Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

«КАЗАНСКИЙ (ПРИВОЛЖСКИЙ) ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

ВЫСШАЯ ШКОЛА ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И ИНТЕЛЛЕКТУАЛЬНЫХ СИСТЕМ

Направление: 09.03.04 – Программная инженерия

Профиль: Технологии разработки информационных систем

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

РАЗРАБОТКА ГЕНЕРАТОРА ЧАТ-БОТА ДЛЯ СБОРА ИЗОБРАЖЕНИЙ КАССОВЫХ ЧЕКОВ, ИХ РАСПОЗНОВАНИЯ И ФОРМИРОВАНИЯ АВАНСОВОГО ОТЧЕТА

Студент 4 курса

группы 11-605

«03» 06 2020 г. МГ Гаврилов М. В.

Научный руководитель

к.н.,заведующий кафедрой программной инженерии

«___»_____2020 г. Хасьянов А.Ф.

Директор Высшей школы ИТИС КФУ

канд. техн. наук

«___»_____2020 г. Абрамский М.М.

Казань – 2020

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Компании разных размеров часто отправляют своих сотрудников в командировку. Командировки осуществляются в основном за счет компании. Из этого возникает проблема управления и менеджментом расходов: фиксирование выдачи денег, проверка покупок. Вся информация для этого собирается в Авансовый Отчет. При составлении такого отчета необходимо четко фиксировать покупки, совершенные сотрудником во время командировки. На практике часто необходимо сохранять чеки физически, что создает большое количество проблем. Во-первых, не удобно собирать и хранить чеки для сотрудника и бухгалтера, также может возникнуть утеря чека. Во-вторых, при составлении отчета необходимо заполнить поля с информацией о покупках: название покупки, сумма покупки, документ подтверждающий покупку. Это является не полезной работой и занимает много времени. Поэтому имело бы смысл автоматизировать этот процесс в какой-то мере.

Чат-бот, который бы автоматизировал мониторинг покупок и создание отчета, также должен быть дешевым. Создание авансового отчета является формальной работой и не является фокусом для компаний, поэтому автоматизация процесса позволила бы перенаправить часть ресурсов на более важные части бизнеса. Автоматизировать в процессе создания авансового отчета можно несколько этапов. Одна из главных проблем у сотрудника является сбор, хранение и передача бухгалтеру чеков от покупок. Автоматизировать можно хранение чека и использование этой информации в качестве подтверждения покупки, например сохранить данные чека в цифровом формате. Также можно автоматизировать формирование авансового отчета. При формировании отчета, все чеки из цифровых источников добавляются в отчет.

Для всех, приведенных выше требований, подойдет разработка чат-бот одной из популярных социальных сетей или мессенджеров, так как они есть у большинства сотрудников и ими легко пользоваться.

Кроме автоматизации чтения, хранения чеков и формирования авансового отчета, возможно добавить еще дополнительный функционал, который будет решать другие проблемы, такие как находить необычные покупки и прогнозировать командировочные расходы, и тем самым упрощать командировки. Дополнительной функцией может быть умная система покупок, которая могла бы прогнозировать командировочные расходы работника заранее. С помощью алгоритмов машинного обучения, на основе предыдущих покупок возможно проанализировать и создать модель, которая бы прогнозировала расходы на командировку. Это позволит компаниям точнее планировать бюджет. Это может быть полезно например в случае длительной и денежно-затратной поездки. Так как в начале эксплуатации чат-бота, модель машинного обучения не будет достаточно обучена, необходимо было создать алгоритм, который до достаточного обучения модели выдавала бы прогноз расходов на командировку. Алгоритм будет состоять из нескольких параметров, такие как местоположение и население. Сами данные будут собираться с интернет ресурсов, таких как википедия.

Также у чат-бота должна быть система авторизации пользователей через чат-бота. Это позволит контролировать людей, которые имеют доступ к чат-боту, тем самым уменьшая количество запросов.

Интерфейс пользователя должен позволить настроить некоторые функции чат-бота и сделать деплой в указанное место.

Кроме требований к функционалу, есть еще и требование дешевизны. Это связано с тем, что мы предполагаем, что приложение будет ориентирована на малые и средние бизнесы, что означает небольшое количество ресурсов. После

анализа различных решений, мы пришли к выводу, что бессерверные вычисления позволят существенно сократить затраты на эксплуатацию. Это связано с тем, что большинство провайдеров облачных вычислений предлагаю достаточное количество вычислений для разрабатываемого чат-бота.
Также целью работы является возможность генерирования чат-бота на платформах разных провайдеров облачных вычислений. Это позволит компаниям выбрать провайдера. На данный момент нет разработок по созданию одного кода для всех платформ, поэтому решения, решающие эту проблему, будут новыми. Разработки для multi-cloud решений для бессерверные вычисления может быть полезно, так как проблема lock-in в платформу одного провайдера становится более важной: возможность поменять провайдера, например если новый провайдер имеет лучшее предложение или используемый провайдер увеличил цены.

1 РАЗРАБОТКА ЧАТ-БОТА

1.1 Проблема и Цели

Актуальность. Предприятия различных размеров периодически отправляют своих сотрудников в командировки. Для каждой из них, сотруднику покрываются расходы на такие нужды как проживание, транспорт, еду и другое. Для мониторинга этих покупок используется авансовый отчет, который регистрирует и подтверждает целесообразность покупок. Для составления авансового отчета приходится сохранять чеки покупок, что создает несколько проблем.

Во-первых, появляется проблема хранения и передачи чеков. Сотруднику в командировке приходится хранить бумажные чеки, что неудобно и иногда может произойти утеря чека, что зачастую влечет трудности.

Во-вторых, при составлении отчета все данные чека приходится переписывать вручную. Это непродуктивное занятие занимает значительное время сотрудников компании, которое могло было бы быть потрачено на более продуктивные занятия.

В-третьих, может появиться проблема распределения чеков по разным командировкам. Если сотрудник отчитывается о нескольких поездках, то иногда чеки могут быть распределены по отчетам неправильно. Требуется способ сбора, хранения и передачи в бухгалтерию чеков, чтобы исключить их потерю и сократить время на ручную обработку.

Другая проблема при командировках является прогноз командировочных расходов. Когда сотрудник отправляется на более длительную поездку его личных денежных ресурсов может быть недостаточно, поэтому появляется необходимость предварительно выдать денежные ресурсы этому сотруднику. Необходим способ, который предварительно сообщает о расходах на командировку.

Также существует проблема проверки целесообразности покупки. Редко, но бывает, что сотрудники злоупотребляют ресурсами компании и совершают покупки, ненужные для поездки. Требуется способ, который бы проверял, что совершены необычные покупки.

После анализа рынка, на сегодня в России существует только один конкурент в области упрощения сбора, хранения чеков и формированию отчета. Банк Точка с недавнего времени предоставляет приложение для чеков, которое сканирует и сохраняет чеки. В сравнении с ним, разрабатываемый чат-бот имеет несколько преимуществ.

Во-первых, он создает отчет и прикрепляет чеки к нему, что автоматизирует создание полного отчета. Также чат-бот предоставляет услугу составления примерной стоимости поездки, что позволяет планировать расходы и выдачу денег перед поездкой. Чат-бот еще может уведомлять о совершенных необычных покупках.

Во-вторых, приложение от банка предоставляется только клиентам этого банка. Только малое количество предприятий пользуются услугами Точки, поэтому для использования услуги им пришлось бы потратить ресурсы на использование нового банковского сервиса.

В-третьих, стоимость услуги от банка Точка является довольно дорогой. За каждый чек приходится платить 15 рублей. Разрабатываемое приложение абсолютно бесплатно для малых и средних предприятий благодаря выбранным технологиям и некоторым другим приемам.

1.2 Задачи

Чат-бот должен иметь следующие функции:

- Чтение фотографии чека, получение информации по покупкам и сохранение этой информации.
- Генерация отчета и отправка пользователю в удобном формате.
- Менеджмент пользователей.
- Умные функции контроля.

Создать решение, которое позволит максимально просто использовать код в выбранных облачных платформах с использованием бессерверных вычислений. Решение должно быть максимально дешевым.

1.3 Технологии

В качестве приложения для чат-бота будет использован Telegram. В выборе приложения играли роль две причины. Первая, Telegram популярен и знаком большому количеству людей. Вторая, API, который предоставляет Telegram. Telegram предоставляет все нужные функции для работы чат-бота: получение текста, фотографий чека.

В выборе технологий для реализации чат-бота были рассмотрены традиционные серверные технологии и бессерверные вычисления. После анализа, оказалось, что бессерверные вычисления больше соответствуют требованиям заданным для разработки чат-бота.

Чат-бот не требует постоянной работы, его работа происходит на основе событий, которые являются сообщением или командой. Бессерверные вычисления идеально вписываются в event-based архитектуру. Помимо этого у бессерверных вычислений есть большой ряд преимуществ.

В архитектуре на основе контейнеров количество развернутых контейнеров определяется разработчиком заранее. В отличие от этого, в архитектуре бессерверных вычислений серверная часть автоматически масштабируется в зависимости от количества запросов.

Бессерверных функции работают только когда это нужно, тогда как за сервер нужно платить значимую сумму даже при самых малых нагрузках. К примеру, AWS Lambda дает бесплатно 1 миллион запросов в месяц, для малого предприятия этого достаточно. Остальные провайдеры имеют примерно такое же количество бесплатного пользования.

Интерфейс API делает работу чрезвычайно интуитивно понятной, реализация собственной маршрутизации может происходить за счет этой видимости.

Размеры и ограничения: если мы проксируете все - тогда нам придется выбрать размер, время ожидания и т. д., которые будут соответствовать всем нашим конечным точкам RESTful.

В качестве языка программирование, был выбран Python3. Он поддерживается всеми тремя выбранными платформами, на которых чат-бот будет генерироваться: AWS, Azure, Google Cloud. Также он работает быстро на все платформах, как показано на изображение ниже.

Cold Starts 2019	node8.1	python3	.net2	ruby3.5	go1x	java
128mb	217	242	456	259	342	670
1024mb	190	245	316	210	340	564
3008mb	191	241	321	217	295	479

Рис.1 Скорость работы языков на AWS Lambda

Другой причиной выбора языка стало использование машинного обучения для умных функций чат-бота, так как язык имеет большое количество инструментов для этого.

1.4 Реализация

Чат-бот состоит из одного хендлера, который принимает запрос от серверов Telegram, потому что telegram имеет только одно соединение через webhook. Все команды пишутся пользователем текстом, и в handler уже парсятся.


```
"message":{
  "message_id":77,
  "from":{
    "id":207777314,
    "is_bot":false,
    "first_name":"Maxim",
    "last_name":"Gavrilov",
    "username":"grandterr",
    "language_code":"en"
  },
  "chat":{
    "id":207777314,
    "first_name":"Maxim",
    "last_name":"Gavrilov",
    "username":"grandterr",
    "type":"private"
  },
  "date":1586544370,
  "text":"dsds"
}
```


Первым шагом будет сбор данных в переменные из запроса от чат-бота.
При получении запроса от чат-бота, проверяются привилегии. Добавлены два вида пользователей: администратор и пользователь. Разделение привилегий нужно для менеджмента пользователей. Администратор имеет возможность добавлять и блокировать пользователей, чтобы не допустить внешнего

пользование чат-ботом, что может увеличить расходы. Информации о пользователях храниться в таблице базы данных USERS_TABLE. Первый пользователь автоматически становиться администратором, чтобы у него была возможность добавлять новых пользователей уже систематически. Добавление пользователей происходит двумя методами: командой “Добавить: nickname”, что создает начальную запись в базе данных. После того когда, указанный пользователь зашел, формируется полная запись в базе данных с нужными параметрами. Реализован второй способ добавления пользователей, на случай, когда не известен nickname пользователя, который происходит через механизм deeplinking. Deeplinking механизм позволяет добавить дополнительный параметр, токен, в url для старта чат-бота, после чего этот токен для доступа исчезает. Токеном в чат-боте служит случайное число, при добавлении с помощью команды “Add link: “ создается url, который передается пользователю. После входа, запись в базе данных с токеном исчезает и url перестает работать. Таким образом неавторизированные люди не могут получить доступа к чат-боту.

Администратор также может просматривать отчеты отдельных сотрудников. Созданы два вида команд: “Отчет:” для получение отчета по дате, а и также команда “Отчеты:”, которая показывает последние 3 отчета пользователя.

Пользователю доступны команды:

- “Начать поездку” - позволяет создать запись с новым отчетом, все добавленные после этого чеки будут добавляться в этот отчет. Переводит пользователя в состояние “in_trip”.
- “Закончить поездку” - завершает формирование отчета и убирает состояние “in_trip” пользователя. Также формируется отчет и отсылается в формате pdf.
- Отправка фотографии чека.

- “Отчет:” - отправка пользователю отчета за определенную дату.
- “Отчеты:” - отправка пользователю последние 3 отчета.
- ”Забыл:” - Во время тестирование было замечено, что пользователь иногда забывает добавить чек и вспоминает о нем, после того как поездка закончилась и отчет был сформирован. Поэтому была добавлена данная команда, которая при ее активации добавит следующий чек в последний отчет.

Некоторые команды требуют сохранения состояния. Единственным способом для добавления состояний в Serverless - это хранения переменных состояний в базе данных. К примеру, состояние “in_trip” храниться в записи пользователя в таблице пользователей.

1.5 Сбор данных по чеку

Первым шагом по сбору данных является обработка чеков. Существует несколько способов для решения этой задачи.

Первый рассмотренный способ был OCR чтение текста чека. OCR может использоваться для преобразования физического бумажного документа или изображения в доступный электронная версия с текстом. Например, если отсканировать бумажный документ или фотографию с помощью принтера. Скорее всего, принтер создаст файл с цифровым изображением. Файл может быть в формате JPG / TIFF или PDF, но новый электронный файл все еще может быть только изображением оригинального документа. Можно загрузить это созданный им отсканированный электронный документ, содержащий изображение, в программу распознавания текста. OCR обрабатывает цифровое изображение, находя и распознавая символы, такие как буквы, цифры и символы. Некоторые программы OCR просто экспортируют текст, тогда как другие программы могут преобразовать символы в редактируемый текст прямо

на изображении. Расширенное программное обеспечение OCR может экспортировать размер и форматирование текста, а также макет текста, найденного на странице. Но такой способ не подходит для нашей задачи. Основная проблема заключается в скорости работы функции и стоимости работы функций. Чат-боту нужно работать довольно быстро, чтобы отвечать на команды, это одна из проблем. Другая проблема в том, что рассчитывание стоимости работы функций на платформах рассчитывается не только по количеству вызовов, но и времени работы. OCR сильно ухудшит все эти показатели, поэтому этот способ лучше не использовать.

Другие два способа связаны с чтением QR кода. Эти два способа, как и первый используют машинное обучение для распознавания чека, но два последних читают только один элемент чека.

QR-код – это двумерная версия штрих-кода, известная по упаковке продуктов в супермаркете. Первоначально разработанный для оптимизации процессов в логистике автомобильной промышленности, QR-код нашел свое применение в мобильном маркетинге благодаря широкому распространению смартфонов. «QR» означает «Быстрый ответ», что означает мгновенный доступ к информации, скрытой в Кодексе. QR-коды набирают популярность, потому что технология «с открытым исходным кодом», то есть доступна для всех. Существенными преимуществами QR-кодов по сравнению с обычными штрих-кодами являются большая емкость данных и высокая отказоустойчивость.

После поисков лучшего способа чтения QR кода, мы выявили два лучших способа проверки QR кода.

Первым способ – это библиотека Opencv. Opencv представляет собой библиотеку для работы с компьютерным ведением. Она работает в реальном времени и хорошо справляется с обработкой изображений, видео. Кроме этого

она способна распознавать лица и объекты. Но как и с первым решением есть проблема скорости. Добавить библиотеку Opencv в функцию не будет составлять большого труда, но проблема скорости все равно будет присутствовать. Так как сразу несколько процессов происходят при чтении чека, в особенности отклик от государственных серверов, из-за этого функция может работать большое количество времени. Также после нескольких тестов с фотографиями чеков, было выявлено, что Opencv не очень хорошо находит qr на плохих фотографиях, что qr код не выделен достаточно хорошо. Поэтому было решено и этот способ обойти и найти какой-то более быстрый. Второй способ это использовать сторонний сервис, который быстро бы обрабатывал qr код и возвращал хранимые переменные в нем. После поиска нашелся сервис, который предоставляет бесплатное API для сканирования чеков. Сервис qrserver имеет api, который просто при отправлении фотографии qr кода быстро возвращает ответ с содержанием qr кода. Одним из главных преимуществ этого сервиса является то, что он может определить qr код на очень плохих фотографиях. Это позволяет не беспокоится о качестве изображения. Также qrserver api в случае ошибок, дает некоторую информацию о виде ошибки.
Из запроса чат-бота нужно выделить ссылку на фотографию. Telegram часто отправляет разное количество файлов одного фото, что зависит от начального разрешения. Поэтому выделяется второй файл фотографии, чтобы удостовериться, что оно будет существовать и разрешение достаточно для чтения qr кода. После чего происходит запрос к api сервиса по чтению qr кодов, и в ответ получаем следующее.


```
[
  {
    "type":"qrcode",
    "symbol":[
      {
```

```
          "seq":0,

"data":"t=20170310T100500&s=1213.46&fn=8710000100256778&i=5219&fp=68885226&n=1",
          "error":null
       }
     ]
  }
]
```


В data хранятся переменные нужные, для получения информации по покупкам. Поэтому их нужно распределить по переменным.

```
t=re.findall(r't=(\w+)', qr_data)[0]
s=re.findall(r's=(\w+)', qr_data)[0]
fn=re.findall(r'fn=(\w+)', qr_data)[0]
i=re.findall(r'i=(\w+)', qr_data)[0]
fp=re.findall(r'fp=(\w+)', qr_data)[0]
```

Получить данные по покупкам возможно только от налоговой. ФНС предоставляет API через который можно получить информацию по чекам - proverkacheka.nalog.ru. Для доступа сначала нужно получить доступ, отправив запрос с телефоном номера, по смс придет код доступа, который нужно сохранить.


```
{
  "document":{
    "receipt":{
      "fiscalDocumentNumber":196631,
      "receiptCode":3,
      "operator":"Гозалишвили Дмитрий Гиаевич, Продавец-кассир",
      "kktRegId":"0001723956006765    ",
      "fiscalSign":2220296527,
      "ecashTotalSum":20309,
```

```
"user":"ООО \"Агроторг\"",
"shiftNumber":322,
"dateTime":"2020-03-06T10:39:00",

"rawData":"AwDEAhEEEAA5Mjg5MDAwMTAwMzgyNDA4DQQUADAwMDE3MjM5NTYwMDY...",
"taxationType":0,
"requestNumber":124,
"userInn":"7825706086",
"items":[
  {
    "quantity":1,
    "price":5077,
    "sum":5077,
    "name":"*3691740 FAIRY Ср.Н.Р.АР.Р.Ж/АЛ.В.450мл"
  },
```


В ответ api от ФНС присылает информацию, которую хранит по чеку.
При тестировании чат-бота было обнаружено, что чеки покупок, которые были совершены недавно, не обрабатываются чат-ботом. Проблема оказалась в том, что разные компании имеют свой процесс передачи информации в ФНС. Например, если касса находится оффлайн, то информация о покупках появится позже. Поэтому появилась необходимость в позднем обращении к фнс. Просить пользователя добавить чек чуть позже ухудшит опыт использования, а также увеличит вероятность забыть добавить этот чек опять. Из этого следует, что информацию по чеку нужно будет сохранить в базе данных. Для автоматической проверки таких чеков нужно событие, которое запускало бы проверку чеков. Возможно запускать проверку, когда приходит запрос на новый чек, но тогда возможно слишком большое количество пустых запросов.

Наилучшим выходом оказалось периодическое событие. У serverless платформ есть функциональность scheduled functions, которые позволяют через какой-то промежуток времени создавать событие. Достаточно каждый час проверять это. Этот ивент имеет маркер “check”, который ловится handler-ом функции и запускает проверку не проверенных чеков.

1.6 Генерация отчета

Генерация отчета происходит в формате pdf, так как он чаще всего используется и для работы с ним существуют хорошие инструменты. Библиотека ReportLab предоставляет обширные графические возможности для создания pdf файлов. Кроме обширного количества методов, генерация pdf происходит очень быстро, что играет важную роль для безсерверных функций. Для создания документа создается документ с помощью метода SimpleDocTemplate(). Для генерации отчета нужно сформировать таблицу с помощью Table(). Также был изменен стиль таблицы. Шрифт по умолчанию не работает с кириллицей, поэтому нужно добавить шрифт как файл в формате .ttf. В результате получается таблица в таком виде:

ID Чека	Дата	Покупка	Сумма
0001723956006765	2020-03-06	R.SP.Шок.ЦЕЛ.ЛЕС.ОРЕХ мол.100г	5077
0001723956006765	2020-03-06	FAIRY Ср.Н.Р.АР.Р.Ж/АЛ.В.450мл	5078

Рис.2 Часть генерируемого отчета.

1.7 База данных

Для выбора базы данных для начала нужно было определить в какой форме информация по чекам будет хранится. Хранение чека в текстовой форме

является необходимой, так как данные используются как в генерации отчета, так и для умных функций, что требует обработки информации с чеков. Возможно было использовать два вида баз данных: традиционную базу данных, например MySQL, либо использовать NoSQL решения, которые отличаются у каждого провайдера. В нашем случае было решено использовать NoSQL решения от провайдеров: DynamoDB, CosmosDB, Datastore. Для того есть несколько причин:

- NoSQL базы данных от провайдеров позволят сократить стоимость, так как каждый провайдер предоставляет достаточно места для хранения чеков для большинства малых и средних предприятий.
- Получение и хранение информации по чекам происходит в формате JSON, для чего NoSQL базы данных подходят хорошо.

В функции используются несколько библиотек. Основные библиотеки, которые доступны на платформах каждого провайдера находятся в текстовом файле requirements.txt, платформа функций автоматически добавит эти зависимости. Но не которые, такие как reportlab.platypus такое решение не подошло, поэтому оно было добавлено в пакет для деплоя.

2 ГЕНЕРАЦИЯ ЧАТ-БОТА

2.1 Задачи

- Создать код, который можно будет максимально использовать между всеми тремя провайдерами.
- Создать конфигурационный файл.
- Создать пользовательский интерфейс для получения доступа к налоговой, к аккаунту платформы и конфигурации чат-бота.
- Успешно произвести деплой на аккаунт пользователя.

2.2 Конфигурирование кода

Одна из главных проблем облачных провайдеров это lock-in в одной платформе. Это является проблема, потому что в какой-то момент компания может перейти на другую платформу по техническим, финансовым, политическим или другим причинам. Данная процедура является затратной, так как на данный момент не существует общих стандартов, каждый провайдер создает свои. Из-за этого многие разработки и сервисы каждого провайдера различаются, что создает проблему при переходе на другую платформу. Возможно ли создать код, который будет работать на всех платформах или с минимальными изменениями? Бизнес логика будет работать везде, так как каждый провайдер поддерживает большинство языков. Проблема заключается в несовместимости сервисов от провайдеров. Каждая платформа разрабатывает свои полезные сервисы, чтобы привлечь к себе пользователей, но это создает и несовместимость платформ.

Поиск существующих решений ничего не дал. Во многом, потому что эта проблема недавно появилась и только некоторые статьи на этот вопрос только начинают появляться. Также формирования кода, который будет работать с минимальными изменениями на платформах нескольких провайдеров во

многом зависит от самого кода. Главным вопросом является какие сервисы от конкретного провайдера используются в коде. В нашем коде используется только один сервис, предоставляемый конкретным провайдером - это NoSQL база данных. После некоторых экспериментов было выявлено, что возможно поместить код этого сервиса в отдельный файл, import его в файле с основным хендлером. Создать три файла с работой для базы данных, по одному для каждого провайдера. А уже при генерации в коде файла с хендлером заменить import на нужный файл. В коде переменная связанная с импортом сохраниться, а импортируемый файл будет изменяться. Таким образом возможно иметь один код и генерировать его сразу для платформ трех провайдеров.
Деплой происходит через скрипт, который использует serverless framework для подключения к выбранному провайдеру и деплою кода.

2.3 Создание конфигурации:

Для конфигурации функций был создан файл шаблон для конфигурационного файла. В шаблон в определенные места будут вставлена нужная конфигурация, в зависимости от выбранных пользователем настроек. Выглядит часть шаблона вот так:

```
provider:
  name: <<provider
  runtime: python3.7
  stage: prod
  region: <<serverLocation

  environment:
    RECEIPTS_TABLE: ${self:service}-${opt:stage, self:provider.stage}
    USERS_TABLE: 'usersTable'
    TOKEN: <<telegramToken
```

YOUR_PHONE: <<phone

YOUR_GOV_PASS: <<govPass

В эту часть будут вставленные некоторые настройки пользователя, такие как локация серверов, телефон, выбор провайдера и другие.
Изменения в шаблоне производятся с помощью библиотеки ruamel.yaml, которая позволяет производить изменения во все конфигурационном файле.

2.4 Front-end

Пользовательских интерфейс состоит из двух шагов.

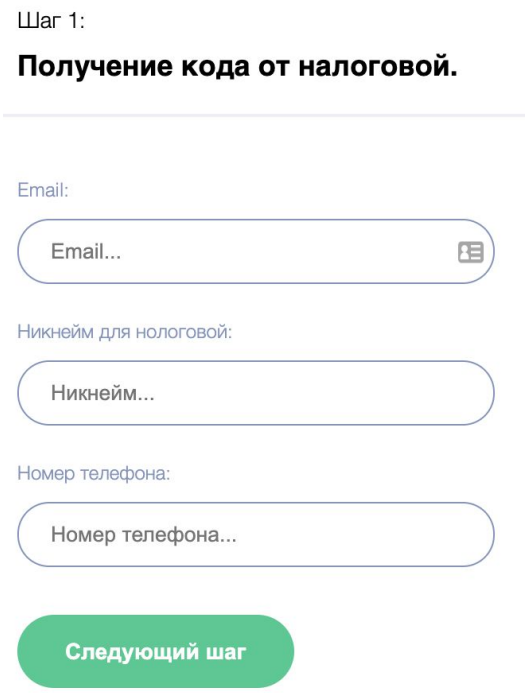

Рис.3 Первый шаг интерфейса

На первом шаге пользователь должен ввести инфомацию, которая будет передана в налоговую службу. При этом пользователю придет смс со

специалным кодом доступ, который будет использоваться для доступа к серверам налоговой при получении информации по чекам.

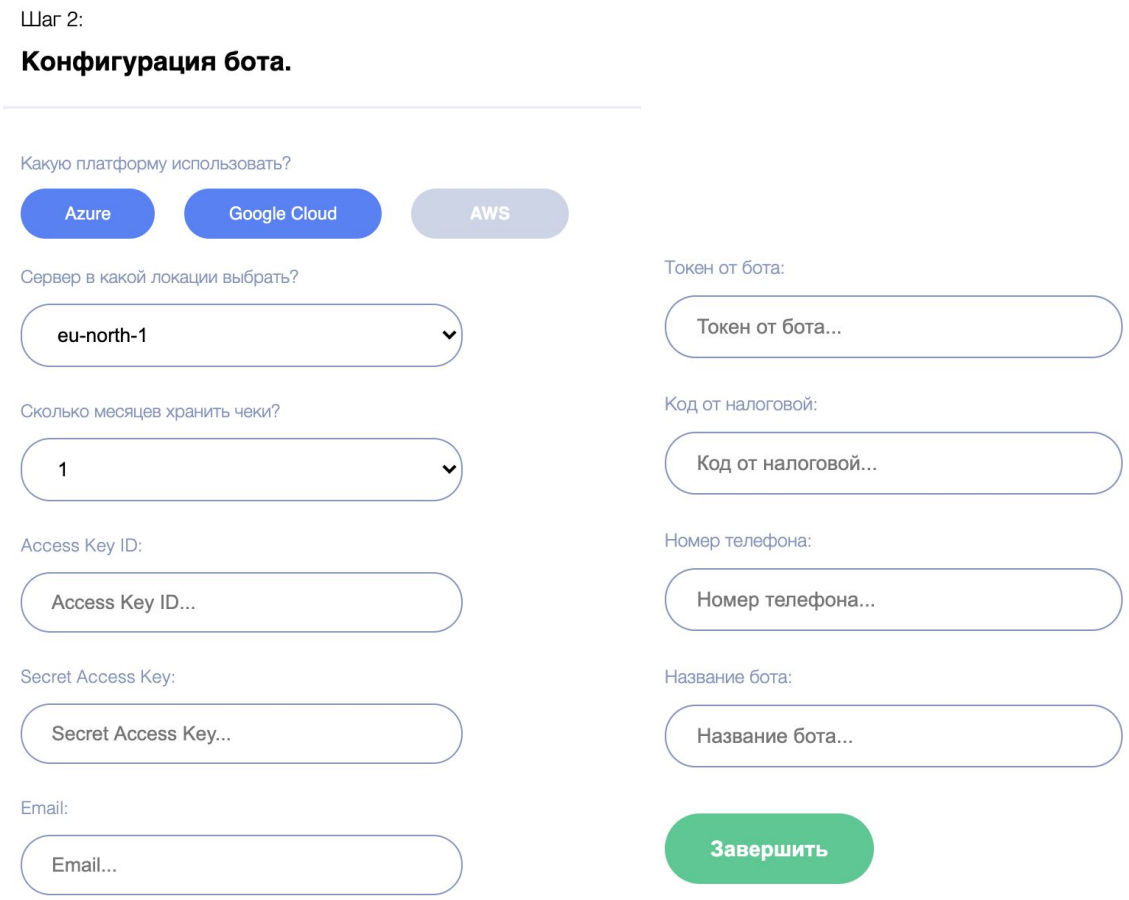

Рис.4,5 Второй шаг интерфейса

На втором шаге пользователь производит конфигурацию чат-бота. Он выбирает платформу, после чего предоставляет некоторую конфигурацию чат-бота, а также добавляет информацию нужную для генерации и деплоя кода на аккаунт пользователя, например код доступа, который разный у каждого провайдера.

2.5 Проблема раздробленности провайдеров

Одна из главных отличительных черт разработки будет генерация чат-бота сразу на нескольких платформах: AWS lambda, Google Cloud Functions и Azure Functions.

Одним из главных недостатков serverless сейчас является vendor-lock. Если вы используете серверное предложение какого-то поставщика облачных услуг (например, AWS Lambda, Azure Functions или Google Cloud Functions) и решите перейти на другого CSP, вам, вероятно, придется внести серьезные изменения в базу кода. И это может занять много времени и денег.

Для этого есть несколько причин. Компания, которая будет пользоваться чат-ботом, может уже использует одну из платформ и ей хотелось бы чат-бота держать на удобной им платформе. Также из-за нестабильности политической ситуации, один из сервисов может перестать работать в стране. Причиной другого типа является попробовать возможно ли генерировать код для всех трех платформ сразу и насколько это имеет смысл.

В генерации кода для нескольких платформ имеет ряд сложностей, все из которых исходят из ситуации на рынке serverless. Так как технология бурно развивается и нет стабильного лидера, каждая компания создает свои стандарты и функционал, который зачастую сильно отличается от представлений от другой платформы.

Во время работы на над чат-ботом, мы поняли, что функция выбора региона сервера будет полезной.

Регион - это географическая область, в которой провайдер serverless имеет центры обработки данных. Каждый регион имеет две или более зон доступности, которые представляют собой независимые центры обработки данных, расположенные близко друг к другу. Зоны доступности используются для резервирования, а также для репликации данных.

Довольно очевидным является тот факт, что для такой большой страны как Россия, имеет смысл выбирать локацию сервера для более быстрого отклика и скоростей передачи. Ниже приведена таблица с время отклика в миллисекундах.

FROM	N. Virginia	Ohio	Oregon	N. California	Sao Paolo	Ireland	Frankfurt	Mumbai	Tokyo	Seoul	Singapore	Sydney
N. Virginia	2	26	323	149	257	140	169	423	291	340	522	449
Ohio	23	4	487	101	299	159	189	592	267	316	481	408
Oregon	159	133	211	41	349	247	280	515	178	228	409	307
N. California	147	101	375	2	376	289	300	494	215	261	384	310
Sao Paolo	261	299	427	376	2	390	383	603	508	567	756	677
Ireland	139	161	449	287	381	1	44	266	415	464	535	590
Frankfurt	170	187	470	301	377	43	5	233	434	493	675	600

Рис.6 Скорость откликов разных локациях AWS

Но это не единственная причина, которую нужно рассмотреть. В последнее десятилетие мы наблюдали, что в область ИТ активно играет роль гео-политика. Это опять же справедливо в случае России.

Иногда, из-за нормативных причин, мы будем вынуждены выбрать конкретный регион из-за изменений в существующих законах, которые указывают где можно хранить определенные данные. Но есть много других факторов, которые необходимо учитывать при выборе региона.

Другая выбора локации причина - это факт того, что стоимость услуг изменяется в зависимости от локации. Ниже приведен график цен в разных регионах для AWS.

Рис.7 Стоимость в разных локациях AWS

Также в дальнейшем предполагается улучшать и добавлять дополнительные возможности чат-боту, например предложения для разработки машинного

обучения. Оказалось, что не все предложения от провайдеров предоставляются во всех регионах. Поэтому это добавило еще одну причину для выбора провайдера.

3 СОЗДАНИЕ УМНЫХ КОМАНД

3.1 Прогнозирование стоимости поездки

Главной умной функцией чат-бота должен стать прогноз примерной стоимости поездки. Пользователь должен ввести данные по поездке, города, которые собирается посетить, в течении какого времени он/она собирается там бы. Чат-бот должен выдать примерную стоимость этой поездки.

Такая функция будет полезна по нескольким причинам. В случае, если поездка длиться довольно долгое время, может появиться нужда выдать деньги заранее. Также такой прогноз может позволить проанализировать насколько добросовестно или эффективно работник потратил ресурсы. Прогнозирование в таких случае будет полезным инструментом.

На данный момент есть несколько способов, чтобы понять сколько будет стоит поездка. Самый очевидный - это понять просто предположить, зная цены на рынке и примерную разницу в городах. У этого способа есть много минусов: не включает меняющиеся факторы, не точен, не знание информации по некоторым городам. Также существуют различные сервисы, которые предоставляют информацию по стоимости жизни. Например сервис Numbeo предоставляет информацию по многим городам по стоимости аренды жилья, стоимости различных продуктов(молоко, бензин), транспорта и других характеристик расходов. Он мало применим в случае проверки стоимости поездки: не предоставляется информация по отелям, необходимо рассчитывать все вручную и зачастую информация по более малым городам отсутствует, особенно в России. Существуют более удобные сервисы, но для запросов к их API они требуют довольно большое количество денег, так как их целевая группа клиентов - это корпорации. Также такие сервисы не предназначены именно для

поездок, поэтому не учитывают многие переменные факторы, такие как праздники, когда цены увеличиваются в особенности на аренду.
После анализа, было выявлено два способа по прогнозированию стоимости поездки. Первый - это создание модели с помощью машинного обучения. Второй - это алгоритм, которая достаточно точно определяет стоимость поездок в разные города.

3.2 Разработка модели по вычислению стоимости командировки:

Для начала рассмотрим способ с машинным обучение. Первым шагом, необходимо было найти данные для обучения модели. Так как целевые пользователи находятся в России, готовые данные довольно сложно найти. Поэтому необходимо собрать данные самому вручную. Выбор параметров для данных состоит из небольшого количества типов данных. Город имеет несколько основных характеристик: месторасположение, население, площадь, уровень и тип экономики. Этих данных недостаточно, поэтому были добавлены несколько других характеристик для городов. Стоимость жизнь в определенном месте связана со спросом, то есть желанием большого количества людей жить там. Климат является важным аспектом для выбора города для жизни, поэтому имеет смысл его добавить. Также можно попробовать использовать индекс счастья, который может указать на скрытую связь. Например в более богатых и напряженных городах люди могут чувствовать себя хуже, по сравнению с спокойными городами. Единственным способ, для нахождения всей этой информации, это поисковая система. Это довольно большая проблема, так как сбор большого количества данных, нужный для хорошо-работающей модели, займет очень большое количество времени. Это один из важных минусов такого подхода.
После сбора данных получилась таблица, ее часть:

long	lat	population	area	economy	weather	happiness_index	target	cityId
56	38	11000	2510	588729	14	40	88000	moscow
60	30	5000	1440	204871	14	46	70000	peterburg
55	83	1500	500	53803	4	54	53000	novosibirsk
49	45	1000	860	35451	18	35	49000	volgograd
43	46	250	320	7832	20	60	37000	groznii
55	40	500	220	17718	12	45	48000	ryazani
55	60	520	220	48912	3	36	45000	chelyabinsk
43	132	600	330	27787	6	36	56000	vladivostok
55	86	540	410	19405	3	35	44000	kemerovo
55	48	180	120		10	33	38000	syzran
53	50	1200	540	44956	14	40	50000	samara
51	36	370	150	16406	16	52	32000	belgorod
49	45	700	1200	73903	28	55	54000	krasnodar
44	40	360	900		29	54	65000	sochi
48	40	230	250		22	33	32000	shahti
47	40	1100	360	50647	23	44	52000	rostov

Рис.8 Информация по городам

Для обозначение расположения используются широта и долгота, это позволит универсально обозначать местоположение городов, а также возможны некоторые скрытые связи, например влияние долготы из-за лучшего климата. Happiness-index был взят из исследования мониторингового агентства NewsEffector и фонда региональных исследований «Регионы России». ВВП показатель доступен только для более больших городов. Так как в данных мало переменных, каждая из них важна, поэтому отсутствующие значение нужно наилучшим образом заменить. В случае ВВП возможно получить значения исходя из соотношения данных известного города к изменяемому: targetCityPopulation / closestCityPopulation * GDPClosest = GDPtarget, где closestCityPopulation - население города с ближайшим количеством людей, targetCityPopulation - население искомого города, GDPClosest - ВВП ближайшего по населению города, GDPtarget - искомое ВВП.

Получаемый результат не идеален, но позволяет с грубой точностью оценить ВВП неизвестного города.

Как видно на гистограмме ниже, большинство городов входят в одну группу цен. Это происходит, потому что данные, которые удалось получить подходят только для более больших городов. Это еще один минус использования машинного обучения для высчитывания стоимости поездки.

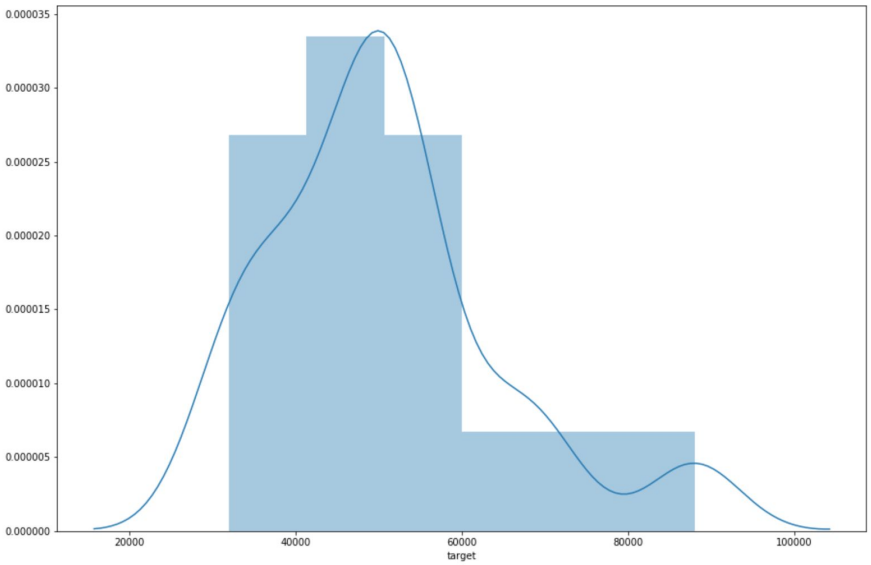

Рис.9 Гистограмма стоимости проживания

Так как задача построить модель, которая будет предсказывать число, можно выбрать несколько регрессионных алгоритмов. Для обучения моделей было выбрано два регрессионных алгоритма: LassoCV и Linear Regression.

Модель сначала была протестирована на Linear Regression. Деление на тренировочные и тестовые данные производились в соотношении 5 к 1.

long	lat	population	area	economy	weather	happiness_index		
1.6213	5.5174	2.4335	8.4007	-3.9787	2.0389	2.4227		

Рис.10 Коэффициенты от Linear Regression

По коэффициентам стало видно, что нет параметра, который сильно влиял бы на стоимость проживания. Только площадь и широта имеют значительное влияние.

Predicted	Actual
79010	70000
25384	45000
43310	44000
30267	38000

Рис.11 Результат от Linear Regression

Первые два города модель предсказала с большой ошибкой, в то время как последние два с высокой или достаточной точностью.

long	lat	population	area	economy	weather	happiness_index		
1.6306	5.6158	2.4431	8.3962	-3.9772	2.0371	2.4232		

Рис.12 Коэффициенты от LassoCV

Predicted	Actual
78005	70000
25491	45000
43411	44000
30369	38000

Рис.13 Результат от LassoCV

LassoCV не показал значимых различий.

Неточное прогнозирование обеих моделей связано с малым количеством данных. Из этого следуют главные минусы построения модели в данном случае:

- Отсутствие готовых данных, требуется огромное количество времени для сбора данных.
- Отсутствие данных для более малых городов.

- Не включает в себя стоимость транспорта во время поездки по нескольким городам.

Если была бы возможность получить данные для большого количества городов, модель работала бы лучше.

3.3 Составление и использование алгоритма

Оценить стоимость проживания возможно используя алгоритм.

Цель рассчитать стоимость проживания, повседневные затраты и проживание в отеле, а также транспорта между городами.

Для большинства городов России стоимость проживания связана с несколькими характеристиками города: население, близость к другим большим городам. Эти данные, население и локация, собираются через MediaWiki. MediaWiki предоставляет api для сбора информации с страницы википедии. Население и локация, в форме координат, доступна для большинства городов.

Для рассчитывания стоимости командировки для любых городов, сначала нужно базовое значение, которым будет один из городов. В качестве такого города подойдет Москва, так как цены здесь составят планку от которой другие города будут считаться. Далее была собрана информация по стоимости жизни в Москве на различных ресурсах. Стоимость повседневных затрат была взята и ресурса Numbeo, стоимость проживания в отеле в сервисе booking. Средняя цена отеля составила 4000 руб/день, стоимость проживания составила 900 руб/день.

Для высчитывания стоимости жизни в других городах, производится их нахождение в близости больших городов россии. В эту групп входят Волгоград, Краснодар, Санкт-Петербург, Уфа, Пермь, Екатеринбург, Тюмень, Омск, Новосибирск, Красноярск, Иркутск, Владивосток и другие города. Согласно

сервису Numbeo, в этих городах в среднем стоимость проживания на 30% ниже, чем в Москве, а стоимость отелей на 50% ниже.

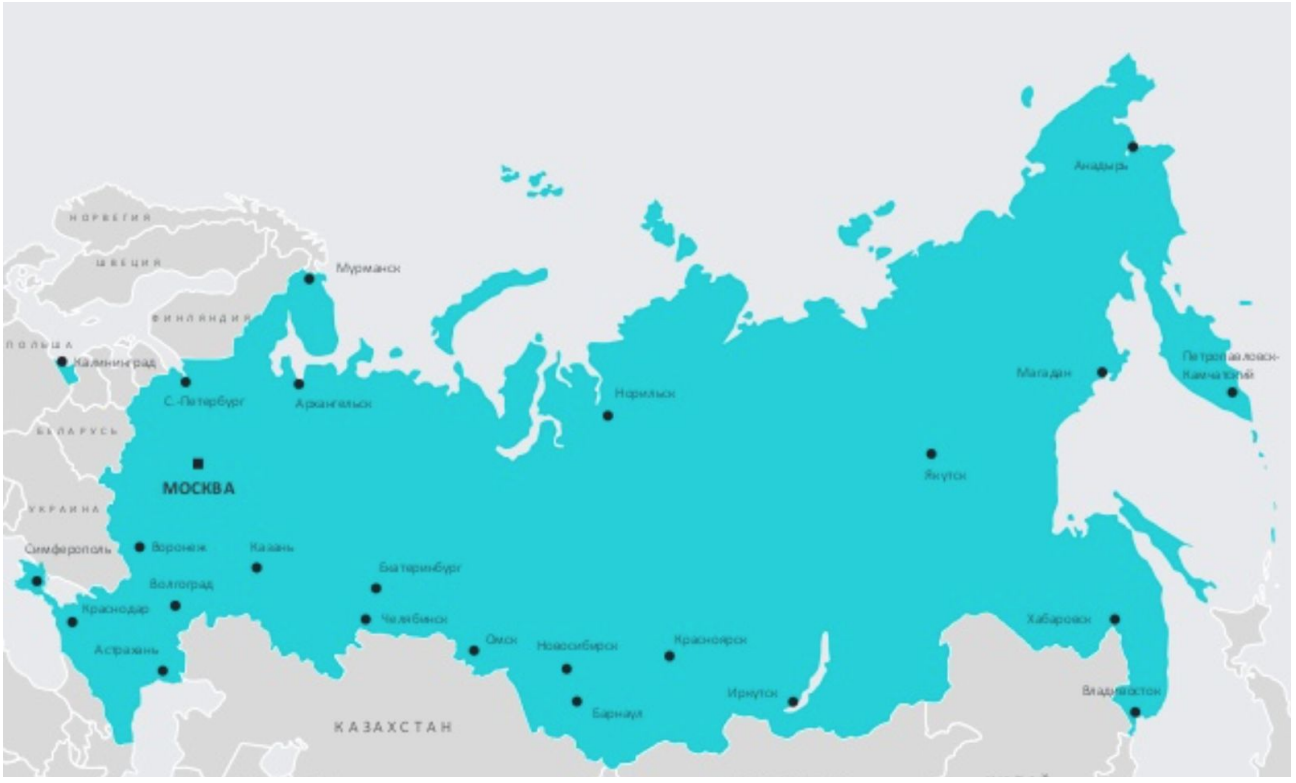

Рис.14 Выбранные города

На рисунке обозначены часть городов, которые вошли в список.

Далее был проведен анализ как быстро стоимость жизни в других городах уменьшается с увеличением расстояния до больших городов. Также, в городах далеких от больших городов, в среднем стоимость проживания на 20% ниже и стоимость отелей на 30% ниже.

Был проведен анализ нескольких городов вокруг Санкт-Петербурга и Екатеринбурга, как их цены уменьшаются с увеличением дистанции. Цены перестают видимо увеличиваться на расстоянии в 100 км. Поэтому алгоритм учитывает, если город находится в этом радиусе от большого города.

Для городов, которые не находятся вблизи больших городов, были установлены средние значения. Стоимость жизни в день: 300 руб, в отеле 1300 руб.

Также нужно учесть увеличение стоимости проживания в отелях во время праздников. Для этого были собраны российские праздники, во время значительно увеличивается стоимость отелей: новый год, рождество, пасха и другие. С помощью сервиса booking.com были проверены даты во время праздников. В среднем цены увеличивались на 25%

Стоимость использования транспорта также рассчитывается.
Для рассчитывания стоимости поездки необходимо знать стоимость за 1 км. Также нужно знать, если стоимость варьирует от региона к региону. Используя сервисы tutu.ru, для поездов, и aviasales.ru, для самолетов, была рассчитана средняя цена за 1 км пути для каждого вида транспорта. Расчеты были проведены для средней стоимости поездов и авиалиний. Также цены не варьировали сильно от региона к региону. В среднем, поездка на поезде стоила 1.9 руб/км, для авиаперелета этот показатель составляет 2 руб/км. Разница незначительная, поэтому было решено не разделять вычисления по виду транспорта и установить 2 руб/км.
Дистанция рассчитывается, используя координаты городов:

```
dlon = lon2 - lon1
dlat = lat2 - lat1
a = (sin(dlat/2))**2 + cos(lat1) * cos(lat2) * (sin(dlon/2))**2
c = 2 * atan2(sqrt(a), sqrt(1-a))
distance = R * c
```

Команда для начала маршрута выглядит следующим образом:
“Оценить: Омск//Казань2/Москва3/Грозный1”, где цифра указывает на количество дней, которые планируется провести в данном городе.
Данная строка парситься. На первом шаге для каждого города расчитывается стоимость проживания. Для этого происходит обращение к WikiMedia api, для

получения населения каждого города и координат. После чего вычисляется по алгоритму стоимость жизни для каждого города. На последнем шаге вычисляется суммарное расстояние между городами с помощью метода, упомянутого выше. Результат умножается на 2. После чего пользователю отправляется сообщение с прогнозом стоимости.
Пример команды: “Оценить: Казань//Омск2/Кормиловка2/Москва2”. Выдал результат 28000 руб.

3.4 Распознавание необычных покупок

Задача распознавать необычные покупки.

В банковских системах необычные покупки базируются на истории клиента, для чего используются модели машинного обучения. В нашем случае такое не возможно, так как у нас нет истории клиентов.
Также нахождение покупок, которые не имеют смысла для покупки в командировке, например мягкая игрушка, не представляется возможным. Это связано с тем, что имена товаров магазины по-разному пишут и часто это сокращения, состоящие из 1-2 букв.
После анализа было найдено и сделано два способа нахождения необычных покупок:

- Базовая трата за день считается, если покупка превышает 60% этого, то покупка фиксируется как необычная.
- Куплено большое количество одного товара, больше 10.

Если есть необычные покупки, они указываются в отчете на дополнительной странице.

ЗАКЛЮЧЕНИЕ

Был создан генератор чат-бота, который автоматизирует работу чеками и генерацию авансовых отчетов. В течении этой работы был найден метод получения информации по покупкам по чеку. Для этого нужно было отсканировать qr код с фотографии, получить доступ к серверам налоговой и после этого получать информацию на запросы по чекам. Был создан генератор отчетов, который выводит отчеты в формате pdf в виде таблицы.

Для реализации чат-бота были выбраны технологии бессерверных вычислений. Это позволило существенно сократить расходы, для малых и средних предприятий чат-бот является практически бесплатным.

Была создан способ для генерации чат-бота сразу на трех платформах провайдеров бессерверных вычислений: Azure, AWS, Google Cloud. Найденный метод позволяет создавать один код и с минимальными изменениями деплоить его на одну из платформ. Был создан пользовательский интерфейс, который позволяет создавать пользователю производить конфигурацию чат-бота, например указать локацию сервера, что имеет смысл принимая во внимание размер страны. Был создан процесс для изменения конфигурации платформы бессерверных вычислений и кода.

Также были добавлены умные функции, которые добавляют дополнительные полезные функции чат-боту. Команда, позволяющая рассчитать примерную стоимость поездки, была создана. Были рассмотрены два способа реализации: с помощью машинного обучения и с помощью алгоритма. Был выбран способ с использование алгоритма, который основан на данных собранных с таких ресурсов как booking.com, Numbeo, близости городов к большим городам и

населении города. Также был создан функционал по выявлению необычных покупок.

Есть много перспектив развития. Одна из них развитие системы, чтобы она работала для учета любых выданных авансов сотрудникам: на оплату канцтоваров в офис, на представительские расходы, на оплату поставщикам. Также умные команды можно развить во многом: улучшить систему рассчитывания стоимости поездки, которая будет в начале также работать на основе алгоритма, но со временем и увеличением количества данных будет использовать машинное обучение.

Исходный код работы размещен по ссылке:
https://github.com/trashgrandterr/Diplom2020

СПИСОК ЛИТЕРАТУРЫ

1. The Multi-Provider Future of Serverless Application Development [Электронный ресурс] // Сайт serverless. Режим доступа: https://www.serverless.com/blog/multi-provider-serverless-video/ (дата обращения: 12.04.2020).
2. Baldini, Ioana, et al. "Serverless computing: Current trends and open problems." Research Advances in Cloud Computing. Springer, Singapore, 2017. С. 1-20.
3. Lehvä, Jyri, Niko Mäkitalo, and Tommi Mikkonen. "Case study: building a serverless messenger chatbot." International Conference on Web Engineering. Springer, Cham, 2017. С. 8-33
4. Create and read QR code with Python [Электронный ресурс] // Сайт medium. Режим доступа: https://medium.com/@stasinskipawel/create-and-read-qr-code-with-python-in-3-minutes-opencv-and-qrcode-38ecc3a6258a (дата обращения: 14.04.2020).
5. Самые дорогие города в России [Электронный ресурс] // Сайт uznayvse. Режим доступа: https://uznayvse.ru/interesting-facts/samyie-dorogie-goroda-v-rossii.html (дата обращения: 18.04.2020).
6. Универсальный API для получения информации по чекам [Электронный ресурс] // Сайт habr. Режим доступа: https://habr.com/ru/post/358966/ (дата обращения: 05.04.2020).
7. What is Serverless Architecture? What are its Pros and Cons? [Электронный ресурс] // Сайт freecodecamp. Режим доступа: https://www.freecodecamp.org/news/what-is-serverless-architecture-what-are-its-pros-and-cons/ (дата обращения: 01.04.2020).

8. Bashing the Bash — Replacing Shell Scripts with Python [Электронный ресурс] // Сайт medium. Режим доступа: https://medium.com/capital-one-tech/bashing-the-bash-replacing-shell-scripts-with-python-d8d201bc0989 (дата обращения: 08.05.2020).
9. Generating pdfs with ReportLab [Электронный ресурс] // Сайт medium. Режим доступа: https://medium.com/@vonkunesnewton/generating-pdfs-with-reportlab-ced3b04aedef (дата обращения: 20.04.2020).
10. Ижевск вошел в Топ-50 городов России по уровню счастья [Электронный ресурс] // Сайт udmurt. Режим доступа: https://udmurt.media/news/obshchestvo/17343/ (дата обращения: 03.05.2020).
11. MediaWiki документация [Электронный ресурс] // Сайт mediawiki. Режим доступа: https://www.mediawiki.org/wiki/API:Query (дата обращения: 05.05.2020).
12. Amazon DynamoDB Tutorial – A Complete Guide [Электронный ресурс] // Сайт edureka. Режим доступа: https://www.edureka.co/blog/amazon-dynamodb-tutorial?ranMID=42536&ranEAID=a1LgFw09t88&ranSiteID=a1LgFw09t88-7WGLnZL1FboyzOs0tppPJw&LSNSUBSITE=Omitted_a1LgFw09t88&utm_source=Optimise&utm_medium=CPS_Campaign_live-sp_213232_20200528cl95li4kqtyp&utm_campaign=April_2020 (дата обращения: 14.04.2020).
13. AWS Lambda Language Comparison: Pros and Cons [Электронный ресурс] // Сайт epsagon. Режим доступа: https://epsagon.com/blog/aws-lambda-programming-language-comparison/ (дата обращения: 14.04.2020).

14. AWS Lambda Language Comparison: Pros and Cons [Электронный ресурс] // Сайт nycdatascience. Режим доступа: https://nycdatascience.com/blog/student-works/cost-of-living-estimator/ (дата обращения: 14.04.2020).
15. Using Machine Learning to Predict Home Prices [Электронный ресурс] // Сайт towardsdatascience. Режим доступа: https://towardsdatascience.com/using-machine-learning-to-predict-home-prices-d5d534e42d38 (дата обращения: 14.04.2020).