечивать отказоустойчивость [7], что особенно значимо при работе модуля в высоконагруженных системах с большим количеством одновременно взаимодействующих с ними клиентов.

Kubernetes управляет нодами с помощью их объединения в кластеры. Кластер — группа связанных нод. С помощью механизма объединения нод в кластеры *Kubernetes* позволят обеспечивать повышенную отказоустойчивость выстраиваемого программного комплекса за счет возможности перераспределения функций выходящих из строя нод на другие ноды кластера.

В нашем случае создано четыре независимых кластера:

- кластер с *backend*-приложениями (бэкенд-приложениями), в котором сосредоточены все модули, выполняющие возлагаемые на программное обеспечение бизнес-функции;
- кластер для *frontend* (фронтенда), обеспечивающий взаимодействие с пользователем;
- кластер с воркерами (англ. *worker*, что в переводе означает «рабочий») — приложениями, обеспечивающими контроль за порядком и временем запуска приложений, за которые они отвечают, и фактически являющимися планировщиками;
- кластер для тестовых экземпляров приложений — этот кластер далее упоминаться не будет, так как не влияет на общую функциональность.

Каждый кластер *Kubernetes* включает в себя два типа нод [8]:

- управляющий (*manager*) — выполняет сервисные контролирующие функции. Управляющие ноды контролируют работоспособность остальных нод, управляют перезапуском нод и приложений или могут запускать процессы вышедшей из строя ноды на другой ноде;
- рабочий (*worker*) — принимает команды от управляющей ноды и выполняет их, осуществляя основную работу по реализации бизнес-логики приложения и запуск под собой *Docker*-контейнеров. Стоит отличать «рабочую ноду» от воркера, от *worker*-приложения (воркер-приложения), осуществляющего планирование запуска других приложений.

Все кластеры, кроме кластера с воркерами, объединяют по два сервера — управляющий и рабочий. В кластере с воркерами используется управляющий и два рабочих сервера для обеспечения большей производительности.

Ноды подразделяются на деплойменты (*deployment*) — наиболее крупные структурные единицы ноды. Деплоймент служит организационной единицей, позволяющей автоматизированно управлять набором запущенных под ним приложений с помощью меньших структурных единиц, называемых подами. Деплойменты осуществляют постоянный мониторинг находящихся в них подов, контролируя работоспособность и балансируя между ними нагрузку. В каждом деплойменте запускаются дублирующие друг друга поды для повышения устойчивости и возможности проведения обновления. Так, при наличии трех дублирующих подов и необходимости их обновления поды поочередно удаляются, разворачиваются новые с обновленной версией приложений. После удаления одного из трех подов нагрузку на себя берут оставшиеся два.

Выполнение команд и обработка информации происходят в приложениях, которые разворачиваются в виде *docker*-приложений (далее — «докеры»). Это — самая низкоуровневая ячейка *Kubernetes*, представляющая собой контейнер со средой, необходимой для работы приложений и непосредственно приложениями. В нашей архитектуре под содержит лишь один докер.

В разработанном программном обеспечении программы условно подразделяются на четыре основных типа:

- воркеры (*workers*) — приложения, структурно похожие на *backend*-приложения, но отвечающие за контроль расписания и запуск других приложений, которые по сути являются планировщиками и распространяют свои функции только на привязанные к ним приложения;
- обработчик (*job*) — часть воркера, реализующая определенную логику, то есть обработчик данных, который запускается по расписанию воркером. В основном это — сервисные приложения, отвечающие, например, за пересчет коэффициентов похожести пассажиров или за мас-

совую обработку статистики по объявлениям. Рассматриваемые программы характеризуются как высоконагруженные, поскольку осуществляют сложные вычислительные операции с большим количеством данных, что может приводить к перегрузке серверных мощностей, остановкам и перезапуску этих программ. Из-за этого они вместе с воркерами вынесены в отдельный кластер *Kubernetes* — производимые в нем операции не критичны с точки зрения пользователя, и их прекращение или медленная работа не сказываются на пользовательском опыте. Поэтому во избежание их влияния на критические приложения они вынесены в отдельный кластер. Обработчики структурно являются частью воркера — воркеры и привязанные к ним обработчики находятся всегда в одном *docker*-контейнере;

- *backend* (бэкенд) — приложения, выполняющие бизнес-логику программного обеспечения. Такого рода приложения работают в режиме приема запросов по *API* от фронтенда. Бэкенд-приложения наиболее критичны для программного обеспечения и выполняют функции, ориентированные на взаимодействие с пользователем. В связи с этим они имеют повышенную важность с точки зрения функционала, и к ним предъявляются повышенные требования в плане отказоустойчивости и безопасности;
- *frontend* (фронтенд) — приложения, обеспечивающие взаимодействие с пользовательским интерфейсом. Они стандартны и не относятся к инновационной части приложения, поэтому им особого внимания не уделено.

Каждое из приложений может быть запущено в нескольких экземплярах для повышения отказоустойчивости. Например, бэкенд-приложения запускают в трех идентичных экземплярах в одном деплойменте для распределения нагрузки, удобства обслуживания, обновления и исключения отказов в работе.

Для целей настоящего раздела архитектура рассматривалась упрощенно, в основных ее отличительных аспектах. Схематически она изображена на рисунке 3.

На рисунке 3 не указаны многие структурные элементы, в том числе находящиеся между изображенными, но выполняющие сугубо сервисные функции. Структурно каждое предложение встроено в цепочку связанных сущностей:

- предложения (*offers*) — такси, страховка, отели и т. д. Это — общие категории, означающие тип дополнительной услуги;
- предложения конкретизируются до наборов объявлений (*ad sets*) — обычно они соответствуют классам типов услуг в соответствующем предложении, например, «такси премиум», «такси бизнес», «такси эконом» для общего предложения «такси»;
- определенные объявления (*ads*), привязанные к наборам объявлений, каждое из которых является самостоятельной законченной сущностью, состоит из определенного рекламного текста или набора текстов и изображения (в случае баннерной рекламы).

Описанные сущности структурно являются вложенными, то есть каждое объявление привязано к набору объявлений, а каждый набор объявлений — к типу предложения. Задача данного модуля — формировать определенные предложения (*ads*). При разработке модуля важно учитывать, что авиакомпании, несмотря на генерацию огромного количества данных, обладают незначительным числом данных о каждом пользователе ввиду небольшого количества транзакций.

В ситуации недостаточной статистики относительно каждой личности особенно необходимо сделать таким образом, чтобы каждая демонстрация рекламы пользователю была для нас максимально ценной с аналитической точки зрения. Для этого важно сохранять компромисс между вариативностью, разнообразием рекламных объявлений и их статистической ценностью. Так, если бы в модуле генерации персональных предложений мы предусмотрели интеграции с четырьмя разными сервисами (*offers*), в каждом из которых было бы по три разных класса услуг (*ad sets*), а для генерации предложений в виде баннера использовали бы на каждый класс пять допустимых вариантов текста, пять допустимых изображений и десять форматов баннера, то количество вариантов баннеров, которые мы могли бы показать данному клиенту, составило бы $4 \times 3 \times 5 \times 5 \times 10 = 3\,000$ штук.

Становится понятным, что в любом случае речь не идет о выявлении персональных предпочтений. Но, даже если делать ротацию такого потенциального количества объ-

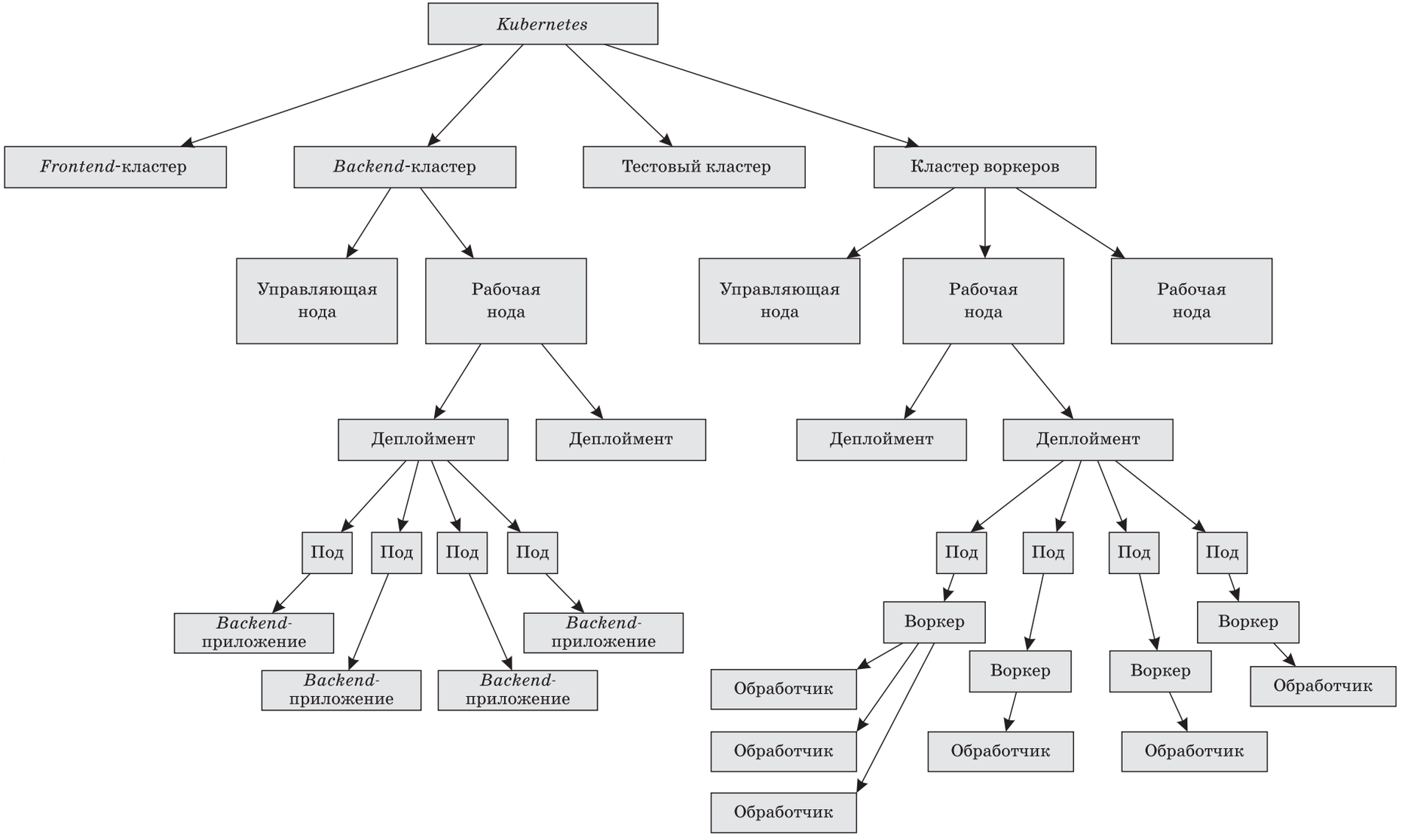

Рис. 3. Схема архитектуры программного обеспечения
Fig. 3. Software architecture diagram

явлений на аудиторию компании в целом, то в день в самом оптимистичном случае будет показано лишь по одному типу объявления. Это тоже мало. Если слишком снизить количество вариантов текстов, изображений и форматов баннеров, то получим недостаточную вариативность для показа объявлений. Полагаем, необходима золотая середина в отношении количества блоков, из которых складывается объявление.

Каждое объявление любого формата — это предложение от определенного поставщика в определенном городе с определенной ценовой категорией. С каждым рекламным объявлением при этом связан свой шаблон (*e-mail*, *sms*, баннер, *push*), ссылка (на сайт партнера или внутренний ресурс), используемый текст, используемое изображение, тип рекламного предложения, категория предложения, данные пользователя, аналитические данные (например, время показа объявления, время клика, время заказа и статус заказа). Все эти данные привязывают к каждому сгенерированному объявлению (*ad*).

Шаблоны — заранее зафиксированные в программном обеспечении форматы рекламы. Ссылка, на которую нужно направлять пользователя, по которой он кликает, выбирая данное предложение, встраивается в уникальный идентификатор клика, позволяющий нам отслеживать в дальнейшем совершаемые пользователем на сайте партнера транзакции.

Текст предложения может быть сохранен в двух вариантах, в зависимости от выбранного канала: для баннерных предложений используются картинки-баннеры, в которых текст служит частью изображения-баннера; для иных форматов текст «зашит» в шаблон предложения. Изображения, используемые в персональных предложениях, также могут быть представлены в двух вариантах: изображения-баннеры, то есть полностью оформленные рекламные предложения, включающие в себя как текстовую информацию, так и полноценное графическое оформление, в том числе муляжи кнопок; изображения для вставки в шаблон, что справедливо для рассылок *e-mail*.

Для *sms* и *push*-уведомлений изображения по умолчанию не используются. В качестве замены изображений можно применять иконки, вставляемые через коды *UNICODE*.

Предположим, что пассажир заново зарегистрировался. В отношении него системой прежде всего генерируются все необходимые агрегаты и теги, которые пересчитываются каждый раз, если у пользователя изменились те или иные данные. Для этого в системе определена отдельная сущность *Aggregate Tracking*, что отражено на рисунке 4, которая хранит для каждого пользователя информацию о том, в какой период в последний раз в отношении него изменены данные и в последний раз пересчитаны агрегаты и теги.

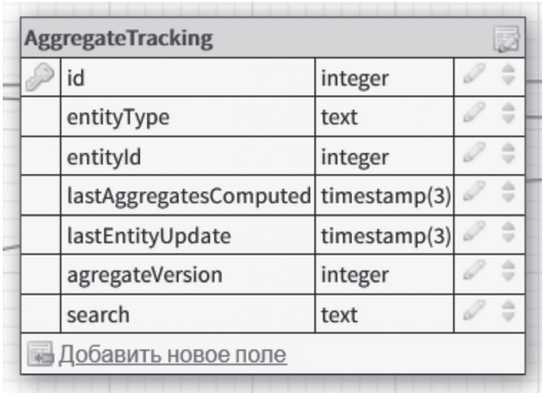

Рис. 4. Сущность *Aggregate Tracking*, отслеживающая информацию о пересчете агрегатов и тегов
Fig. 4. *Aggregate Tracking* entity that tracks information about recalculation of aggregates and tags

Стоит отметить, что расчет рекомендаций — затратная относительно ресурсов процедура, и поэтому она не проводится «на лету». Все агрегированные данные, такие как теги пользователей, коэффициенты похожести, персональные рекомендации, рассчитывают с помощью обработчиков по расписанию. Эту задачу выполняет именно кластер *Kubernetes* с обработчиками (воркерами) [9].

В результате проведенного исследования, на основе лучших практик работы с микросервисными приложениями, нами разработана модель модуля динамической генерации персональных предложений дополнительных услуг для пассажиров авиакомпаний, отвечающая современным стандартам в аспекте надежности, скорости работы и гибкости масштабирования. Разработанная модель позволяет отправлять клиентам персонализированные сгенерированные предложения, варьируя не только их содержание (тип/класс предложения), но и канал доставки, а также этапы жизненного цикла клиента.

Список источников

1. *Абрамов В. И., Чуркин Д. А.* Предиктивная аналитика взаимоотношений с клиентами как метод адаптации компании к изменениям и повышения ценности предложения // Экономика, предпринимательство и право. 2022. Т. 12. № 6. С. 1709–1722. DOI: 10.18334/epp.12.6.114842
2. *Абрамов В. И., Абрамов И. В., Поливанов К. В., Семенков К. Ю.* Цифровая трансформация системы управления отношениями с клиентами // Вопросы инновационной экономики. 2023. Т. 13. № 1. DOI: 10.18334/vinec.13.1.117051
3. *Абрамов В. И., Чуркин Д. А.* Оценка уровня зрелости системы управления взаимоотношениями с клиентами // Вестник университета. 2022. № 12. С. 5–13. DOI: 10.26425/1816-4277-2022-12-5-13
4. *Vayghan L. A., Saied M. A., Toeroe M., Khendek F.* Microservice based architecture: Towards high-availability for stateful applications with Kubernetes // 2019 IEEE 19th International conference on software quality, reliability and security (QRS) (Sofia, 22–26 July 2019). Piscataway, New Jersey: IEEE, 2019. P. 176–185. DOI: 10.1109/QRS.2019.00034
5. *Омаров А. С.* Модель рекомендательной системы для телекоммуникационных компаний // Наука и молодежь: материалы XVIII Всерос. науч.-техн. конф. студентов, аспирантов и молодых ученых: в 2 т. Т. 1 Ч. 1. Инженерно-технические науки (Барнаул, 19–23 апреля 2021 г.) / отв. ред. М. В. Гунер. Барнаул: Алтайский государственный технический университет имени И. И. Ползунова, 2021. С. 73–74.
6. *Сафронов Д., Стоноженко К. М., Никифоров И. В.* Автоматическая балансировка нагрузки между потоковой обработкой данных и внутренними задачами кластера с использованием Kubernetes // Современные технологии в теории и практике программирования: сборник материалов конф. (Санкт-Петербург, 23 апреля 2020 г.). СПб.: Политех-Пресс, 2020. С. 165–167.
7. *Басин Д. Д.* Анализ механизмов отказоустойчивости веб-приложений в платформах оркестрации контейнеров Kubernetes и Docker Swarm // Научно-технический семинар кафедры МОЭВМ: сборник докладов студентов и аспирантов. СПб.: Санкт-Петербургский государственный электротехнический университет «ЛЭТИ» имени В. И. Ульянова (Ленина), 2021. С. 12–19.
8. *Mao Y., Fu Y., Gu S., Vhaduri S., Cheng L., Liu Q.* Resource management schemes for cloud-native platforms with computing containers of docker and kubernetes // TechRxiv. 2020. DOI: 10.36227/techrxiv.13146548.v1
9. *Carrión C.* Kubernetes scheduling: Taxonomy, ongoing issues and challenges // ACM Computing Surveys. 2022. Vol. 55. No. 7. P. 1–37. DOI: 10.1145/3539606

References

1. Abramov V.I., Churkin D.A. Predictive analytics of customer relationships as a method of adapting the company to changes and increasing the value of the offer. *Ekonomika, predprinimatel'stvo i pravo = Journal of Economics, Entrepreneurship and Law.* 2022;12(6):1709-1722. (In Russ.). DOI: 10.18334/epp.12.6.114842
2. Abramov V.I., Abramov I.V., Polivanov K.V., Semenkov K.Yu. Digital transformation of the customer relationship management system. *Voprosy innovatsionnoi ekonomiki = Russian Journal of Innovation Economics.* 2023;13(1). (In Russ.). DOI: 10.18334/vinec.13.1.117051
3. Abramov V.I., Churkin D.A. Assessment of the maturity level of the customer relationship management system. *Vestnik universiteta (Gosudarstvennyi universitet upravleniya).* 2022;(12):5-13. (In Russ.). DOI: 10.26425/1816-4277-2022-12-5-13
4. Vayghan L.A., Saied M.A., Toeroe M., Khendek F. Microservice based architecture: Towards high-availability for stateful applications with Kubernetes. In: 2019 IEEE 19th Int. conf. on software quality, reliability and security (QRS). (Sofia, 22-26 July, 2019). Piscataway, NJ: IEEE; 2019:176-185. DOI: 10.1109/QRS.2019.00034
5. Omarov A.S. A model of a recommender system for telecommunication companies. In: Science and youth: Proc. 18th All-Russ. sci.-tech. conf. of students, graduate students and young scientists (in 2 vols.). Vol. 1. Pt. 1. Engineering and technical sciences (Barnaul, 19-23 April, 2021). Barnaul: Polzunov Altai State Technical University; 2021:73-74. (In Russ.).
6. Safronov D., Stonozhenko K.M., Nikiforov I.V. Automatic load balancing between streaming data processing and internal cluster tasks using Kubernetes. In: Modern technologies in the theory and practice of programming: Conf. proc. (St. Petersburg, 23 April, 2020). St. Petersburg: Polytech-Press; 2020:165-167. (In Russ.).
7. Basin D.D. Analysis of web application fault-tolerance mechanisms in Kubernetes and Docker Swarm container orchestration platforms. In: Department of Software and Computer Applications sci.-tech. seminar: Coll. pap. of students and graduate students. St. Petersburg: St. Petersburg Electrotechnical University "LETI"; 2021:12-19. (In Russ.).

8. Mao Y., Fu Y., Gu S., Vhaduri S., Cheng L., Liu Q. Resource management schemes for cloud-native platforms with computing containers of docker and Kubernetes. TechRxiv. 2020. DOI: 10.36227/techrxiv.13146548.v1
9. Carrión C. Kubernetes scheduling: Taxonomy, ongoing issues and challenges. *ACM Computing Surveys*. 2022;55(7):1-37. DOI: 10.1145/3539606

Сведения об авторах

Александр Дмитриевич Столяров
аспирант
Национальный исследовательский ядерный университет «МИФИ»
115409, Москва, Каширское шоссе, д. 31

Владимир Владимирович Гордеев
Генеральный директор
Общество с ограниченной ответственностью «АЭРОЛАБС»
115201, Москва, Котляковская ул., д. 3, стр. 13

Виктор Иванович Абрамов
доктор экономических наук, кандидат физико-математических наук, профессор кафедры управления бизнес-проектами
Национальный исследовательский ядерный университет «МИФИ»
115409, Москва, Каширское шоссе, д. 31

Поступила в редакцию 01.03.2023
Прошла рецензирование 24.03.2023
Подписана в печать 30.03.2023

Information about Authors

Aleksandr D. Stolyarov
postgraduate student
National Research Nuclear University “MEPhI”
31 Kashirskoe Highway, Moscow 115409, Russia

Vladimir V. Gordeev
CEO
Aerolabs LLC
3 Kotlyakovskaya St., bldg. 13, Moscow 115201, Russia

Victor I. Abramov
D.Sc. in Economics, PhD in Physical and Mathematical Sciences, Professor at the Department of Business Project Management
National Research Nuclear University “MEPhI”
31 Kashirskoe Highway, Moscow 115409, Russia

Received 01.03.2023
Revised 24.03.2023
Accepted 30.03.2023

Конфликт интересов: авторы декларируют отсутствие конфликта интересов, связанных с публикацией данной статьи.

Conflict of interest: the authors declare no conflict of interest related to the publication of this article.