Санкт-Петербургский государственный университет

Криворучко Денис Игоревич

Выпускная квалификационная работа

Разработка сервиса Заправок Яндекс.Go для бизнеса

Уровень образования: бакалавриат

Направление *02.03.03 «Математическое обеспечение и администрирование информационных систем»*

Основная образовательная программа *СВ.5006.2019 «Математическое обеспечение и администрирование информационных систем»*

Научный руководитель:
доцент кафедры информатики, к.т.н., Л. Н. Федорченко

Рецензент:
руководитель службы ООО «Яндекс.Такси Технологии», к. ф.-м. н. С. Е. Мацкевич

Санкт-Петербург
2023

Saint Petersburg State University

Denis Krivoruchko

Bachelor's Thesis

B2B Yandex.Go Tanker service development

Education level: bachelor

Speciality *02.03.03 "Software and Administration of Information Systems"*

Programme *CB.5006.2019 "Software and Administration of Information Systems"*

Scientific supervisor:
C.Sc., Informatics chair docent L.N. Fedorchenko

Reviewer:
head of service at "Yandex.Taxi Technologies" LLC, C.Sc. S.E. Matskevich

Saint Petersburg
2023

Оглавление

Введение

На сегодняшний день наблюдается устойчивая тенденция к автоматизации различных сфер жизни человека. Все больше компаний предлагают и продают свои услуги через интернет [21]. С этим связано увеличение общего объема рынка доставки еды и заказа такси. По данным РБК, объем рынка заказа такси вырос с 644 млрд рублей в 2020 году до 820 млрд рублей в 2021 году [20], а объем рынка доставки еды — с 290 млрд рублей в 2020 году до 613 млрд рублей в 2021 [17].

Несколько лет назад в приложение Яндекс.Go был добавлен сегмент B2B (business-to-business) [15]. В нем для корпоративных клиентов, помимо заказа такси и еды, доступны также Яндекс.Заправки — сервис, позволяющий оплачивать топливо на заправках, не выходя из машины, и Яндекс.Драйв — сервис аренды автомобилей. Данный сегмент рынка во многом отличается от сегмента B2C (business-to-consumer) и имеет свою специфику. Юридическим лицам, помимо качества используемых сервисов, важна также финансовая отчетность, возможность изменять тарифные планы и управлять корпоративной оплатой, — как персонально для отдельных сотрудников, так и массово для подразделений и департаментов. Такие требования к приложениям, призванным решать описанные задачи, делают особенно важными их надежность и скорость работы. Необходимо обеспечить приватность данных компаний и их сотрудников, а также обезопасить клиентов от мошенничества.

Приложение Яндекс.Go для бизнеса становится все более востребованным, а одним из самых активно развивающихся его разделов является сервис Заправок. Валовая стоимость услуг (GMV) и количество активных клиентов сервиса в месяц (MAU) непрерывно растет.

Данная работа посвящена разработке и развитию сегмента B2B сервиса Яндекс.Заправки. Перед автором была поставлена задача создать Автопарк — программную систему, позволяющую вести учет автомобилей компании, управлять ими и назначать на них лимиты, а также интегрировать описанную систему в личный кабинет пользователя и добавить информацию об автомобилях в отчеты Заправок.

1. Постановка задачи

Целью работы является программная реализация управления автомобилями в личном кабинете Яндекс.Go для бизнеса и использования лимитов на автомобили для оплаты заказа Заправок. Для её выполнения были поставлены следующие задачи:

1. Собрать требования к программной системе и разработать ее архитектуру.
2. Реализовать функциональность управления и использования автомобилей.
   (a) Реализовать создание и редактирование данных об автомобилях.
   (b) Реализовать закрепление лимитов на автомобиль за сотрудниками.
   (c) Реализовать новый вид лимитов: лимиты на объем залитого топлива.
   (d) Реализовать возможность оплаты топлива с лимита на автомобиль.
   (e) Добавить информацию об автомобилях в отчеты.
3. Провести отладку, тестирование и внедрение разработанной системы.

2. Обзор

В данном разделе представлен обзор приложения Яндекс.Go для бизнеса и его аналогов на рынке B2B, а также описаны программные инструменты, используемые в разработке.

2.1. Обзор аналогов

Многие компании, работающие в области доставки еды, заказа такси, каршеринга и заправок развивают сегмент B2B для своих продуктов.

В сфере заправок самыми крупными компаниями, предоставляющими подобные услуги, помимо Яндекса, являются Лукойл [14] и Газпромнефть [13]. Данные компании занимаются не только продажей, но и производством топлива. Это позволяет им предлагать покупателям более низкие цены от производителя, что дает преимущество на данном рынке в сегменте B2B.

Однако, у сервиса Яндекс.Заправки есть свои преимущества. Во-первых, он является частью экосистемы компании, что позволяет водителям использовать данный сервис из любого привычного для них приложения: Яндекс.Навигатор, Яндекс.Карты, Яндекс.Заправки. Во-вторых, B2B-сегмент сервиса Яндекс.Заправки интегрирован в приложение Яндекс.Go для бизнеса, где корпоративные клиенты могут подключить сервисы доставки еды, заказа такси, каршеринга и заправок для своих сотрудников, управлять выбранными услугами через единый интерфейс, а также формировать финансовые отчеты, что упрощает управление компанией. Таким образом, рассматриваемое приложение является уникальным на российском рынке.

2.2. Приложение Яндекс.Go для бизнеса

Разработанная автором программная система Автопарк, предназначенная для управления автомобилями, является частью приложения

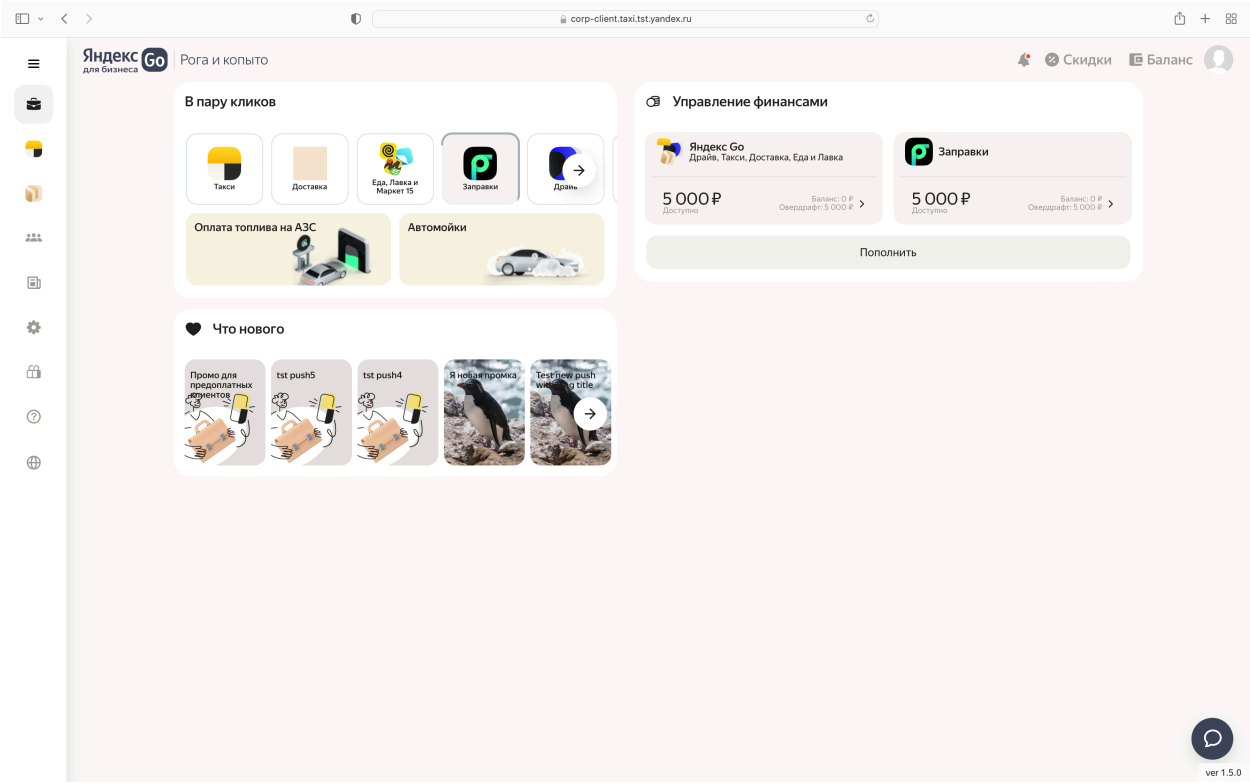

Рис. 1: ЛК Яндекс.Go для бизнеса: главная страница.

Яндекс.Go для бизнеса. Данное приложение рассчитано на корпоративных клиентов, которые пользуются сервисами Яндекса.

Главная страница личного кабинета приложения представлена на рисунке 1 (показаны тестовые данные). Здесь клиент имеет возможность открыть меню каждого доступного сервиса или открыть страницу подключения, посмотреть краткую информацию о балансе и перейти на вкладку пополнения корпоративного счета. Помимо этого, из меню доступны страница отчетности, управления сотрудниками и другие.

Реализованная автором функциональность представлена на странице ”Сотрудники и доступы”. На ней расположены вкладки управления сотрудниками и автомобилями, лимитами, методами оплаты и центрами затрат. Управление автомобилями осуществляется на вкладке ”Автопарк”, пример этого показан на рисунке 2. Здесь можно изменить марку, госномер, лимит и доступность автомобиля для сотрудников.

Лимиты позволяют задавать некоторые ограничения на использование сервисов, доступных в Яндекс.Go для бизнеса. В частности, корпоративный клиент может изменить доступность конкретных сервисов

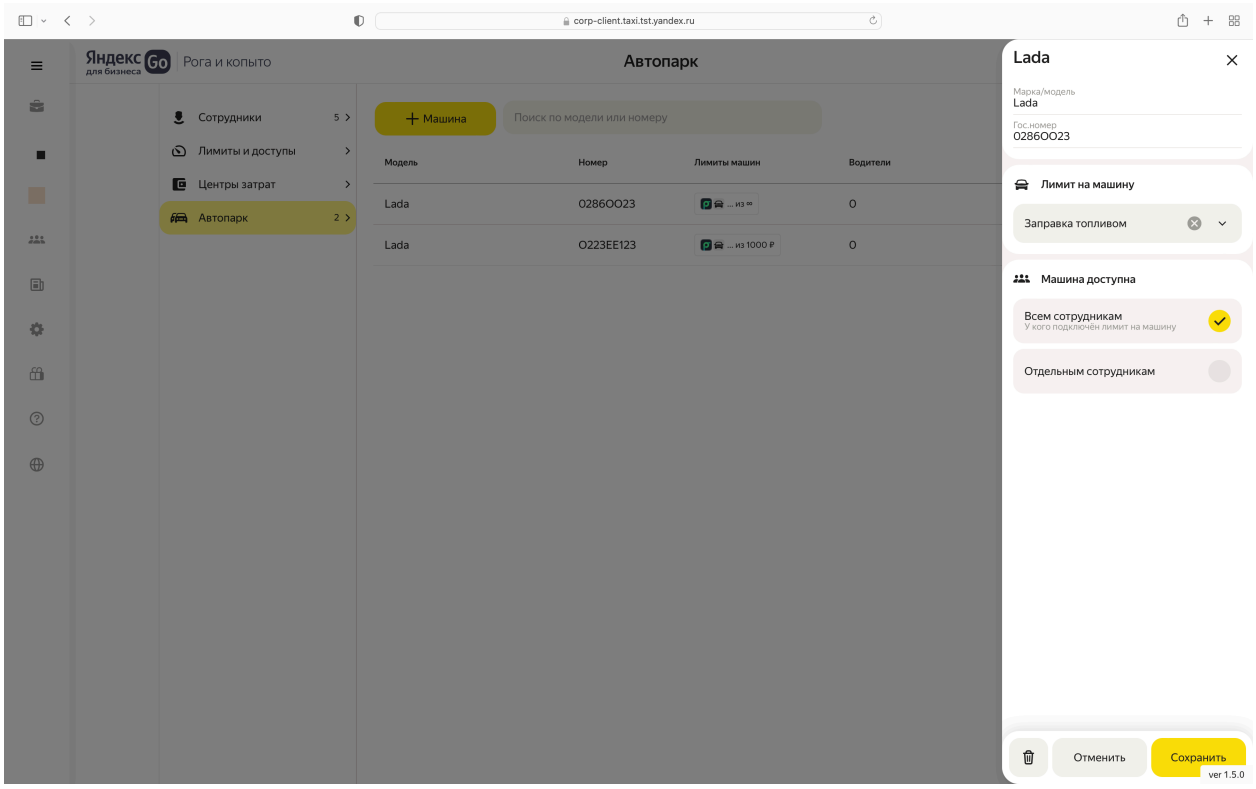

Рис. 2: ЛК Яндекс.Go для бизнеса: страница ”Сотрудники и доступы”.

для выбранного сотрудника и задать ограничение на сумму заказов по ним. Например, позволить сотрудникам выбранного департамента тратить на такси не более 2000 рублей в месяц.

Использование сервисов Яндекс.Go с корпоративной оплатой может происходить как в личном кабинете, так и в специализированном приложении. Например, для заправки служебного автомобиля за счет компании, водитель может зайти в Яндекс.Заправки и выбрать, с какого из доступных лимитов списать деньги. Сотруднику будут доступны те лимиты, которые указаны в личном кабинете Яндекс.Go для бизнеса.

2.3. Описание используемых инструментов

Функциональность системы разработана автором на языках C++ и Python3. Python3 обладает богатой стандартной библиотекой, позволяющей работать с http-запросами и файлами, проводить тестирование, писать асинхронный и многопоточный код. Однако, реализация на языке Python3 тех частей приложения, где критически важна скорость об-

работки запроса и возвращения ответа пользователю, приведет к ощутимой задержке, что является недопустимым. По этой причине данные части программной системы реализованы автором на языке C++ с использованием фреймворка userver [19]. В частности, на C++ реализовывалась функциональность цикла оплаты заказа.

Помимо обработки запросов, необходимо хранить пользовательскую информацию. Для хранения информации используются базы данных PostgreSQL [8] и MongoDB [7]. PostgreSQL — наиболее современная реляционная база, выбранная для хранения и обработки данных с постоянной, не изменяющейся структурой. Однако, в современном мире информация обладает свойством изменчивости и заранее продумать четкую схему порой невозможно. Для хранения данных подобного рода используется документо-ориентированная база MongoDB. Однако, важно хранить данные в нескольких экземплярах, поскольку с оригинальным хранилищем могут возникнуть проблемы: данные исчезнут или будут повреждены. Во избежание этого для всех хранилищ используется репликация данных [4].

Для репликации применяется MapReduce-система YT [10], которая используется в компании Яндекс повсеместно. Этот инструмент позволяет хранить и быстро обрабатывать данные, объем которых не позволяет работать с ними в рамках одной базы.

Разработанная автором программная система является частью приложения, в работе над которым принимает участие команда программистов, поэтому для совместной разработки используется система контроля версий Arc [1], созданная в Яндексе. Ее реализация и использование обусловлены невозможностью управлять кодовыми базами подобных размеров с использованием стандартных систем контроля версий, таких как Git.

При возникновении проблем в работе системы важно быстро их обнаружить, оперативно найти причину их возникновения и устранить ее. Поэтому для проверки текущего состояния сервиса и локализации ошибок используются инструменты Grafana [5] для мониторинга графиков и Kibana [6] для просмотра логов.

3. Сбор требований и проектирование

Поскольку реализованная автором в рамках данной работы функциональность системы Автопарк является частью приложения Яндекс.Go для бизнеса, первое требование к системе заключается в том, что она должна удовлетворять всем нормам и стандартам данного приложения. Этим обусловлен выбор инструментария и метод разработки — созданная автором система включена в микросервисную архитектуру Яндекс.Go [16] и использует существующие модели и базы данных. Однако, основная часть требований исходила от менеджеров приложения и была продиктована исключительно бизнесом.

Ожидалось, что система для управления автомобилями позволит корпоративным клиентам работать с автомобилями так же как с сотрудниками: вся информация об автомобилях должна быть собрана в одном месте, должна быть поддержана возможность задавать лимиты на автомобиль и оплачивать топливо с этого лимита в приложении Яндекс.Заправки. Помимо этого, необходимо было позволить пользователю разграничивать доступы сотрудников к лимитам автомобилей и добавить информацию об автомобилях в отчеты, которые формируются в личном кабинете.

Сущности сотрудник и автомобиль тесно связаны между собой, поскольку автомобили должны закрепляться за сотрудниками, а лимиты могут быть привязаны как к сотруднику, так и к автомобилю. В связи с этим, функциональность управления автомобилями реализована автором в микросервисе corp-users, где хранится логика управления сотрудниками и их лимитами. Кроме того, автором была добавлена функциональность проксирования запросов [18] в микросервис taxi-corp, предназначенный специально для этого. Для хранения данных об автомобилях автором было принято решение использовать реляционную базу данных, потому что изначально было понятно, что схема этих данных имеет четкую структуру и не будет изменяться с течением времени. В таких случаях реляционные базы имеют преимущество перед другими, так как реляционная алгебра, которая является их ядром, позволяет

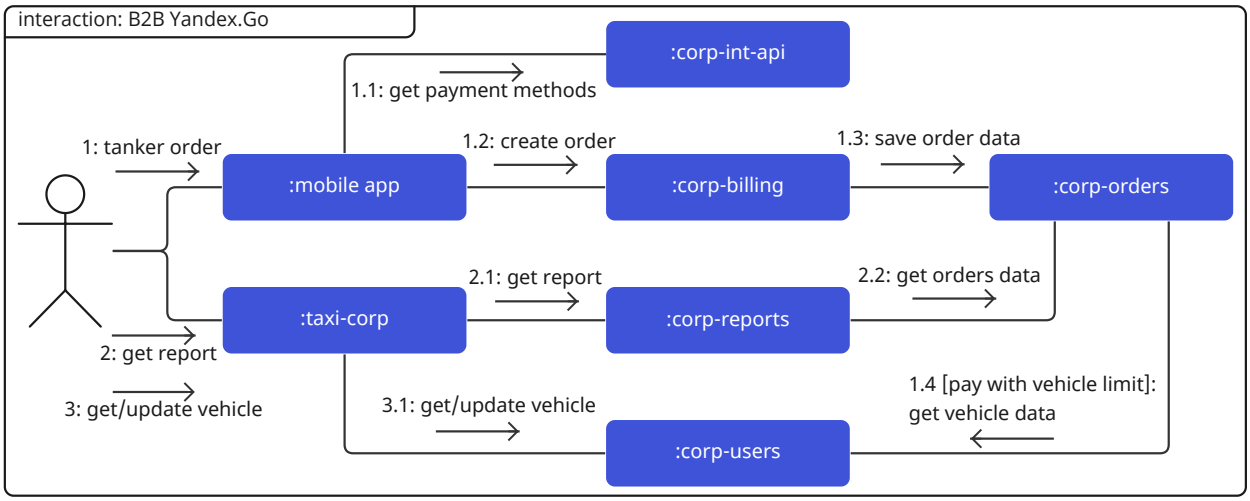

Рис. 3: диаграмма коммуникаций микросервисов в приложении Яндекс.Go для бизнеса.

оптимизировать операции, что увеличивает скорость запросов [11].

Чтобы корпоративные клиенты и их сотрудники имели возможность использовать лимиты на авто при оплате топлива, автором были внесены изменения в функциональность цикла оплаты заказа. В частности, в микросервис corp-int-api, откуда в приложении Яндекс.Заправки берется информация о доступных методах оплаты пользователя, и в микросервис corp-billing, где и обрабатывается запрос на создание заказа Заправок.

Помимо реализации логики управления автомобилями, необходимо было модифицировать логику формирования отчетов, чтобы в них попадала информация о заправленных транспортных средствах. Формирование отчетов реализовано в микросервисе corp-reports, однако здесь только обрабатывается информация о заказах, полученная из микросервиса corp-orders, поэтому для того, чтобы в отчеты попала нужная информация, изменения были внесены автором именно в corp-orders. Диаграмма коммуникаций описанной выше части приложения представлена на рисунке 3.

4. Детали реализации

В данном разделе представлено подробное описание реализации системы Автопарк.

4.1. Реализация создания, изменения и просмотра информации об автомобилях

Как было сказано в главе 3, для хранения данных об автомобилях, автором была создана реляционная база данных. В ней содержатся две таблицы: vehicles и vehicles_access. Первая — для хранения данных непосредственно об автомобилях клиентов. В ней содержатся идентификаторы автомобиля и клиента, которому он принадлежит, а также модель, номерной знак, лимит на Заправки и служебные поля, показывающие дату создания, изменения и удаления автомобиля. Вторая таблица нужна для хранения информации о доступности автомобиля сотрудникам организаций. Здесь содержатся данные о типе доступа к автомобилю (всего их может быть три: машина доступна всем сотрудникам организации, выбранным сотрудникам или сотрудникам конкретного департамента). Помимо этого, содержатся соответствующие идентификаторы пользователей и департаментов и вышеупомянутая служебная информация.

Для управления автомобилями из личного кабинета пользователя автором были созданы обработчики http-запросов. Помимо стандартной проверки (проверка доступа, проверка корректности задаваемого лимита), в запросах на создание и изменение автомобилей валидируется номерной знак, введенный пользователем. Во-первых, формат госномера должен соответствовать государственным стандартам [12] — эта проверка реализована с помощью регулярных выражений. Во-вторых, введенный номер региона должен содержаться в базе ГИБДД — для реализации данной проверки, таблица допустимых регионов была выгружена автором из открытых источников и сохранена в конфигурационный файл приложения. Поскольку пользователь при создании и из-

менении регистрационных знаков может использовать как русские, так и английские буквы в любом регистре, номера транспортных средств были унифицированы автором (приведены к английским буквам в верхнем регистре) для удобства поиска в базе.

В качестве лимитов на автомобиль автором было принято решение использовать уже существующую модель персональных лимитов сотрудников на Заправки. Этот подход облегчил использование системы для клиентов — не нужно создавать новые лимиты, можно переиспользовать уже существующие. Данное решение также упростило разработку — не пришлось добавлять логику работы с новыми моделями.

За каждым сотрудником или подразделением может быть закреплено несколько транспортных средств. Для удобства управления автопарком важно было дать пользователям возможность массово изменять лимиты на автомобили, принадлежащие, например, выбранному департаменту. На момент решения задачи, в системе уже была функциональность массового обновления лимитов сотрудников. Во избежание перегрузки приложения и медленной обработки запросов, использовалась концепция Tasks Queue [9]. Однако, изначально предполагалось, что количество автомобилей в личном кабинете пользователя будет на порядок ниже, чем количество сотрудников, а сам запрос в случае с автомобилями требует меньше проверок и обращений к базе данных, поэтому использование подхода Tasks Queue в этом случае имеет неоправданную сложность, в связи с чем автор принял решение создать обработчик http-запросов, в котором обновляются лимиты выбранных автомобилей.

Помимо этого, на стороне frontend-части приложения необходимо определять, нужно ли отображать вкладку Автопарк в конкретном личном кабинете, поскольку не у всех корпоративных клиентов подключен сервис Яндекс.Заправки. Для этого автором внесены изменения в два обработчика запросов: изменяющий и возвращающий информацию о сервисе Заправок по идентификатору клиента. В схему запроса первого и ответа второго, соответственно, добавлено новое поле is_fleet_enabled — признак включения Автопарка. Данное свойство

хранится в коллекции corp_clients, где содержится информация непосредственно о корпоративных клиентах и настройках их сервисов. Для инициализации данного поля существующим клиентам автором был реализован скрипт. Значение по умолчанию — False.

Логика работы с автомобилями предполагает частые обращения к базе данных, содержащую информацию о сотрудниках. Эта база имеет внушительные размеры, поэтому при извлечении и изменении хранимой информации, остро стоит вопрос скорости. Для оптимизации получения данных из нее, автором были проиндексированы все запросы, связанные с автомобилями.

Для того, чтобы при потере данных в результате каких-либо проблем в работе сервиса, у разработчиков была возможность восстановить информацию о действиях пользователя, связанных с Автопарком, автором реализовано сохранение истории внесения изменений в Автопарк (создание, изменение, удаление машин) и репликация таблицы с данными о машинах на YT [10] — система хранения данных, позволяющая работать с информацией эффективно и безопасно. Однако, подобные фоновые процессы не должны сильно влиять на производительность приложения и скорость ответа пользователям, поэтому для репликации используется концепция Cron Tasks [3] (это реализовано не автором), а история сохраняется автором с помощью балковых операций базы Mongo [2].

В результате реализации описанной выше логики управления автомобилями, корпоративные клиенты получили возможность создавать и редактировать данные об автомобилях в личном кабинете пользователя приложения Яндекс.Go для бизнеса.

4.2. Реализация закрепления лимитов на автомобиль за сотрудниками

После того, как в личный кабинет пользователя приложения Яндекс.Go для бизнеса была добавлена логика управления автомобилями и назначения лимитов на них, нужно было реализовать возможность

закрепления лимитов на автомобили за сотрудниками. В частности, необходимо было позволить корпоративным клиентам выбирать способ оплаты заказов Заправок: с персонального лимита сотрудника или с лимита заправляемого им автомобиля. Для реализации этой возможности автором рассматривалось два способа.

Первый — хранить в базе данных для каждого сотрудника флаг has_fleet_access, обозначающий, есть ли у него доступ к автопарку. Если значение флага — True, то сотруднику доступны только лимиты на автомобили, за которыми он закреплен, если False — только личный лимит. Второй способ — для каждого корпоративного клиента с подключенным автопарком создать в базе данных специальный лимит, имеющий свойство is_fleet_limit=True, который обозначает лимит на автомобили. Это работает следующим образом: у сотрудника в базе данных хранится ссылка на лимит, который за ним закреплен. Если этот лимит не имеет указанного выше свойства (is_fleet_limit=True), заправка производится с соответствующего персонального лимита. Если же за сотрудником закреплен лимит на автомобили, при оплате заправок ему будет предложено выбрать автомобиль, с лимита на который будет оплачен заказ. После этого, средства будут списаны с лимита, закрепленного за выбранным автомобилем.

В итоге автором был выбран второй способ, поскольку в таком случае не придется изменять структуры существующей базу данных, а также будет возможность использовать готовую функциональность лимитов, что упростит логику системы. Для инициализации данного специального лимита существующим клиентам в базе данных, автором был создан скрипт.

В результате реализации закрепления лимитов на автомобиль за сотрудниками, корпоративные клиенты получили возможность управлять доступностью Автопарка для различных сотрудников, закрепляя за ними лимиты на автомобиль в личном кабинете пользователя приложения Яндекс.Go для бизнеса.

4.3. Реализация лимитов на объем залитого топлива

В приложение Яндекс.Go для бизнеса автором также были добавлены специфичные для сервиса Заправки лимиты, задающие ограничения не на сумму заказов, а на количество залитого топлива. Все обработчики запросов на обновление, создание и получение лимитов используют общие схемы, поэтому для добавления нового вида лимитов на Заправки, в соответствующую схему было добавлено поле kind в котором хранится наименование вида лимита: ограничение на стоимость заказов или объем топлива. Помимо этого, были внесены изменения в логику работы приложения: добавлена функциональность обработки нового поля, то есть сохранения его значения в базу данных и получения из базы, а также проверки, что сотрудник может выполнить заказ и не превысил порог количества литров, задаваемый новым лимитом. Кроме того, для инициализации поля kind в базе данных, автором был создан скрипт. Значение по умолчанию — "money" (лимит на стоимость заказов).

В результате добавления лимитов на объем залитого топлива, пользователи приложения Яндекс.Go для бизнеса получили возможность ограничивать количество топлива, оплаченного сотрудниками с корпоративного счета компании.

4.4. Реализация возможности оплаты топлива с лимита на автомобиль

Для оплаты топлива через сервис Яндекс.Заправки предлагается использовать одно из мобильных приложений: Яндекс.Навигатор, Яндекс.Карты, Яндекс.Заправки. Когда сотрудник некоторой организации совершает заказ и выбирает корпоративный счет для оплаты, происходит запрос в приложение Яндекс.Go для бизнеса (микросервис corp-int-api, диаграмма 3), чтобы осуществить проверку данного метода оплаты. В частности, в одном из шагов верификации проверяется, что пользователь не превысил лимит. После этого, если все данные корректны,

происходит запрос на оплату (микросервис corp-billing, диаграмма 3) и на сохранение информации о заказе для дальнейшего формирования отчетов (микросервис corp-orders, диаграмма 3). Для того, чтобы поддержать лимиты на автомобили и на объемы топлива при оплате, автором были внесены изменения в логику всех перечисленных выше этапов цикла заказа.

Автором были внесены изменения в логику проверки метода оплаты, чтобы поддержать описанные выше новые виды лимитов. Проверка лимитов реализована следующим образом: каждый пользователь имеет свой виртуальный счет, на который записываются все его траты. При очередном заказе с персонального лимита происходит проверка, не превышают ли траты сотрудника за период, установленный лимитом, соответствующую сумму. В случае, если заказ доступен — его сумма записывается на виртуальный счет сотрудника. Для аналогичной работы с лимитами на автомобили и на объем залитого топлива, автором была обновлена схема обработчика запросов, который используется для описываемой проверки. В него добавлен новый опциональный параметр — идентификатор автомобиля. В случае, если он передан — будет проверяться именно лимит данного автомобиля. Эта проверка производится так же, как и с персональным лимитом сотрудника — для каждого автомобиля создан отдельный виртуальный счет, где ведется учет всех его трат. Случай с лимитами на объем залитого топлива реализован аналогично лимитам на стоимость заказа. Для каждого сотрудника и автомобиля, с целью учета литров, создан отдельный виртуальный счет. То есть, сейчас в приложении поддержано четыре вида виртуальных счетов, и, в зависимости от параметров запроса, при проверке используется конкретный.

Помимо этого, в обработчик запросов на оплату заказа автором внесены следующие изменения: в схему запроса добавлен параметр vehicle_id — идентификатор автомобиля. Он используется для записи заказа на виртуальные счета выбранного автомобиля, если это необходимо. Помимо этого, для формирования отчетов, данные о заказе предварительно сохраняются во временное хранилище. Эта функциональность так

же претерпела изменения — необходимо было добавить сохранение еще одного свойства заказа — описанного выше идентификатора заказа.

В результате реализации логики, описанной в данной секции, сотрудники организаций, подключенных к Яндекс.Go для бизнеса, получили возможность оплачивать топливо с лимита на автомобиль.

4.5. Реализация функциональности добавления информации об автомобилях в отчеты Заправок

В случае, когда сотрудник использует лимит на автомобиль для оплаты заказа Заправок, соответствующую информацию необходимо отразить в отчетах. Для удобства клиентов, автором была добавлена в отчеты следующая информация: идентификатор, номерной знак и модель заправленного автомобиля. Эта информация берется из базы данных заказов. Чтобы она там появилась, автором была изменена схема соответствующей таблицы — в нее были добавлены новые поля, а значения этих полей для каждого заказа запрашиваются из сервиса corp-users, в котором, как было сказано в главе 3, расположена логика управления автомобилями. Здесь автором реализован обработчик http-запросов, который по идентификатору возвращает всю информацию об автомобиле. Если в заказе использовался другой метод оплаты, добавленные в отчет поля отображаются пустыми.

Информация о заказе может изменяться в процессе его выполнения. Например, изначальный заказ — 100 литров топлива, но залито 98.9. Во избежание отображения некорректной информации в пользовательских отчетах, необходимо синхронизировать данные о заказах, для чего используется уже упомянутая выше концепция Tasks Queue — задачи на синхронизацию с идентификатором заказа в качестве аргумента ставятся в очередь для фонового выполнения. Для каждого заказа это происходит дважды: на этапе создания и завершения. Однако, в силу многопоточности и асинхронности приложения, иногда возникает проблема, что в результате выполнения фоновых задач по синхронизации информации о заказе, в базу сохраняются неправильные

итоговые данные. Например, заказ зависает в статусе выполнения. В таких случаях необходимо еще раз актуализировать его статус (поставить новую задачу в соответствующую очередь). Чтобы не делать это вручную каждый раз, автором в был реализован обработчик запросов, который выполняет данное действие. Данный обработчик используется в Админке — вспомогательном инструменте такси, предназначенном для решения проблем, выявления багов и управления различными частями приложения.

В результате изменения логики сохранения заказов, информация о заправленных автомобилях была добавлена в отчеты, формируемые в личном кабинете пользователя приложения Яндекс.Go для бизнеса.

5. Тестирование и внедрение разработанной системы

Перед тем, как реальные клиенты смогут воспользоваться разработанной функциональностью, ее необходимо интегрировать в приложение Яндекс.Go для бизнеса. Однако, перед этим нужно провести тестирование и апробацию, чтобы проверить написанный код и устранить ошибки.

5.1. Тестирование

Тестирование было проведено в четыре этапа. Первый этап — юнит-тестирование. Для каждого обработчика запросов автором были созданы тесты, содержащие как корректные, так и некорректные данные, чтобы протестировать не только ожидаемые сценарии работы программы, но и правильность логики обработки пользовательских запросов, содержащих ошибки в данных. Сервером должен быть обработан любой запрос, и возвращен либо ожидаемый ответ с кодом 200, либо сообщение об ошибке и соответствующий код: 4xx (ошибка на стороне клиента), 5xx (ошибка на стороне сервера) [22]. Это необходимо для отображения информации об ошибке клиенту и быстрого обнаружения и устранения неисправностей, если проблема на стороне сервера. Помимо новых тестов, запускались уже существующие, чтобы убедиться, что добавленная функциональность не ломает существующую логику.

Второй этап — интеграционное тестирование. Как было упомянуто выше, приложение Яндекс.Go для бизнеса имеет микросервисную архитектуру. Для проверки изменений, затрагивающих несколько сервисов, соответствующий пуллреквест (pull-request) должен пройти интеграционные тесты.

Третий этап — выкатка в testing — среду, в которой аппаратное обеспечение и код практически совпадают с версией для реальных пользователей (production). Это происходит следующим образом: в основную ветку разработки, где находится код, используемый в production, до-

бавляется пуллреквест с тестируемыми изменениями, после чего новая версия появляется на сервере тестовой среды. Далее необходимо сделать запрос на указанный сервер и убедиться, что всё работает корректно.

Четвертый этап проверки — Code Review. Другие разработчики проверяли код каждого пуллреквеста, а автор исправлял его, с учетом всех замечаний. После этого, в зависимости от серьезности внесенных после проверки кода изменений, все описанные выше этапы могли повториться. Затем функциональность попадала в основную ветку разработки.

5.2. Внедрение

После тестирования и добавления кода в основную ветку разработки, производилось постепенное обновление production-серверов для добавления туда новой функциональности. Сначала происходила выкатка в pre-stable — обновлялась половина серверов, на которых работает соответствующий сервис, в логику которого внесены изменения. Далее, с учетом состояния графиков (рисунок 4) и логов сервиса после выкатки в pre-stable, принималось решение, можно ли выкатывать сервис в stable, то есть обновлять оставшуюся половину серверов. Описанный процесс построен таким образом для того, чтобы в случае возникновения проблем на pre-stable, неудачные запросы перенаправлялись на сервера с предыдущей версией сервиса, которая работает корректно, и пользователь не испытывал проблем при работе в личном кабинете.

Перед внедрением разработанной программной системы Автопарк в приложение Яндекс.Go для бизнеса, автором была проведена дополнительная проверка. Команда, работающая над приложением, имеет доступ к нескольким личным кабинетам компаний, зарегистрированных специально для тестирования функциональности, с которой работают реальные пользователи. Автор зашел в личный кабинет одной из таких компаний и проверил, что все действия с автомобилями выполняются корректно. После этого было проведено внедрение на реальных пользователей.

Рис. 4: графики сервиса. NDA данные скрыты.

Внедрение разработанной программной системы в приложение Яндекс.Go для бизнеса проходило в несколько этапов. Сначала ответственные менеджеры убедились, что все требования выполнены, а система имеет ожидаемые свойства. Затем доступ к разработанной системе предоставлялся все большему проценту компаний-клиентов. Однако, автор принимал участие в данном процессе только с технической стороны — исправлял найденные ошибки и устранял возникающие в процессе взаимодействия с системой проблемы клиентов.

Заключение

В результате проделанной работы автором были получены следующие результаты:

1. Проведен сбор и анализ требований к реализуемой системе, выявлены ее ожидаемые свойства и разработана архитектура программной системы.

2. В системе реализованы следующие возможности для пользователей приложения Яндекс.Go для бизнеса:

   - Управление парком автомобилей компании и закрепление за ними параметров ограничений на использование сервиса Яндекс.Заправки.
   - Ограничение на количество залитого топлива при использовании сотрудниками корпоративного счета для заправки служебных автомобилей.
   - Просмотр данных о заправленных автомобилях в отчетах.

3. Разработанная программная система протестирована двухуровневой системой тестов и внедрена в эксплуатацию, как часть приложения Яндекс.Go для бизнеса.

В результате автором были внесены изменения в 5 микросервисов, а тестовое покрытие составило 100%. Временные затраты на разработку, отладку и внедрение — порядка 800 часов.

Реализованная в рамках данной работы программная система уже используется реальными пользователями приложения Яндекс.Go. В первые недели после полного внедрения системы, Автопарк начали использовать десятки клиентов, а уже через несколько месяцев — более сотни. Справка о внедрении прилагается.

Список литературы

[1] Arc — система контроля версий для монорепозитория. Доклад Яндекса. — URL: https://habr.com/ru/company/yandex/blog/482926/ (дата обращения: 21-01-2023).

[2] Bulk() — MongoDB Manual. — URL: https://www.mongodb.com/docs/manual/reference/method/Bulk/ (дата обращения: 28-03-2023).

[3] Cron Jobs — пособие для начинающих. — URL: https://tproger.ru/translations/guide-to-cron-jobs/ (дата обращения: 26-03-2023).

[4] Data Replication: Examples, Techniques How to Solve Challenges. — URL: https://airbyte.com/blog/what-is-data-replication/ (дата обращения: 18-01-2023).

[5] Grafana — платформа для визуализации данных, просмотра и анализа метрик. — URL: https://grafana.com/ (дата обращения: 23-01-2023).

[6] Kibana — инструмент визуализации и изучения данных. — URL: https://www.elastic.co/kibana/ (дата обращения: 24-01-2023).

[7] MongoDB Documentation. — URL: https://www.mongodb.com/docs/ (дата обращения: 17-01-2023).

[8] PostgreSQL: Documentation. — URL: https://www.postgresql.org/docs/ (дата обращения: 17-01-2023).

[9] Web Server Task Queue (How it works?). — URL: https://medium.com/@alanchu_65598/web-server-task-queue-how-it-works-64cabe4b5ac1 (дата обращения: 22-03-2023).

[10] YT: зачем Яндексу своя MapReduce-система и как она устроена. — URL: https://habr.com/ru/company/yandex/blog/311104/ (дата обращения: 18-01-2023).

[11] Заметки о SQL и реляционной алгебре. — URL: https://habr.com/ru/post/275251/ (дата обращения: 27-02-2023).

[12] Знаки государственные регистрационные транспортных средств. — URL: https://docs.cntd.ru/document/1200160380 (дата обращения: 20-03-2023).

[13] Личный кабинет Газпромнефть для бизнеса. — URL: https://opti-24.com/ (дата обращения: 17-10-2022).

[14] Личный кабинет Лукоил для бизнеса. — URL: https://auto.lukoil.ru/ru/ForBusiness/PersonalAccount (дата обращения: 16-10-2022).

[15] Личный кабинет Яндекс.Go для бизнеса. — URL: https://business.go.yandex/ (дата обращения: 15-10-2022).

[16] Микросервисы (Microservices). — URL: https://habr.com/ru/post/249183/ (дата обращения: 25-02-2023).

[17] Объем рынка доставки еды в России за 2019-2021 гг вырос почти в семь раз и достиг 613 млрд руб. — URL: https://marketing.rbc.ru/articles/13702/ (дата обращения: 14-10-2022).

[18] Паттерн проектирования «Заместитель» / «Proxy». — URL: https://habr.com/ru/post/88722/ (дата обращения: 26-02-2023).

[19] Просто и на C++. Основы userver — фреймворка для написания асинхронных микросервисов. — URL: https://habr.com/ru/company/yandex/blog/474438/ (дата обращения: 16-01-2023).

[20] Разгон до торможения. Аналитический центр при правительстве представил данные о рынке пассажирских автоперевозок за

2021 год. — URL: https://www.rbc.ru/newspaper/2022/04/21/62601dec9a79472bc87defb3 (дата обращения: 13-10-2022).

[21] Рынок онлайн-продаж в России вырос на 52%. — URL: https://habr.com/ru/news/t/696044/ (дата обращения: 12-10-2022).

[22] Список 55 ошибок HTTP с расшифровкой. — URL: https://allerrorcodes.ru/http-2/ (дата обращения: 01-03-2023).