Федеральное государственное бюджетное образовательное учреждение   
высшего образования

|  |  |
| --- | --- |
| Gerb-BMSTU_01 | «Московский государственный технический университет  им. Н.Э. Баумана (национальный исследовательский университет)» (МГТУ им. Н.Э. Баумана) |

ФАКУЛЬТЕТ – Информатика и управления

КАФЕДРА – Информационные системы и телекоммуникации

Курсовая работа

на тему

Разработка системы управления терминалами

Студент группы ИУ3-71 (подпись) 17.09.2021 А.Т. Марчук

Руководитель курсовой работы (подпись) 17.09.2021 И.М. Сидякин

Москва, 2021

Содержание

Введение

На сегодняшний день остро стоит вопрос безналичных платежей. С каждым годом их востребованность растет, как и растет потребность в системах удаленного контроля, мониторинга и наладки для терминалов безналичных платежей. Для упрощения данного процесса осуществляют разработку так называемых TMS – Terminal Management System – системы управления терминалами. Данные системы должны осуществлять мониторинг состояния терминалов и позволять удаленно настраивать их.

На сегодняшний день существует множество различных систем управления терминалами от различных производителей. Возникает закономерный вопрос – так ли остро стоит вопрос разработки еще одной? Проблема заключается в том, что имеет место быть сильная зависимость между аппаратной составляющей терминала и тем, какая система управления может быть к нему применена. Потребность разработки собственной системы управления обусловлено несовместимостью изготавливаемых терминалов с теми системами управления, которые в настоящий момент есть на рынке. Однако для понимания характерных признаков данных систем стоит привести выборку из нескольких, наиболее популярных, и проанализировать их преимущества и недостатки.

1. Исследовательская часть

Настоящий раздел является теоретическим и включает:

* анализ требований технического задания;
* описание предметной области разработки;
* анализ существующих решений в данной предметной области;
* выбор и обоснование технического решения.
  1. Анализ требований технического задания

/\*заполню после того, как непосредственно получу задание\*/

* 1. Описание предметной области разработки

Введем несколько определений и аббревиатур для дальнейшего использования.

EMV – Стандарт взаимодействия чиповых карт для проведения платежей по карте

EMV приложение – Микропрограмма и блок параметров на карте, относящейся к международной платежной системе EMV

ПИН-пад – Электронное устройство, предназанченное для ввода PIN-кода

PIN код – Секретный код, вводимый клиентом, для подтверждения того, что он обладает полномочиями по использованию карты

Терминал – Аппаратно-программный комплекс, предназначенный для регистрации факта приобретения товара и упрощения процедуры приобретения товара

Код действия терминалов (TAS) – Действия ПО терминала при возникновении конкретной ситуции (отказ от операции, проведение операции онлайн или действие по умолчанию)

Коммуникационный скрипт – Минипрограмма (последовательность команд), определяющая сценарий подключения терминала к серверу авторизации или серверу TMS

Клиент/Держатель – Физическое лицо (в том числе уполномоченный представитель юридического лица), на имя которого в силу заключенного между ним и Эмитентом договора и/или на имя которого по указанию лица, заключившего договор с Эмитентом, выпущена Карта.

Магнитная полоса - Полоса, содержащая идентификационную информацию о карте и ее держателе.

Операция - Расчетная операция, осуществляемая с использованием Карты с обязательной Авторизацией в соответствии с законодательством Российской Федерации, нормативно-правовыми актами Банка России и договором с Эмитентом, проводимая в Торговой точке по требованию Держателя.

Операция возврата - Расчетная операция, совершаемая с использованием Карты при возникновении необходимости в возврате Держателю денежных средств, списанных с его счета в рамках ранее совершенной Операции.

Операция отмены - Процедура отмены Кода Авторизации до формирования Электронного журнала.

Электронный журнал - Документ или совокупность документов в электронной форме, сформированные за определенный период времени при совершении Операций/Операций с использованием POS-терминала.

Программное обеспечение (ПО) - Программное обеспечение POS-терминала.

Профиль - Один из объектов системы, предназначенный для хранения настроек групп параметров, определяющих алгоритм работы терминала в процессе выполнения операции с картой или административной операцией.

Реквизиты Карты - Номер Карты, срок действия Карты, код безопасности CVV2/CVC2/Batch code/СVV II/CAV2 (для Карт Платежных систем Visa International/MasterCard Worldwide/American Express/Diners Club International/JCB International соответственно).

Сервисный код карты - Код карты, который указывается область ее действия и тип.

Сервисный код коммерсанта (MCC) - Четырехзначный номер, который обозначает профиль деятельности, которым занимается коммерсант.

Скрипт эмитента - Команда или командная строка, передаваемая от эмитента к терминалу с целью выполнения команд(ы) микропроцессорной картой.

Способ верификации владельца карты - Способ верификации того, что человек, предъявляющий карту, имеет права на ее использование (например, сверка подписи или введение PIN кода).

Справочник - Один из объектов системы, предназначенный для хранения основных

значений, шаблонов, терминов и алгоритмов, которыми оперирует система.

Транзакция - Группа последовательных операций, которая представляет собой логическую единицу работы с данными. Транзакция может быть выполнена либо целиком и успешно, соблюдая целостность данных и независимо от параллельно идущих других транзакций, либо не выполнена вообще и тогда она не должна произвести никакого эффекта. Транзакции обрабатываются транзакционными системами, в процессе работы которых создаётся история транзакций.

Хост, Коммуникационный сервер - Сервер базы данных платежной системы, который отвечает на запросы POS-терминала точки обслуживания, а также снабжает его необходимой информацией.

Эквайер - Банк или компания, осуществляющая весь спектр операций по взаимодействию с точками обслуживания карточек, которая состоит из терминалов в торгово-сервисной сети и банкоматов.

Эмитент - Организация, выпустившая (эмитировавшая) карточки для развития и

финансирования своей деятельности

AID - Application Identification Number (Идентификационный номер приложения)

AVN - Application Version Number (Номер версии приложения)

BIN - Bank Identification Number (Банковский идентификационный номер)

CAP - Cash register Application

CVM - Cardholder Verification Method (Способ верификации владельца карты)

DDOL - Dynamic Data Object List (Список элементов данных необходимых для динамической аутентификации)

ICC - Integrated Circuit(s) Card (Карта с интегрированными электронными схемами)

ID - Identifier (идентификатор)

ISO - International Organization for Standardization (Международная организация стандартизации)

MCC - Merchant Category Code (Сервисный код коммерсанта)

MS - Magnetic Stripe (магнитная полоса)

TAC - Terminal Action Code(s) (Код(ы) действия терминалов)

PAN - Primary Account Number (Индивидуальный номер карты)

PIN - Personal Identification Number (Персональный идентификационный номер)

POS - Point of Sale (точка продажи, сервиса)

RID - Registered Application Provider Identifier (Зарегистрированный номер провайдера приложения; например, VISA, MasterCard)

TDOL - Transaction Data Object List (Список данных о транзакции участвующих при вычислении сертификата)

TMM - Terminal Management Mode (модуль управления терминалами)

TMS - Terminal Management System (система управления терминалами)

СУБД - Система управления базами данных

Современные POS-системы представляют собой комбинацию программных и аппаратных решений, позволяющих проводить платежные операции и облегчающих ежедневные бизнес-процессы. Говоря о POS-ах, обычно имеют в виду кассовые аппараты, терминалы оплаты и другие привычные составляющие торговых магазинов. Однако, архитектура POS не ограничивается только этими элементами. Пример полной процедуры транзакции приведен на рисунке 1.



Рисунок 1 – пример полной процедуры транзакции

Сначала держатель проводит картой по считывателю. Данные кредитной карты поступают в терминал, откуда отправляются в POS-систему. Далее, POS-система связывается сервисом обеспечения платежей, который, в зависимости от типа кредитной карты, обращается в банк для прохождения процедуры авторизации транзакции. В этот момент держателю предлагается ввести PIN-код для подтверждения транзакции. Если все прошло успешно, код авторизации возвращается из банковской сети в PSP и передается в POS-систему и терминал.

В рамках данной дипломной работы будет осуществены анализ и доработка системы, осуществляющей работу с серверной частью По и позволяющей осуществлять мониторинг и конфигурирование удаленных терминалов.

* 1. Анализ существующих решений в данной предметной области

Для анализа было выбрано 3 реализации TMS – XENTURION, OtiTms и ARCUS2. Ввиду невозможности исследования проприетарного исходного кода, рассмотрим список особенностей, которые они предоставляют. Последовательно проанализируем документацию каждой из систем управления терминалами, выделяя отличительные черты.

XENTURION предлагает следующий функционал:

* Удаленное управление терминалами
* Организация и настройка набора терминалов
* Использование мощного поискового движка для установления работающего списка терминалов
* Просмотривать статуст терминалов
* Производить обновления в 1 клик
* Просматривать историю действий
* Загружать прозвольное программное обеспечние в терминалы
* Планировать умные и динамические обновления в нерабочие часы
* Управлять загрузками для различных семесйств терминалов
* Просматривать прогресс обновлений
* Загружать и выгружать любой параметризованный объект с терминала на сервер
* Доставлять начальную конфигурацию на сервер
* Разделять параметры между группой терминалов
* Просмотреть обновления парметров
* Устанавливать повестку для синхронизации сервисов
* Планирование включений и отключений
* Ограничивать доступ для партеров с ограниченным допуском
* Просмотривать историю для каждого из пользователей
* Упрощение удаленное распределение ключей и сертификатов для терминалов
* Просматривать историю изменения ключей

В то же самое время, список преимуществ OtiTms следующий:

* Управление финаисовой, технической и продуктовой информцией
* Мониторинг аппаратной материальной базы
* Детализированный технический мониторинг
* Настраиваемое расписание уведомлений
* Мониторинг и оповещение продаж, как наличных, так и безналичных
* Торговая аналитика по каждому клиенту/местоположению/терминалу
* Удаленная смена настроек конфигурации.

Список отличительных черт ARCUS2:

* Высокая скорость совершения транзацкций
* Длительная наработка на отказ
* Работа в режиме многозадачности
* Стандартная поддержка не только магнитных лент, но и бесконтактных карт
* Множественность каналов связи для подключения

Резюмирую, можно сказать, что для TMS являются ключевыми следующие параметры:

* Возможность удаленного мониторинга и настройки
* Высокая степень оптимизации
* Возможность контроля доступа и логгирования
  1. Выбор и обоснование технического решения

В рамках данного проекта мною будет разрабатываться веб-приложение с использованием фреймсмворка для серверной разработки Spring Boot, фреймворка для разработки клиентской части ReaсtJS, а так же с использоваием СУБД MySQL. Данный выбор был обусловлен тем, что данные решения являются востребованными на рынке в настоящее время, хорошо масштабируются и подходят для решения поставленных задач. Кроме того, определенная доля разрабатываемой системы управления терминалами была разработана предварительно с использованием данного перечня технологий, ввиду чего мной было принято решение оставить данный стед до завершения проекта. Кроме того, на финальных этапах разработки потребуется сервис для помещения всех терминалов в одну локальную сеть. Для этого будет использоваться wireGuard.

Конструкторская часть

В рамках данной курсовой работы были предприняты следующие действия:

Разработана система логгирования событий, расширяющая и дополняющая существовашую логгирующую систему:

Переработна сисьема получения терминалам информации об изменении конфигурационных файлов, а также доработана система сбора статичстики, позволяющая получать данные не по инициативе оконечного терминала, а посредством запроса с сервера, что существенно расширяет существующий фкункционал.

Разработан файл конфигурации образа вирутальной машины, позволяющий быстрои кроссплатформенно разворачивать сервер с программным продуктом, а также использовать специализированные решения для автоматической балансировки нагрзуки в высоконагруженных системах.

Разработано кросплатформенное приложение для мобильных устройств, служащее для тестирования и отладки серверной части приложения, а также – имеющее потенциал для последующего преобразования в мастер-терминал.

Ниже будет рассмотрен каждый из вышеописанных этапов.

Система логгирования событий.

Ввиду того, что основное ядро программного продукта написано в виде монолитного сервиса на языке Java, для снижения сложности поддержки системы в целом, упрощении ее дальнейшей поддержки и возможности постепенного перехода на микросервисную архитектуру, логгирование осуществляется посредством написания отдельного программного продукта, придерживающегося паттерна построения «микросервис». Данный микросервис осуществляет только одну задачу – принимает запрос от некоторого сетевого хоста и обрабатывает его. В частности, в роли сетевого хоста выступает сервер фронтенда, а в качестве результата обработки запроса можно назвать добавление новой записи в таблице логов в базе данных. Таблица логов в базе данных не является сильно связанной с остальными таблицами базы данных. Поэтому добавление в нее записей можно вынести в отдельную зону ответственности, что и было сделано путем разработыки микросервиса логгирования.

В качестве языка разработки для данного микросервичса был выбран NodeJS. Причиной этому послужили множественные фоакторы, такие как простота разработки и поддержания кода, написанного на данном языке, тот факт, что NodeJS является актуальным и динамично развивающимся языком, который нашел свое применение в множественных отраслях разработки серверных приложения, а также крайне большой ассортимент необходимых библиотек. В частности, библиотека mysql2 позволяет гибко настраивать произвольные запросы к базе данных посредством инъекций переменных в предопределенные структуры на языке SQL, избегая излишних абстракций, что, в свою очередь, приводит к повышению производительности, а также позволяет участвовать в разработке проекта лиц, в не зависимости от их умения использования языки серверной разработки, например, инженеров баз данных. Формат запроса к микросервису также претерпел изменения по сравнению с ядром. Если в основной программе серверного приложения используется устаревший формат передачи данных XML, то в микросервисах, дописываемых поверх него, вся логика взаимодействия микросервисов построена на использовании формата JSON, больше распространенного в современной WEB разработке. Конечно, стоит отметить, что парсинг данных не производится вручную – анализирование поступающей информации осуществляется с помощью набора из нескольких специализированных библиотек, которые позволяют абстрагироваться от сетевого уровня действия протоколов и работать с поступающими данными так, словно они поступают не с терминала, удаленного, возможно, на многие и многие мили, но - хранятся на том же устройстве, что и код. Их обрабатывающий.

Однако, помимо общего описания стека технологий, требуется также описать логику обработки данных, использующуюся в данном микросервисе. В рамках данного микросервиса происходит создание соединения с базой данных, опеределяюшей параметры подключения, такие как используемая учетная запись, пароль и хост. В общем случае, подобные данные должны быть получаемы из переменных окружения, однако в демонстрационных целях в рамках данной курсовой работы было принято решение вставить их непосредственно в текст программы.

Затем определяется особый объект, называемый приложением. От определяет используемый порт – в данном случае был выбран порт 8081 – используемый тип кодирования и утилиту для расшифорвки JSON, текущую политику защиты от междоменных запросов и подобные параметры. Затем для данного приложения опеределяется один обрабатываемый путь - в частности, это приложение может принимать POST запросы с путем /api/log, причем данные запросы должны содержать в своем теле поля date, level, user и message. Стоит отметить, что поле data является опциональным, так как имеет определенный смысл использовать для логгирования не клиентское время пользователя, эксплуатирующее систему, а серверное время, единое для всех терминалов. С другой стороны, нельзя не отметить, что в терминале булл реализован определенный функционал, которые не будет позволять пользоваться сервером хостам, чье отставание или опережение по времени находится за пределами определенной, допустимой погрешности. Данный прием осуществляется главным образом из соображений безопасности, но нельзя не заметить, что корректность выставления времени на каждом из устройств является необходимым условием поддержания порядка в системе, что чрезвычайно важно для любого информационного продукта в целом и для системы управления банковскими терминалами в частности.

Параметр date определяет время, в которое произошло логгируемое событие. Level определяет уровень важности произошедгего события – оно может быть информационным, предупреждающим или критическим. Однако, стоит отметить, что, ввиду того, что поле, принимающее данный параметр, является строковым, множество различных уровней можно крайне существенно расширить. Параметр message определяет некоторое сообщение, которое может быть опционально переданно в систему и характеризующее произошедшее событие, напрер, авторизация пользователя, удаление, добавление или изменение каких либо записей. User определяет псевдоним пользователя. Чьи действия вызвали процедуру логгирования. В случае, если процедура логгирования ене была связана с пользователем, в данном поле предполагается указывать некоторое зарезервированное слово, например System.

Кроме того, в рамках данного микросервиса определена функция, которая принимает на вход вышеописанные параметры, преобразует их аргументы для команды INSERT языка SQL и передает в базу данных в виде строкового текста. Затем дожидается ответа, стрингифицирует его и передает в качестве ответного запроса на входящий запрос вопрошающей стороне.

Микросервис сокетов

Нельзя преумалять важность логгирования действий, происходящих в системе. Именно подробные логи позволяют анализировать происходящие в системе процессы, определить зоны ответсвенности польователей или – определить, что послужило причной той или иной ошибки или сбоя. Однако в рамках данной курсовой работы не меньшего внимания заслуживает и микросервис сокетов.

Данное называние не совсем корректно, так как оно в большей степени отражает технологию реализации процессов, осуществляемых данным сервисом, нежели задачи, которые данный сервис решает, однако ввиду отсутвия сколь-либо приемлимыхх альтернатив – данное название можно назвать подходящим, по крайней мере в рамках рабочей документации.

Для описания решаемых задач стоит рассмотреть два подхода к доставке информации на удаленные терминалы. Сразу стоит оговориться, что терминалы являются пассывными устройствами, они могу высылать запросы на сервер, однако не могут определять сложную логику рассылки данных запросов. С учетом вышесказанного, очевидными являются только 2 варианта получения терминалами информации – это либо регулярный опрос сервера, либо использование запросов, высылаемых с сервера. Первый вариант был реализован в ранных стадиях реализации программного продукта, однако оказался плох, ввиду своей низкой производительности, невозможности работать с большим количеством устройств, а также чрезмерным потребелением интернет-канала. Второй способ можно разбить на 2 варианта – можно либо использовать сервер на стороне терминала, посылая на него POST запросы, либо организовать прямое соединение по типу сокета. Каждый из них имеет свои преимущества и недостатки, однако соединение посредством сокета оказывается более предпочтительным, так как его преимущества являются более весомыми. В частности, оно потребляет меньше ресурсов на ртерминале, может быть активировано как терминалом, так и сервером и имеет крайне малое время отклика ввиду отсутствия необходимости установки соединения. С другой стороны, данный подход ограничивает количество подключаемых терминалов значением 65534 на один белый IP, чего в общм случае должно быть достаточно, особенно учитывая ограниченные вычислительные мощности сервера.

В данном контексте особого внимания заслуживает фреймворк Socket.IO. Данный фреймворк является компоексом высокоуровневых оберток над библиотекой WebSocket, что позволяет не реализовывать существенный объем требуемой логики вручную, а полагаться на программные решения, предварительно созданные разработчиками. В частности, в отличии от вебсокетов, сокет.ио поддерживает восстановление соединения после кратковременного разрыва, позволяет использовать вебсокеты посредсвом альтернативных каналов связи, поддерживают проксирования, что может оказаться кртичиески важным при необходимости работы в нескольких сетях, а также поддерживают изящную деградацию, что в свою очередь обеспечивает простоту их внедрения даже на ранних стадиях написания продукта. Более детально вопросы проксирования будут рассмотрены ниже, когда речь пойдет о настройке среды и тотальной виртуализации системы, которая требуется для возможности передачи информации не только от терминалов к серверу, но и от сервера к терминанам по его инициативе, в условиях отсутствия возможности использования белого IP адреса для каждого из терминалов, ввиду низкой экономической эффективности данного технического решения.

Ключевой момент. Который стоит рассматривать, говоря о данном микросервисе, это именно обработка логики сокетов. Однако нельзя не упомянуть, что значительную часть объема кода занимает первичное конфигурирование приложения, а также ряд внутренних функций, например – позволяющих прелставлять аргументы строки запроса в виде объекта языка JavaScript. Настройка приложения в общем смысле аналогична настройке приложения для сервиса логгирования, поэтому акценитирования внимания на ней удастся избежать, однако пристальному рассмотрению должен подлежать объект TID\_connections и метод fint\_TID. В объекте TID\_connections расположено множесво объектов с полями, описывающими id соединения, объект соединения и интеднификатор терминала. При первом подключении какого-либо устройства сервер проверчяет, есть ли ииндентификатор терминала хотя бы одном из объектов, хранящихся в коллекции соединений, и игнорирует попытку соединения в случае наличия. При этом в случае отсутствия в данном массие объекта, попадающего под хаданные критерии, данный объект созадется и добавляется.