ООО «МИ-МИ»
Правообладатель программного обеспечения «FinMe»

Описание технической архитектуры программного обеспечения «FinMe»

Состав, взаимодействие компонентов, хранение данных, безопасность и инфраструктура

Москва, 2026

1. Общие сведения о системе

Настоящий документ содержит описание технической архитектуры (внутреннего устройства и порядка взаимодействия составных частей) программного обеспечения «FinMe» (далее — Система), правообладателем которого является общество с ограниченной ответственностью «МИ-МИ» (ООО «МИ-МИ»).

Система «FinMe» представляет собой универсальную систему управления партнерскими сетями по модели оплаты за целевое действие (CPA — Cost Per Action, то есть модель расчетов, при которой вознаграждение начисляется за фактически совершенное пользователем полезное действие, например регистрацию или оформление заявки). Система позволяет управлять участниками партнерской сети, учитывать переходы пользователей, регистрировать факты совершения целевых действий и формировать отчеты.

По типу реализации Система является клиент-серверным веб-приложением (программой, состоящей из двух взаимодействующих частей — клиентской, работающей в браузере пользователя, и серверной, работающей на удаленном сервере). Модель предоставления доступа — SaaS (Software as a Service, программное обеспечение как услуга: доступ к Системе предоставляется через веб-браузер, установка какого-либо программного обеспечения на устройство пользователя не требуется).

Ниже перечислены основные технологии, применяемые в Системе.

Серверная часть реализована на языке программирования Java версии 21 с применением платформы (набора готовых программных средств для построения приложений) Spring Boot версии 3.3.4 и подсистемы защиты Spring Security.

Клиентская часть реализована на языках JavaScript и TypeScript с применением каркаса построения пользовательского интерфейса Vue.js версии 3, инструмента сборки Vite (программа, собирающая исходный код в готовый к работе набор файлов), библиотеки готовых визуальных компонентов Ant Design Vue и набора стилевого оформления Tailwind CSS. В качестве веб-сервера (программы, отдающей файлы клиентской части браузеру) применяется nginx.

Характеристика	Значение
Наименование продукта	«FinMe»
Правообладатель	ООО «МИ-МИ»
Назначение	Управление партнерскими сетями по модели оплаты за целевое действие (CPA)
Тип программного обеспечения	Клиент-серверное веб-приложение
Модель предоставления доступа	SaaS (программное обеспечение как услуга, доступ через браузер)
Язык серверной части	Java 21
Платформа серверной части	Spring Boot 3.3.4, Spring Security
Языки клиентской части	JavaScript, TypeScript
Каркас клиентской части	Vue.js 3 (сборка Vite, компоненты Ant Design Vue, стили Tailwind CSS)
Веб-сервер клиентской части	nginx

2. Общая схема архитектуры

Архитектура Системы построена по принципу разделения на самостоятельные компоненты, каждый из которых решает свою задачу и обменивается данными с другими компонентами по строго определенным правилам. Такое разделение повышает надежность (сбой одного компонента не приводит к остановке всей Системы), упрощает сопровождение и позволяет масштабировать нагрузку.

Поток данных при работе пользователя выглядит следующим образом. Пользователь открывает веб-интерфейс в браузере. Браузер загружает клиентскую часть с веб-сервера nginx. Клиентская часть обращается к серверной части по программному интерфейсу REST (согласованному способу обмена данными между программами, при котором данные передаются по протоколу HTTP в текстовом формате JSON). Серверная часть обрабатывает запрос, при необходимости обращается к хранилищам данных и возвращает результат клиентской части, которая отображает его пользователю.

Поток данных при обращении к хранилищам организован по назначению данных. Оперативные данные (справочники, настройки, учетные записи, текущие операции) серверная часть записывает в базу данных PostgreSQL версии 15. Аналитические данные, предназначенные для построения отчетов по большим объемам событий, записываются в специализированное хранилище ClickHouse. Загружаемые пользователями файлы сохраняются в объектном хранилище S3 (хранилище данных в виде отдельных объектов-файлов, доступное по сети).

Поток данных при внешних интеграциях возникает, когда серверная часть обращается к внешним службам: объектному хранилищу файлов, внешней службе проверки данных, мессенджеру для уведомлений об ошибках и службе мониторинга. Все обращения к внешним службам не являются обязательными для работы ядра Системы и управляются флагами настройки.

Ниже приведена сводная таблица составных частей Системы с указанием применяемых технологий, назначения и взаимодействующих компонентов.

Компонент	Технология	Назначение	Взаимодействует с
Клиентская часть (веб-интерфейс)	Vue.js 3, TypeScript, Vite, Ant Design Vue, Tailwind CSS	Отображение данных и прием команд пользователя в браузере	Веб-сервер nginx, серверная часть (через REST)
Веб-сервер клиентской части	nginx	Отдача файлов клиентской части браузеру пользователя	Браузер пользователя, клиентская часть
Серверная часть (программный интерфейс)	Java 21, Spring Boot 3.3.4, Spring Security	Обработка запросов, бизнес-логика, управление доступом	Клиентская часть, PostgreSQL, ClickHouse, S3, внешние службы
База данных оперативных данных	PostgreSQL 15	Хранение справочников, настроек, учетных записей, текущих операций	Серверная часть
Аналитическое хранилище	ClickHouse	Хранение событий и построение отчетов по большим объемам данных	Серверная часть
Объектное хранилище файлов	S3 (Yandex Cloud)	Хранение загружаемых пользователями файлов	Серверная часть

Кэш справочников	Ehcache	Ускорение доступа к часто используемым справочным данным	Серверная часть

3. Описание компонентов системы

3.1. Клиентская часть

Клиентская часть представляет собой одностраничное веб-приложение, исполняемое в браузере пользователя. Она построена на каркасе Vue.js версии 3 с применением языков JavaScript и TypeScript. Сборка исходного кода в готовый набор файлов выполняется инструментом Vite. Визуальные элементы интерфейса (таблицы, формы, кнопки, окна) построены на библиотеке готовых компонентов Ant Design Vue, стилевое оформление задается набором Tailwind CSS.

Клиентская часть не хранит данные постоянно и не содержит бизнес-логики учета: она отвечает исключительно за отображение данных, полученных от серверной части, и за передачу команд пользователя. Все содержательные операции выполняются на серверной части. Готовые файлы клиентской части отдаются браузеру веб-сервером nginx.

3.2. Серверная часть и ее доменные модули

Серверная часть реализована на языке Java версии 21 с применением платформы Spring Boot версии 3.3.4. Серверная часть содержит основную бизнес-логику Системы и организована по доменным модулям (обособленным группам программного кода, отвечающим за определенную предметную область). Внедрение зависимостей между компонентами выполняется через конструкторы, кэширование справочных данных обеспечивается средством Ehcache.

Перечень доменных модулей серверной части приведен в таблице ниже.

Модуль	**Назначение**
cpa	Управление партнерской сетью: партнеры, предложения, организации-заказчики, переходы пользователей, обратные вызовы о совершении целевых действий, отчеты
showcase	Конструктор посадочных страниц (витрин) — страниц, на которые направляется пользователь
smartlink	Интеллектуальная маршрутизация переходов (автоматический выбор, куда направить пользователя)
user / security	Управление пользователями, правами доступа и проверкой подлинности
alerts	Оповещения и уведомления
monitoring	Периодический сбор показателей работы Системы
config	Общие настройки: кэш, планировщик задач, хранилище, безопасность
common	Общие служебные средства, используемые несколькими модулями

Между модулями сохраняется единый порядок построения кода: обработчик запроса — объект передачи данных — служба — хранилище данных — сущность. Такой единый порядок упрощает сопровождение и развитие Системы.

3.3. Программный интерфейс REST

Взаимодействие клиентской и серверной частей осуществляется через программный интерфейс REST — согласованный способ обмена данными, при котором запросы и ответы передаются по протоколу HTTP в текстовом формате JSON. Программный интерфейс сгруппирован по назначению обрабатываемых данных.

Всего серверная часть предоставляет **27 программных контроллеров** (точек доступа REST — обособленных групп операций программного интерфейса, каждая из которых отвечает за свой набор действий над определенным видом данных). Обращения к программному интерфейсу проходят проверку подлинности и разграничение доступа средствами подсистемы Spring Security.

4. База данных и хранение данных

В Системе применяется принцип разделения хранилищ данных по назначению. Каждый вид данных хранится в том хранилище, которое наилучшим образом подходит для соответствующего сценария использования. Это повышает быстродействие и надежность Системы.

4.1. PostgreSQL — оперативные данные

Основным хранилищем оперативных данных является реляционная база данных PostgreSQL версии 15. В ней хранятся справочники, настройки, учетные записи пользователей, права доступа и текущие операции. Взаимодействие серверной части с базой данных построено на объектно-реляционном отображении (автоматическом сопоставлении программных объектов и записей в таблицах базы данных).

4.2. ClickHouse — аналитика и отчеты

Для хранения большого объема событий и построения отчетов применяется специализированное аналитическое хранилище ClickHouse. Оно предназначено для быстрой обработки запросов, охватывающих значительные объемы данных — например, при подсчете количества переходов и целевых действий за период. Обращение серверной части к ClickHouse выполняется напрямую по программному интерфейсу доступа к данным, отдельно от основной базы данных.

4.3. S3 — файлы

Загружаемые пользователями файлы хранятся в объектном хранилище S3 — хранилище данных в виде отдельных объектов-файлов, доступном по сети. Хранение файлов вне базы данных разгружает базу данных и упрощает управление файлами.

4.4. Принцип разделения хранилищ

Сводное распределение данных по хранилищам приведено в таблице ниже.

Хранилище	Вид данных	Назначение
PostgreSQL 15	Оперативные данные	Справочники, настройки, учетные записи, текущие операции
ClickHouse	Аналитические данные	События и построение отчетов по большим объемам данных
Объектное хранилище S3	Файлы	Хранение загружаемых пользователями файлов

5. Интеграции с внешними системами

Система взаимодействует с рядом внешних систем и служб. Все внешние интеграции управляются флагами настройки (отдельными параметрами включения и выключения) и не являются обязательными для работы ядра Системы: при отключении или недоступности любой из внешних служб основная работа Системы не нарушается.

- **Объектное хранилище S3 (Yandex Cloud)** — используется для хранения загружаемых пользователями файлов.
- **Внешняя служба проверки данных** — вызывается по программному интерфейсу для проверки данных. При недоступности служба возвращает пустой результат, не нарушая работу Системы.
- **Мессенджер Яндекса** — применяется для автоматической отправки уведомлений об ошибках работы Системы обслуживающему персоналу.
- **Служба мониторинга Yandex Cloud** — используется для периодического сбора показателей работы Системы.

Такой подход обеспечивает устойчивость Системы: временная недоступность внешних служб не приводит к остановке обработки основных операций, а отключение необязательных интеграций возможно без изменения программного кода.

6. Безопасность и разграничение доступа

В Системе реализован комплекс мер по защите данных и разграничению доступа пользователей. Ниже приведены основные механизмы.

6.1. Проверка подлинности

Проверка подлинности пользователей (подтверждение того, что пользователь является тем, за кого себя выдает) выполняется по технологии JWT (JSON Web Token — подписанный электронный маркер доступа, подтверждающий права пользователя). Маркер подписывается по алгоритму HS256 (HMAC-SHA-256 — способ формирования электронной подписи на основе секретного ключа). Срок действия маркера составляет 24 часа. При каждом обращении к серверной части маркер передается в заголовке запроса Authorization.

6.2. Хранение паролей

Пароли пользователей не хранятся в открытом виде. Для каждого пароля вычисляется и сохраняется необратимый хеш (результат преобразования пароля, из которого нельзя восстановить исходный пароль) по алгоритму BCrypt с добавлением случайной соли (дополнительного случайного набора символов, делающего одинаковые пароли разных пользователей различными в хранилище). Это исключает раскрытие паролей даже при получении злоумышленником доступа к содержимому базы данных.

6.3. Ролевая модель доступа

Разграничение доступа построено по ролевой модели: каждому пользователю назначается роль, определяющая набор доступных ему действий. Перечень ролей приведен в таблице ниже.

Роль	Права доступа
SUPER_ADMIN	Полный доступ ко всем функциям Системы
CPA_MANAGER	Управление справочниками и отчетами
CPA_VIEWER	Только просмотр данных без внесения изменений
MFO_MANAGER	Зарезервированная роль (предусмотрена для будущего использования)

6.4. Управление сессиями и транспортное шифрование

Управление сессиями организовано без сохранения состояния (stateless — сервер не хранит сведений о сеансе пользователя между запросами, а определяет права по предъявленному при каждом запросе маркеру доступа). Такой подход упрощает масштабирование и повышает надежность.

Транспортное шифрование HTTPS/TLS (защищенная передача данных по сети с шифрованием) обеспечивается внешним контуром — веб-сервером и сервисом-балансировщиком нагрузки. Внутри защищенного контура серверная часть работает по протоколу HTTP на порту 8080; обращения к внешнему контуру пользователи выполняют только по защищенному соединению.

7. Инфраструктура и развертывание

Система размещается в облачной инфраструктуре Yandex Cloud, регион ru-central1 (расположенный на территории Российской Федерации). Применяются управляемые сервисы облачной платформы, обслуживание которых (обновление, резервное копирование, обеспечение доступности) выполняется силами платформы.

Перечень применяемых управляемых сервисов облачной платформы:

- Управляемый PostgreSQL — кластер (группа взаимосвязанных узлов) из двух узлов с защищенным соединением;
- Управляемый ClickHouse — аналитическое хранилище;
- Объектное хранилище — для хранения файлов;
- Managed Kubernetes — управляемая система оркестрации контейнеров (автоматического запуска, размещения и перезапуска программ, упакованных в контейнеры);

- Yandex Container Registry — реестр контейнеров (хранилище готовых образов программ для развертывания).

Развертывание Системы выполняется на основе контейнеризации Docker (упаковки программы вместе со всем необходимым окружением в самодостаточный образ-контейнер). Серверный образ построен на основе среды выполнения Java 21, клиентский образ — на основе веб-сервера nginx. Образы разворачиваются в управляемом кластере Kubernetes по манифесту (файлу описания состава развертывания), включающему объекты Service (описание способа доступа к приложению) и Deployment (описание запускаемого приложения и числа его копий). Серверная часть слушает порт 8080, для виртуальной машины Java (JVM) выделено от 5 до 6 гигабайт оперативной памяти.

Порты и протоколы, используемые Системой, приведены в таблице ниже.

Порт	Протокол	Назначение
443	HTTPS	Доступ пользователей к веб-интерфейсу (через балансировщик нагрузки)
80	HTTP	Веб-сервер клиентской части (nginx) внутри защищенного контура
8080	HTTP	Серверная часть (программный интерфейс) внутри защищенного контура
6432 / 5432	PostgreSQL	База данных оперативных данных
8443	HTTPS	Аналитическое хранилище ClickHouse (защищенное соединение)
443	HTTPS	Объектное хранилище и внешние службы

8. Обеспечение доступности и резервирование

Для обеспечения непрерывности работы и сохранности данных в Системе предусмотрен ряд мер резервирования и распределения нагрузки.

База данных PostgreSQL развернута как управляемый кластер высокой доступности из двух узлов — ведущего (основного, обрабатывающего запросы) и резервного (принимающего работу при отказе ведущего). Резервное копирование данных выполняется автоматически силами облачной платформы, что позволяет восстановить данные на выбранный момент времени.

Аналитическое и объектное хранилища предоставляются как управляемые отказоустойчивые сервисы облачной платформы: их доступность и сохранность данных обеспечиваются средствами платформы.

Распределение нагрузки между копиями серверной части выполняется через сервис-балансировщик кластера Kubernetes (компонент, равномерно направляющий поступающие запросы на доступные копии приложения). Это позволяет обрабатывать

возрастающую нагрузку и обеспечивает продолжение работы при выходе из строя отдельной копии приложения.

Совокупность перечисленных мер обеспечивает высокий уровень доступности Системы, сохранность данных и устойчивость к отказам отдельных компонентов.