Вариант по методичке Калабина

# Перечень принятых сокращений

# Введение

# Глава 1. Техническая часть

## Описание предметной области.

## Анализ подходов и методов решения задачи.

## Разработка технического задания

# Глава 2. Специальная часть

## Выбор среды разработки

## Выбор средств проектирования

## Выбор языка программирования

## Выбор библиотек для разработки

# Глава 3. Проектная часть

## Информационное обеспечение

### Анализ и выбор архитектуры приложения

### Анализ и выбор СУБД для проекта

### Проектирования приложения. Компоненты приложения

### Проектирование API структуры приложения и его модулей

## Программное обеспечение

### Решения по обеспечению доступа к базе данных (к данным).

### Описание программных модулей.

### Схема взаимосвязи (взаимодействия) программных модулей.

## Отладка программных модулей.

# Глава 4. Документация

## Описание технических требований к условиям эксплуатации программного продукта (модуля).

## Программа и методика испытаний.

## Руководство программиста и пользователя.

# Заключение

# Используемый источники

Свое творчество

# Перечень принятых сокращений

# Введение

# Анализ предметный области

## Анализ и выбор методологий разработки // какие есть и почему выбрал водопад

## Анализ и выбор СУБД для проекта //Какие субд есть и почему выбрал MySql Анализ и выбор архитектуры приложения //Какие архитектуры есть и почему выбрал микросервисы

## Выбор языка программирования

## Выбор библиотек для разработки // Тут будет про спринг и его компоненты

## Описания архитектуры REST API // Что такое как применяется и.т.д

# Разработка

## Проектирование //Uml – абстрактно

## Структура приложения/проекта // уже более подробно

## Структура БД

### Тут будет подзаголовок для каждого сервиса ( 6 сервисов)

## Описание REST API

### Тут будет подзаголовок для каждого сервиса ( 6 сервисов)

## Программная реализация //Пример работы системы

# Описание результата

# Деплой проекта (docker) // пока не уверен но если воды не налью то добавлю

# Заключение

# Источники

# (Части кода, sql , презентация итд)

# Перечень принятых сокращений

API (Application Programming Interface) – программный интерфейс приложения.

JVM (Java Virtual Machine) - Виртуальная машина Java, которая служит для исполнения Java байт-кода.

OS – (Operation System) – Операционная система

GC – (Garbage Collector) – Сборщик мусора

HTTP (HyperText Transfer Protocol) – протокол передачи гипертекста.

ID (Identifier) – идентификатор; JSON (JavaScript Object Notation) – текстовый формат обмена данными, основанный на JavaScript.

REST (Representational State Transfer) – передача состояния представления.

SQL (Structured Query Language) – язык структурированных запросов.

UML (Unified Modeling Language) – унифицированный язык моделирования.

БД – База данных.

# Введение

# Глава 1. Техническая часть

## Описание предметной области. В ходе выполнения работы рассматривается деятельность магазина электроники. Необходима выделить основные аспекты работы магазина и создать бекэн

## Анализ подходов и методов решения задачи.

CMS vs нормальное приложение ( как варик )

## Разработка технического задания

Требования к системе:

Система должна предоставлять следующие операции:

Для не авторизированных пользователей:

* Регистрации в системе.
* Получение всех продуктов для данной группы.
* Получение конкретного продукта.
* Получение отзывов для продукта.
* Получение рекомендаций к продукту.

Для авторизированных пользователей:

* Все возможности не авторизированных пользователей.
* Добавление отзывов для продукта.
* Редактирование своих отзывов.
* Удаление своих отзывов.
* Получение рекомендация основанных на предпочтениях пользователя.
* Возможность добавлять продукты в корзину.
* Возможность редактировать количество продуктов в корзине.
* Возможность осуществлять заказ продуктов.
* Возможность отменять заказ.
* Возможность следить за статусом заказа.

Для администраторов системы:

* Все возможности не авторизированных пользователей.
* Добавление продуктов в систему.
* Удаление продуктов из системы.
* Редактирование продуктов.
* Удаление отзывов других пользователей.
* Редактирование отзывов других пользователей.
* Возможность отменять заказы других пользователей.

Общие требование для системы

* Система должна быть масштабируемая.
* Все данные должны храниться в Базе данных.
* Система должна быть проста в развертывание на сервере.
* API системы должен быть задокументирован.
* Описание внутренней структуры базы данных.
* Описание процесса установки системы.

Требования к сопровождению системы Этапы внедрения системы

* Установка базы данных на сервере.
* Тестирование работы системы.
* Исправление ошибок.
* Техническая поддержка системы.

# Глава 2. Специальная часть

## Анализ предметной области

При разработке программных продуктов необходимо провести анализ существующих вариантов для решения задачи. В данном разделе рассматриваются общие теоретические вопросы, освещающие технологические аспекты, необходимые для реализации программного продукта корпоративного класса. В частности, производится выбор среды разработки, СУБД, архитектуры приложения, языка программирования и архитектуры API. В каждом подразделе обосновывается выбор того или иного технического решения задачи.

## Выбор среды разработки

В качестве основной платформы для написания дипломной работы была выбрана Java Virtual Machine.

Виртуальная машина Java это абстрактная вычислительная машина. Как и реальная вычислительная машина, она имеет набор инструкций и манипулирует разными участками памяти во время своей работы.

Первая реализации виртуальной машины Java был представлена компанией Sun Microsystems, Inc в 1994 году

Непосредственно виртуальная машина Java «не знает» ничего о языке программирования, на котором написан исходный код приложения, ей лишь известен заданный формат двоичных файлов – файлов, имеющих расширение .class. Эти файлы содержат инструкции виртуальной машины (байткод), таблицы символов и другую вспомогательную информацию. Из соображений безопасности виртуальная машина Java предъявляет строгие синтаксические и структурные требования на код, расположенный в .class файле. Тем не менее, любой язык, функциональность которого может быть выражена средствами корректного .class файла, может быть интерпретирован для виртуальной машины Java. Привлечённые общедоступностью и платформенной независимостью, разработчики компиляторов других языков могут использовать виртуальную машину как удобную платформу для своих реализаций.

JVM имеет две основные функции:

1. Позволяет запускать Java-приложения на любых устройствах или операционных системах (принцип — «Написал один раз, запускай везде»)
2. Управляет и оптимизирует память, используемую приложением

Таким образом JVM – это одна из лучший платформ для разработки приложений из-за совей кроссплатформенности нам не нужно заботиться о том на какой OS запущено наше приложение и об организации работы с памятью GC очистит память за нас, а из того, что JVM – языка независимая платформа, то допускается смешивания языков программирования, например в Java проекте может быть код на Kotlin, Groovy или Scala. Поскольку JVM довольно старая платформа, то под нее написано довольно много библиотек, что является еще одним плюсом в качестве выбора этой платформы.

## Анализ и выбор СУБД

Существуют два вида СУБД: реляционные (SQL) и нереляционные (NoSQL).

Что бы выбрать какой вид базы данных нам использовать нужно знать характерные отличия их друг от друга.

Можно выделить следующий отличительные признаки:

1. Разница в структуре и типе хранимых данных. В реляционных СУБД должна быть однозначно определена структура хранения данных, в то время как в NoSQL БД нет таких строгих ограничений, и они допускают хранение любого типа данных. Для наших микросервисов подходит структура данных с однозначно определенными связями между объектами таблиц.
2. Различия в структуре запросов. В SQL БД данные могут быть получены только посредством языка SQL, так как реляционные СУБД соответствуют стандартам SQL. В нереляционных СУБД нет четкого определения, как должен выглядеть запрос и поэтому для каждой NoSQL БД реализуется свой способ работы с данными. Для нашей системы требуется построение множества запросов с выборками по условию, и использование нереляционной СУБД будет сопряжено с проблемами с созданием запросов, поэтому выбор реляционной СУБД для работы здесь очевиден.
3. Масштабируемость. В большинстве случаев SQL базы данных вертикально масштабируемые, то есть вы можете увеличивать нагрузку на отдельно взятый сервер, наращивая мощность центральных процессоров, объёмы ОЗУ или системы хранения данных. А NoSQL базы данных горизонтально масштабируемы. Хотя NoSQL базы данных лучше масштабируют горизонтально, для нашей систем более важным является строгая структура хранения данных, а при увеличении нагрузки можно прибегнуть кластеризации базы данных.
4. Надежность. SQL базы данных являются более надежными, по сравнению с NoSQL БД. Поэтому реляционные БД более предпочтительны для выбора в качестве решения для микросервиса
5. Поддержка. Реляционные СУБД появились гораздо раньше NoSQL, поэтому существуют уже готовые платные и бесплатные решения и поддержка, и найти ответ на свой вопрос гораздо проще, чем когда речь идет о сложном решении в нереляционной СУБД. И снова выбор в пользу SQL.
6. Хранение и доступ к сложным структурам данных. SQL базы данных выигрывают в производительности по сравнению с нереляционными СУБД, так как изначально предполагалось, что они будут работать именно со сложными структурами данных.

Подводя итоги анализа выбора между реляционных и нереляционных СУБД для нашей системы, можно с уверенностью сказать, что очевидным выборам является реляционная СУБД.

## Выбор реляционной СУБД

Для выбора наиболее подходящей PСУБД необходимо проанализировать имеющиеся на рынке популярные и развивающиеся SQL базы данных. А именно: MS SQL Server, Oracle Database, MySQL, PostgreSQL.

### MS SQL Server

Microsoft SQL Server — система управления реляционными базами данных (РСУБД), разработанная корпорацией Microsoft. Основной используемый язык запросов — Transact-SQL, создан совместно Microsoft и Sybase. Transact-SQL является реализацией стандарта ANSI/ISO по структурированному языку запросов (SQL) с расширениями. Используется для работы с базами данных размером от персональных до крупных баз данных масштаба предприятия; конкурирует с другими СУБД в этом сегменте рынка.

Достоинства РСУБД MS SQL Server:

* простота использования;
* возможность регулировки и отслеживания уровня производительности, чтобы уменьшить загрузку;
* последняя версия продукта работает стабильно;
* возможность интеграции с другими продуктами Microsoft;
* возможность визуализации на мобильных устройствах.

Недостатки РСУБД MS SQL Server:

* очень высокая стоимость продукта для юридических лиц;
* возможны проблемы в работе служб интеграции импорта файлов;
* высокая ресурсоемкость SQL Server (даже при настройке производительности).

При всех упомянутых достоинствах, у данной РСУБД есть очень существенный недостаток, который делает ее недоступной для использования – ее высокая цена для юридических лиц. Бесплатная версия не обладает достаточными возможностями для полной реализации базы данных для микросервиса

### Oracle Database

Oracle Database - это объектно-реляционная система поддерживающая некоторые технологии, реализующие объектно-ориентированный подход, то есть обеспечивающих управление создания и использования баз данных.

Достоинства ОРСУБД Oracle Database:

* надежность;
* современный функционал и новые разработки, доступные в последней версии;

Недостатки ОРСУБД Oracle Database:

* высокая стоимость продукта для юридических лиц;
* для системы необходимы аппаратные ресурсы, обладающие достаточной мощностью и производительностью;

К сожалению, высокая стоимость не позволяет использовать данную ОРСУБД, бесплатная версия позволяет развертывать небольшие по объему базы данных

### PostgreSQL

PostgreSQL – свободная объектно-реляционная система управления базами данных с открытым исходным кодом, разработчиком которой является сообщество PostgreSQL

Достоинства PostgreSQL:

* бесплатная и для личного и для коммерческого использования;
* хорошо масштабируется;
* возможность обрабатывать терабайты данных;
* поддержка формата JSON;
* существует множество готовых, встроенных функций;
* хорошая документация;
* большое сообщество с поддержкой.

Недостатки PostgreSQL:

* неэффективная репликация данных;
* трудности с обновлением;
* Производительность: В простых операциях чтения PostgreSQL может уступать своим соперникам.

Поскольку нам важна скорость операции чтения, например получение продуктов или отзывов, то PostgreSQL проигрывает по этому параметру.

### MySQL

MySQL – свободная реляционная система управления базами данных. С 2010 года права на торговую марку получила компания Oracle. MySQL является бесплатным программным обеспечением, регулярно выходят новые версии, дополняющие функционал. Существуют и платные версии для коммерческих организаций. В бесплатной версии сделан упор на надежность и скорость работы, но не на полноту функционала. MySQL является довольно гибкой СУБД за счёт возможности использовать множество типов таблиц.

Достоинства MySQL:

* бесплатная для личного использования;
* имеет хорошую документацию;
* безопасность;
* возможность использования множества функций в бесплатной версии;
* возможность взаимодействия с другими базами данных, например DB2 и Oracle.
* Реплицируемость: Базу данных MySQL можно распределять между несколькими узлами, таким образом уменьшая нагрузку и улучшая масштабируемость и доступность приложения;
* Шардинг: В то время как шардинг невозможен на большинстве SQL баз данных, MySQL является исключением.

Недостатки MySQL:

* проблемы с надежностью;
* платная для коммерческого использования
* некоторые ограничения функционала;

Поскольку MySQL весьма гибкий и простой в обращении инструмент, который к тому же обладает высокой степенью безопасности, а также поддерживает репликацию и шардинг было принято решение использовать именно эту СУБД.

## Выбор архитектуры приложения

Существует два основных типа архитектуры для проектирования веб-приложения: монолитная и микросервисная. Чтобы понять какую архитектуру выбрать нужно проанализировать особенности каждой архитектуры.

### Монолитная архитектура приложения

Разработка нормального приложения начинается с модульной многоуровневой или шестиугольной архитектуры. Эта архитектура состоит из следующих типов слоев:

1. **Уровень представления.** Это уровень графического интерфейса пользователя, который обрабатывает запросы протокола передачи гипертекста (HTTP) с использованием HTML или XML / JSON.
2. **Уровень** бизнес-логики: бизнес-логика приложения присутствует на этом уровне.
3. **Уровень доступа** к базе данных: все обращения к базе данных, включая SQL и NoSQL приложений, происходят на этом уровне.
4. **Уровень интеграции приложений**: все интеграции программного обеспечения с другими системами происходят на этом уровне.

Несмотря на то, что монолитная архитектура имеет логическую многоуровневую архитектуру, конечные приложения будут упакованы в один монолит и затем развернуты таким образом. Монолитным приложениям не хватает надлежащей модульности, и она имеет только одну кодовую базу. На рисунке 1 показано графическое представление монолитной архитектуры.

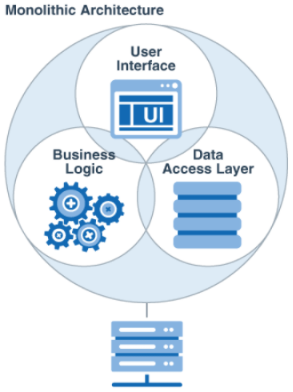


Рисунок 1 — графическое представление монолитной архитектуры

Преимущества монолитной архитектуры:

* сравнительно простая реализация, развертывание и управление;
* проще поддерживать согласованность кода и обрабатывать ошибки;
* легко добавить типичный функционал к компонентам;
* достаточно высокая производительность для малых и средних приложений.

Но при кажущихся преимуществах монолитной архитектуре присущи достаточно серьезные недостатки:

* возрастающая сложность и запутанность приложения;
* с увеличением приложения все сложнее поддерживать изолированность сервисов, чтобы масштабировать каждый из них независимо или осуществлять поддержку кода;
* внесение изменений сопряжено с проблемами и занимает много времени;
* сбой одного сервиса приводит к сбою всего приложения.

Последний недостаток, при котором от сбоя в одном сервисе начинает сбоить всё приложение, является неприемлемым, если рассматривать его в контексте работы интернет-магазина.

### Микросервисная архитектура приложения

## Выбор средств проектирования

## Выбор языка программирования

## Выбор библиотек для разработки

# Глава 3. Проектная часть

## Информационное обеспечение

### Анализ и выбор архитектуры приложения

### Анализ и выбор СУБД для проекта

### Проектирования приложения. Компоненты приложения

### Проектирование API структуры приложения и его модулей

## Программное обеспечение

### Решения по обеспечению доступа к базе данных (к данным).

### Описание программных модулей.

### Схема взаимосвязи (взаимодействия) программных модулей.

## Отладка программных модулей.

# Глава 4. Документация

## Описание технических требований к условиям эксплуатации программного продукта (модуля).

## Программа и методика испытаний.

## Руководство программиста и пользователя.

# Заключение

# Используемый источники

<https://en.wikipedia.org/wiki/Java_virtual_machine>

<http://javacogito.net/index.php/Спецификация_виртуальной_машины_Java#.D0.9D.D0.B5.D0.BC.D0.BD.D0.BE.D0.B3.D0.BE_.D0.B8.D1.81.D1.82.D0.BE.D1.80.D0.B8.D0.B8>

<https://ru.wikipedia.org/wiki/Microsoft_SQL_Server>

<https://tproger.ru/translations/sqlite-mysql-postgresql-comparison/>