Общество с ограниченной ответственностью «ФИНЧ»

ОГРН 1147746053872 ИНН 7701384862

107014, город Москва, Попов проезд, дом 1, корпус 1, подвал 0 помещение XV

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

«Система управления контентом «FINCH CMS»

Документация, содержащая описание функциональных характеристик программного обеспечения

на 10 листах

2022 г.

1 ОБЩЕЕ ОПИСАНИЕ ПО

Программное обеспечение «Система управления контентом «FINCH CMS» (далее ПО, FINCH CMS) представляет собой набор технологий для создания и управления контентом, который позволяет построить back-end без привлечения разработчиков и затрат на дополнительное программирование. Основное конкурентное преимущество и цель развития - создание удобной среды управления контентом, которую сможет использовать неподготовленный пользователь без специальных технических знаний. Таким образом, FINCH CMS дает возможность любому пользователю управлять контентом максимально просто и оперативно. А также снижает порог входа и стоимость разработки для интернет-проектов.

FINCH CMS состоит из компонентов, которые позволяют решать следующие задачи:

A) Панель управления через веб-интерфейс дает возможность создавать модели данных, организовывать связи между ними и заполнять контентом. Возможность создания/управления пользователями в системе и настройки доступов.

B) Back-end автоматически генерирует типизированное GraphQL API, что позволяет быстро начать разработку.

C) Модуль для интеграции с разными типами БД в различных комбинациях.

2 ЭЛЕМЕНТЫ И ПРИНЦИПЫ ВЗАИМОДЕЙСТВИЯ ПО

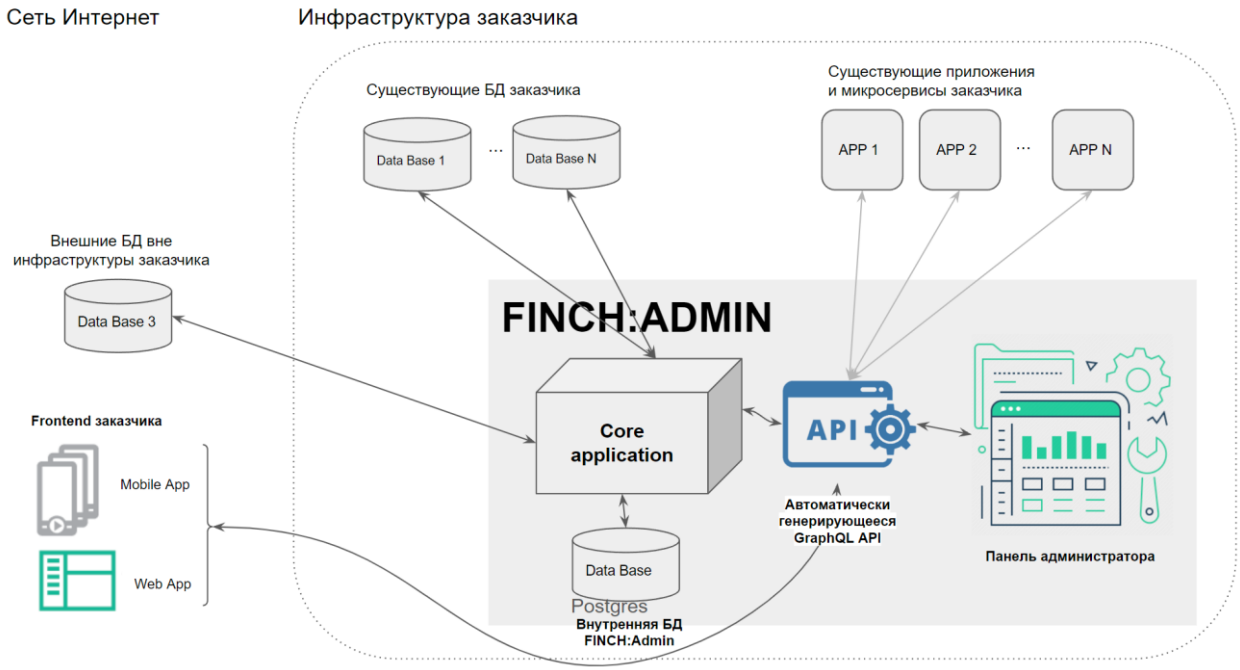
Finch CMS базируется на следующих элементах и их принципах взаимодействия:

- Finch CMS
- Базовые сценарии бизнес-логики
- Сервисы
- Внешние компоненты

Все настройки данных, связь между ними, наполнение контента, создание пользователей и управление ролями производятся в единой среде. Для этого в CMS Finch используется унифицированный набор объектов и их типов как базовых компонентов, с помощью которых можно построить сущности любой сложности.

Пользователь через панель управления может создать модель сущности с требуемыми параметрами, которая будет сохранена в БД. Далее для неё будут сгенерированы CRUD (create, read, update, delete) операции и информация передана в GraphQL API, которое потом будет использоваться для работы с фронтами.

Общая схема



2.1. Бизнес-логика

Представляет собой начальный набор правил, которые определяют возможности пользователя по созданию и управлению контентом на основание доступных ему прав.

Администратор может создавать пользователей и назначать им различные возможности по взаимодействию с данными:

1. Создание
2. Чтение
3. Обновление
4. Удаление

Возможно создание ролей:

1. С независимым доступом на основные операции (часть пользователей

могут только читать, часть обновлять, часть создавать).

2. С ограничением доступа к различным моделям данных (каждая часть пользователей может работать только с необходимыми им данными).

3. Возможно создание групп пользователей, с автоматическим присвоением ролей при попадании пользователя в группу.

Набор правил для взаимодействия и возможные роли можно расширять в зависимости от требований проектов.

Finch CMS

Основная библиотека проекта, содержит в себе набор компонентов, которые предоставляют базовые возможности для создания и управления данными. Использует:

1. finch-json: компонент для сериализации, десериализации комплексных объектов и их списков.
2. google-api-client: компонент для создания докер образа
3. spring-security-core: базовый компонент, отвечающий за авторизацию и регистрацию, с разными типами способов логина
4. spring-boot-starter-web: компонент позволяющий реализовывать эндпоинты для рест-взаимодействия с сервисом
5. swagger-models: автоматическая генерация документации к методам проекта. доступным снаружи, и возможностью выполнять тестовые запросы
6. spring-boot-starter-data-jpa: подключение к sql базам данных. создание таблиц в них, и конфигурация моделей.
7. graphql-java: компонент для использования и конфигурации graphql эндпоинта
8. aws-java-sdk-s3: пример использования облачного хранения данных в базе амазона
9. ktor: компонент для асинхронного взаимодействия с эндпоинтами сервиса
10. junit: тестирование базового функционала cms
11. hibernate-core: компонент сериализации и десериализации данных из реляционных таблиц

12. spring-restdocs-mockmvc: компонент генерации тестовых данных, без использования реальных значений из бд

13. jsoup: компонент парсинга содержимого html для последующего извлечения данных из него, а так же шаблонизации данных в html

14. postgresq: компонент для корректного взаимодействия с postgre sql

15. spring-boot-starter: компонент для автоматического конфигурирования большинства сервисов и библиотек для web'a

2.2. Сервисы

Набор сервисов, которые предоставляют системе возможности для взаимодействия с пользователями, хранением и обработкой данных.

User service

Сервис предоставляет возможности по авторизации пользователей, базово авторизация работает через пару логин и пароль. Помимо этого сервис может работать со следующими видами авторизации:

1. Авторизация через смс по номеру телефона
2. Авторизация через социальные сети
3. LDAP - Lightweight Directory Access Protocol
4. OAuth - Open Authorization

Storage

Данный сервис отвечает за хранение данных. Позволяет работать с независимыми хранилищами и разными типами БД:

1. Postgresql
2. Mysql
3. MongoDB

Хранение данных может быть сконфигурировано под любые задачи: могут быть использованы различные типы баз данных, одна или несколько разных, организовано резервирование.

Пользователь в панели управления (без участия программистов) может создавать различные модели данных и сразу же задавать место их хранения и устанавливать связи между ними.

Data management

Сервис отвечает за сериализацию и десериализацию данных.

Plugins

Представляет собой набор плагинов, которые расширяют основные возможности по взаимодействию с данными или другими системами.

1. Возможности по отслеживанию изменений другим сущностями.
2. Возможность трансформации моделей в другие форматы.
3. Возможность обогащения моделей новыми данными.
4. Возможность создание сводных таблиц/графиков.
5. Возможность работы с медиахостингами.
6. Возможности мониторинга использования ресурсов и состояния системы.
7. Возможности создания различных триггеров с настройкой дальнейших действий.

2.3. Внешние компоненты

Внешние компоненты получают доступ к данным Finch CMS через эндпоинты и могут показывать контент или создавать его. Это могут быть мобильные приложения, веб-сайты.

Панель управления

Панель управления представляет собой веб-интерфейс, который предоставляется сразу в системе - не нужно запускать и настраивать отдельный фронт. Предоставляет пользователям следующие возможности:

1. Авторизация
2. Управление пользователями и их возможностями
3. Создание моделей данных и настройка доступа к ним
4. Показ добавленного контента и управление им
5. Настройка навигации панели управления

6. Фильтрация и сортировка данных

3 ТЕХНОЛОГИИ

3.1. Требования к оборудованию, программному обеспечению пользователя

Для развертывания системы требуется аппаратный комплекс на базе процессоров x86-64, где должна быть установлена любая ОС с поддержкой докер образов (linux debian/ubuntu, centos).

Для использования веб панели управления можно использовать любое устройство с поддержкой веб браузеров и JS, основные браузеры:

1. Firefox версия 98+
2. Opera версия 80+
3. Chrom версия 95+
4. Safari версия 14+
5. Microsoft edge версия 98+

3.2. Архитектура системы

Построена на следующих технологиях:

1. **Infrastructure platform:**
2. 1. Docker/docker-compose: система оркестрация контейнеров.
3. 2. Registry: Система хранения образов контейнеров.
4. 3. Nginx-ingress: балансировка входящих, а также межсервисных запросов.

Backend:

1. PostgreSQL: Реляционная база данных для хранения структурной информации.
2. Java, Kotlin: Основной язык разработки
3. Maven: Менеджер пакетов

Frontend:

1. Nodejs, VUE: Основной язык разработки ObjectStorage:

2. Mino S3: Объектное хранилище файлов

3.3. Описание расположения файлов ПО в файловой системе

Ресурсы проекта хранятся внутри контейнера со следующей структурой:

bash-4.4# ls classes/ libs/ resources/

classes/:

META-INF db finch

libs/:

FastInfoset-1.2.15.jar jul-to-slf4j-1.7.32.jar

HikariCP-4.0.3.jar kotlin-reflect-1.6.0.jar

annotations-13.0.jar kotlin-stdlib-1.6.10.jar

antlr-2.7.7.jar kotlin-stdlib-common-1.5.30.jar

antlr4-runtime-4.9.2.jar kotlin-stdlib-jdk7-1.6.10.jar

asm-7.3.1.jar kotlin-stdlib-jdk8-1.6.10.jar

asm-analysis-7.3.1.jar kotlinx-coroutines-core-jvm-1.5.1-native-mt.jar

asm-commons-7.3.1.jar ktor-client-cio-jvm-1.6.4.jar

asm-tree-7.3.1.jar ktor-client-core-jvm-1.6.4.jar

asm-util-7.3.1.jar ktor-client-jackson-1.6.4.jar

aspectjweaver-1.9.7.jar ktor-client-json-jvm-1.6.4.jar

aws-java-sdk-core-1.12.128.jar ktor-client-logging-jvm-1.6.4.jar

aws-java-sdk-kms-1.12.128.jar ktor-http-cio-jvm-1.6.4.jar

aws-java-sdk-s3-1.12.128.jar ktor-http-jvm-1.6.4.jar

byte-buddy-1.11.20.jar ktor-io-jvm-1.6.4.jar

checker-compat-qual-2.5.5.jar ktor-network-jvm-1.6.4.jar

checker-qual-3.5.0.jar ktor-network-tls-jvm-1.6.4.jar

classmate-1.5.1.jar ktor-utils-jvm-1.6.4.jar

commons-codec-1.11.jar libphonenumber-8.12.42.jar

commons-logging-1.2.jar listenablefuture-9999.0-empty-to-avoid-conflict-with-guava.jar

error_prone_annotations-2.7.1.jar log4j-api-2.14.1.jar

failureaccess-1.0.1.jar log4j-to-slf4j-2.14.1.jar

finch-jacqueline-5.5.4.jar logback-classic-1.2.7.jar

finch-jacqueline-front-2.7.3.jar logback-core-1.2.7.jar

finch-json-1.0.0.jar nashorn-core-15.3.jar

flyway-core-8.4.3.jar nimbus-jose-jwt-9.15.2.jar

google-api-client-1.32.2.jar opencensus-api-0.28.0.jar

google-api-client-jackson2-1.32.2.jar opencensus-contrib-http-util-0.28.0.jar

google-http-client-1.40.1.jar p6spy-3.9.1.jar

google-http-client-apache-v2-1.40.1.jar postgresql-42.3.1.jar

google-http-client-gson-1.40.1.jar reactive-streams-1.0.2.jar

google-oauth-client-1.32.1.jar slf4j-api-1.7.22.jar

graphql-java-17.3.jar snakeyaml-1.29.jar

graphql-java-extended-scalars-17.0.jar spring-aop-5.3.13.jar

grpc-context-1.27.2.jar spring-aspects-5.3.13.jar

gson-2.8.8.jar spring-beans-5.3.13.jar

guava-31.0.1-android.jar spring-boot-2.6.0.jar

hibernate-commons-annotations-5.1.2.Final.jar spring-boot-autoconfigure-2.6.0.jar

hibernate-core-5.6.1.Final.jar spring-boot-starter-2.6.0.jar

hibernate-types-52-2.14.0.jar spring-boot-starter-aop-2.6.0.jar

httpclient-4.5.13.jar spring-boot-starter-data-jpa-2.6.0.jar

httpcore-4.4.14.jar spring-boot-starter-jdbc-2.6.0.jar

ion-java-1.0.2.jar spring-boot-starter-json-2.6.0.jar

istack-commons-runtime-3.0.7.jar spring-boot-starter-logging-2.6.0.jar

j2objc-annotations-1.3.jar spring-boot-starter-security-2.6.0.jar

jackson-annotations-2.13.0.jar spring-boot-starter-tomcat-2.6.0.jar

jackson-core-2.13.0.jar spring-boot-starter-web-2.6.0.jar

jackson-databind-2.13.0.jar spring-context-5.3.13.jar

jackson-dataformat-cbor-2.12.3.jar spring-core-5.3.13.jar

jackson-datatype-jdk8-2.13.0.jar spring-data-commons-2.6.0.jar

jackson-datatype-jsr310-2.13.0.jar spring-data-jpa-2.6.0.jar

jackson-module-jaxb-annotations-2.11.0.jar spring-expression-5.3.13.jar

jackson-module-kotlin-2.13.0.jar spring-jcl-5.3.13.jar

jackson-module-parameter-names-2.13.0.jar spring-jdbc-5.3.13.jar

jakarta.activation-api-1.2.2.jar spring-ldap-core-2.3.4.RELEASE.jar
jakarta.annotation-api-1.3.5.jar spring-orm-5.3.13.jar
jakarta.persistence-api-2.2.3.jar spring-security-config-5.6.0.jar
jakarta.transaction-api-1.3.3.jar spring-security-core-5.6.0.jar
jakarta.xml.bind-api-2.3.3.jar spring-security-crypto-5.6.0.jar
jandex-2.2.3.Final.jar spring-security-web-5.6.0.jar
java-dataloader-3.1.0.jar spring-tx-5.3.13.jar
javax.activation-api-1.2.0.jar spring-web-5.3.13.jar
javax.persistence-api-2.2.jar spring-webmvc-5.3.13.jar
jaxb-api-2.3.1.jar spring-websocket-5.3.13.jar
jaxb-runtime-2.3.1.jar stax-ex-1.8.jar
jboss-logging-3.4.2.Final.jar swagger-annotations-1.6.3.jar
jboss-transaction-api_1.2_spec-1.1.1.Final.jar swagger-models-1.6.3.jar
jcip-annotations-1.0-1.jar tomcat-embed-core-9.0.55.jar
jmespath-java-1.12.128.jar tomcat-embed-el-9.0.55.jar
joda-time-2.8.1.jar tomcat-embed-websocket-
9.0.55.jar
jsoup-1.13.1.jar txw2-2.3.1.jar
jsr305-3.0.2.jar
resources/:
META-INF application-local_dev.yml application.yml db finch