ОБЩЕСТВО С ОГРАНИЧЕННОЙ ОТВЕТСТВЕННОСТЬЮ

«Смарт-Ком»

«Система для управления и оптимизации промоакций ProSpace.Promo»

Инструкция по установке

Версия 1.0

2026

Содержание

1. Введение....4
2. Общие сведения....5
2.1. Назначение и область применения....5
2.2. Эксплуатационное назначение Программы....5
2.3. Задачи, решаемые Программой....5
2.4. Состав Программы и архитектурное окружение....6
3. Требования к аппаратным и программным средствам....8
3.1. Аппаратные требования....8
3.2. Программные требования....8
3.3. Сетевые требования....9
4. Подготовка инфраструктурных компонентов....11
4.1. PostgreSQL 15....11
4.2. MongoDB 4.2....12
4.3. Redis 7.0....13
4.4. RabbitMQ 4.0.8....13
4.5. HashiCorp Vault (опционально)....14
5. Управление сертификатами....16
5.1. HTTPS-сертификат Kestrel....16
5.2. Сертификат подписи (signing)....16
5.3. Сертификат шифрования (encryption)....16
5.4. TLS-сертификат клиентской части....16
6. Конфигурирование Программы....18
6.1. Переменные окружения....18
6.2. Строки подключения....18
6.3. Конфигурация сервисов....19
6.4. Конфигурация журналирования (Serilog)....20
7. Миграции базы данных....21
Вариант А. Через инструменты миграции (рекомендуется)....21
Вариант Б. В составе развёртывания Kubernetes....21
8. Развёртывание серверной части в Kubernetes....22
8.1. Назначение и предварительные требования....22
8.2. Сборка образов контейнеров....22
8.3. Структура пространств имён....23
8.4. Шаг 1. Инициализация конфигурации (helm-init)....23

8.5. Шаг 2. Развёртывание сервисов (helm) 23
8.6. Хранилище и ингресс 24
9. Развёртывание серверной части на выделенном сервере 25
9.1. Подготовка операционной системы 25
9.2. Настройка сетевой безопасности 26
9.3. Размещение файлов сервисов и переменные окружения 26
9.4. Настройка служб systemd 27
9.5. Файловое хранилище 28
9.6. Настройка обратного прокси Nginx 28
10. Развёртывание клиентской части 30
10.1. Сборка 30
10.2. Переменные окружения 30
10.3. Конфигурация времени выполнения 30
10.4. Раздача клиентской части 31
11. Порядок запуска сервисов 32
12. Проверка работоспособности 33
12.1. Конечные точки контроля состояния 33
12.2. Метрики и интерфейс описания API 33
12.3. Контрольная проверка (smoke-тест) 33
13. Откат версии 35
Сценарий Kubernetes 35
Сценарий выделенного сервера 35
Приложение А. Перечень компонентов серверной части 36
Приложение Б. Справочник по окружениям 37
Лист регистрации изменений 37

1. ВВЕДЕНИЕ

Настоящий документ содержит порядок установки программного обеспечения «Система для управления и оптимизации промоакций ProSpace.Promo» (далее — «Программа»).

В разделе «Общие сведения» приведены сведения о назначении Программы и информация, достаточная для понимания её функций, состава и условий эксплуатации.

В разделе «Требования к аппаратным и программным средствам» указаны минимальные требования к вычислительным ресурсам, системному программному обеспечению и сетевой связности, необходимые для развёртывания Программы.

В разделах «Подготовка инфраструктурных компонентов», «Управление сертификатами», «Конфигурирование Программы» и «Миграции базы данных» описана подготовка общего инфраструктурного слоя, который требуется для обоих поддерживаемых сценариев развёртывания серверной части.

Программа поддерживает два сценария развёртывания серверной части:

— развёртывание в кластере Kubernetes средствами пакетного менеджера Helm — раздел «Развёртывание серверной части в Kubernetes»;

— развёртывание на выделенном сервере (bare-metal / виртуальной машине) под управлением системного менеджера служб systemd — раздел «Развёртывание серверной части на выделенном сервере».

В разделе «Развёртывание клиентской части» описаны сборка и публикация пользовательского веб-интерфейса. В разделах «Порядок запуска сервисов» и «Проверка работоспособности» приведены последовательность ввода сервисов в работу и контрольные проверки.

Раздел «Откат версии» содержит порядок возврата серверной части к предыдущей рабочей сборке. Приложения содержат справочные перечни компонентов и окружений.

Эксплуатация Программы после установки осуществляется в соответствии с эксплуатационной документацией на неё.

ПРИМЕЧАНИЕ. Конкретные значения доменных имён, адресов узлов, паролей, токенов и идентификаторов в настоящем документе заменены обобщёнными обозначениями в угловых скобках (например, <ХОСТ>, <ПАРОЛЬ>, <URL_ПРИЛОЖЕНИЯ>, <ОКРУЖЕНИЕ>). Фактические значения задаются при развёртывании в соответствии с параметрами конкретного контура.

2. ОБЩИЕ СВЕДЕНИЯ

2.1. Назначение и область применения

Программа используется для автоматизации управления торговыми промоакциями на всех этапах их жизненного цикла: от создания и планирования до проведения и пост-акционной аналитики.

Программа позволяет организациям оптимизировать стратегии продвижения продукции, повышать прозрачность маркетинговых бюджетов и обеспечивать кросс-функциональное взаимодействие подразделений продаж, маркетинга и финансов.

2.2. Эксплуатационное назначение Программы

Область применения Программы — производители товаров массового спроса, предприятия розничной торговли, компании электронной торговли, производственные компании, оптовики и дистрибьюторы.

2.3. Задачи, решаемые Программой

Программа решает следующие задачи:

— Планирование промо-календаря: создание акций вручную или по шаблонам, массовый ввод, гибкая настройка атрибутов (механика, размер скидки, даты проведения и отгрузки).

— Настройка и управление цепочками согласования промоакций на основе гибкой модели правил.

— Контроль ключевых показателей эффективности: автоматический расчёт объёма продаж, инкрементального объёма, прибыльности, возврата инвестиций (ROI) и других метрик.

— Управление маркетинговыми бюджетами: планирование, согласование, контроль расходования бюджета на любом уровне иерархии.

— Формирование отчёта о прибылях и убытках по акции с учётом всех затрат как для производителя, так и для розничной сети.

— Прогнозная аналитика с применением машинного обучения: прогноз объёма продаж на промо, выделение инкрементальных объёмов.

— Анализ план/факт: автоматическое сопоставление плановых и фактических показателей акции.

— Визуализация данных: преднастроенные и пользовательские дашборды и отчёты с помощью встроенного построителя.

— Управление пользователями и разграничение прав доступа на основе ролевой модели.

— Интеграция с внешними корпоративными системами для обмена данными.

2.4. Состав Программы и архитектурное окружение

Серверная часть Программы состоит из набора взаимосвязанных микросервисов. Перечень микросервисов и назначаемых им сетевых портов приведён в таблице.

Сервис	Порт (разработка)	Порт (Kubernetes)	Назначение
Gateway (YARP)	7032	443 / 80	Обратный прокси, трансформация авторизации, маршрутизация запросов
Promo API	7053	443 / 80	Основная бизнес-логика управления промоакциями
Security API	7010	5002 / 5003	Аутентификация, авторизация, управление пользователями
TaskManager API	7018	443 / 80	Оркестрация фоновых задач (Hangfire)
FileStorage API	7176	5014 / 5015	Загрузка и выдача файлов, раздача статики
Import API	7017	443 / 80	Импорт данных (CSV, Excel)
ETL Service	7129	443 / 80	Обработка файлового обмена и интеграционных потоков
Mars.Promo	7277	443 / 80	Интеграционный сервис обмена данными

Для работы серверной части требуется набор внешних инфраструктурных компонентов, перечисленных в таблице.

Компонент	Версия	Порт	Назначение
PostgreSQL	15	5432	Основная реляционная база данных (4 базы данных)

Компонент	Версия	Порт	Назначение
MongoDB	4.2	27017	Хранилище унаследованных данных (legacy)
Redis	7.0	6379	Кеш, публикация/подписка, распределённые блокировки
RabbitMQ	4.0.8	5672 / 15672	Асинхронный обмен сообщениями
HashiCorp Vault	опционально	8200	Централизованное управление секретами

Логическая структура взаимодействия компонентов приведена ниже. Клиентский веб-интерфейс обращается к шлюзу Gateway, который маршрутизирует запросы к прикладным сервисам; сервисы используют общие инфраструктурные компоненты для хранения данных, кеширования и обмена сообщениями.

```
Браузер / SPA
    |
    v
Gateway (YARP)  --->  Security API  --->  PostgreSQL / Redis
    |
    +--->  Promo API       --->  PostgreSQL / MongoDB / Redis / RabbitMQ
    +--->  TaskManager API --->  PostgreSQL / RabbitMQ
    +--->  FileStorage API --->  PostgreSQL / MongoDB / Redis
    +--->  Import API      --->  Promo API
    +--->  ETL Service     --->  PostgreSQL / Redis / RabbitMQ
    +--->  Mars.Promo      --->  PostgreSQL
```

Клиентская часть представлена одностраничным веб-приложением (SPA) на платформе Vue 3, которое раздаётся веб-сервером Nginx. Информационный обмен осуществляется по протоколам HTTP(S)/REST + JSON и WebSocket; интеграция с внешними системами — средствами пакетной и потоковой обработки (ETL), а также REST API по запросу.

3. ТРЕБОВАНИЯ К АППАРАТНЫМ И ПРОГРАММНЫМ СРЕДСТВАМ

3.1. Аппаратные требования

Минимальные требования к вычислительным ресурсам серверной части (на основе ограничений ресурсов для развёртывания в Kubernetes) приведены в таблице.

Сервис	CPU, запрос	CPU, лимит	ОЗУ, запрос	ОЗУ, лимит
Promo API	500m	1500m	3,5 Gi	5 Gi
Gateway	250m	500m	250 Mi	500 Mi
Security API	250m	500m	250 Mi	500 Mi
TaskManager API	250m	500m	500 Mi	800 Mi
FileStorage API	250m	500m	250 Mi	500 Mi
Import API	250m	250m	130 Mi	250 Mi
ETL Service	250m	500m	250 Mi	500 Mi
Mars.Promo	250m	500m	250 Mi	500 Mi
Итого (сервисы приложения)	2,25 CPU	4,75 CPU	5,38 Gi	8,55 Gi

Дополнительно для инфраструктурных компонентов: PostgreSQL — не менее 2 ядер CPU и 4 Gi ОЗУ; Redis — 0,5 ядра и 512 Mi; RabbitMQ — 0,5 ядра и 512 Mi; MongoDB — 1 ядро и 1 Gi.

ПРИМЕЧАНИЕ. Рекомендуемая минимальная конфигурация для промышленной эксплуатации: 8 ядер CPU, 16 Gi ОЗУ, 100 Gi дискового пространства на твердотельном накопителе.

3.2. Программные требования

Перечень системного и инструментального программного обеспечения приведён в таблице. Набор требуемых инструментов зависит от выбранного сценария развёртывания.

Программное обеспечение	Версия	Назначение
.NET Runtime / ASP.NET Core Runtime	8.0+	Среда выполнения серверной части
.NET SDK	8.0+	Сборка серверной части
Node.js	22+	Сборка клиентской части
pnpm	9.12.3+	Пакетный менеджер клиентской части
Docker	20.10+	Контейнеризация и запуск инфраструктуры
Kubernetes	1.25+	Оркестрация (сценарий Kubernetes)
Helm	3.0+	Управление пакетами Kubernetes
dotnet-ef	8.0.11	Выполнение миграций базы данных
Nginx	1.25+	Раздача клиентской части (промышленная эксплуатация)

3.3. Сетевые требования

Входящие порты, открываемые клиентам, приведены в таблице.

Порт	Протокол	Назначение
443	HTTPS	Клиентская часть (Nginx) и Gateway
80	HTTP	Перенаправление на HTTPS

Внутренние порты, используемые при взаимодействии сервисов и инфраструктурных компонентов, приведены в таблице.

Порт	Протокол	Компонент
5432	TCP	PostgreSQL
27017	TCP	MongoDB
6379	TCP	Redis
5672 / 15672	TCP	RabbitMQ (AMQP / веб-интерфейс управления)
8200	TCP	HashiCorp Vault
7010–7277	HTTPS	Микросервисы серверной части (режим разработки)

Требования к службе доменных имён (DNS): запись типа A для домена приложения (например, promo.example.com); при развёртывании в Kubernetes — внутреннее разрешение имён сервисов вида *.prospace-promo-app.svc.

Требования к защите канала (TLS): действующий TLS-сертификат для домена клиентской части; самоподписанные или внутренние сертификаты удостоверяющего центра для межсервисного взаимодействия.

4. ПОДГОТОВКА ИНФРАСТРУКТУРНЫХ КОМПОНЕНТОВ

Раздел описывает подготовку внешних инфраструктурных компонентов, общих для обоих сценариев развёртывания серверной части. Компоненты могут быть развёрнуты в контейнерах (Docker), как управляемые сервисы платформы или как системные службы. Ниже приведены примеры настройки через Docker и форматы строк подключения.

ВНИМАНИЕ. В примерах конфигурации значения паролей и токенов заменены обозначением <НАДЁЖНЫЙ_ПАРОЛЬ>. Запрещается хранить значения секретов в открытом виде в файлах конфигурации, передаваемых по незащищённым каналам. Передача учётных данных, файлов сертификатов и маркеров доступа допускается только через защищённые каналы и хранилища.

4.1. PostgreSQL 15

Для работы Программы требуются четыре базы данных PostgreSQL, перечисленные в таблице.

База данных	Используется сервисами	Ключ строки подключения
Promo	Promo API, FileStorage API, Security API	DbContext-postgres
TaskManager	TaskManager API	TaskManager-postgres
Integration	ETL Service	Integration-postgres
Security	Security API	DbContext-postgres (Security)

Пример настройки через Docker:

```
# docker-compose.yml
prospace-promo-postgres:
  image: postgres:15-alpine
  ports:
    - "5432:5432"
  environment:
    POSTGRES_USER: postgres
    POSTGRES_PASSWORD: <НАДЁЖНЫЙ_ПАРОЛЬ>
```

```
  POSTGRES_DB: Promo
 volumes:
  - postgres-data:/var/lib/postgresql/data
  - ./init-scripts/postgres:/docker-entrypoint-initdb.d:ro
 shm_size: '1g'
```

Скрипт инициализации создаёт остальные базы данных (init-scripts/postgres/01-init-databases.sql):

```
CREATE DATABASE "TaskManager";
CREATE DATABASE "Integration";
CREATE DATABASE "Security";

GRANT ALL PRIVILEGES ON DATABASE "Promo"       TO postgres;
GRANT ALL PRIVILEGES ON DATABASE "TaskManager" TO postgres;
GRANT ALL PRIVILEGES ON DATABASE "Integration" TO postgres;
GRANT ALL PRIVILEGES ON DATABASE "Security"    TO postgres;
```

Формат строки подключения:

```
Host=<ХОСТ>;Port=5432;Database=<ИМЯ_БД>;Username=<ПОЛЬЗОВАТЕЛЬ>;
Password=<ПАРОЛЬ>;Include Error Detail=true;MaxPoolSize=300;Timeout=30
```

Рекомендации для промышленной эксплуатации: использовать пул соединений; включить архивацию журнала упреждающей записи (WAL) для резервного копирования; установить параметр shared_buffers равным 25 % от объёма ОЗУ; задать shm_size не менее 1 Gi.

4.2. MongoDB 4.2

ВНИМАНИЕ. MongoDB используется для хранения унаследованных данных (legacy). Компонент сохраняется для совместимости и не применяется для новой функциональности.

Пример настройки через Docker:

```
prospace-promo-mongo:
 image: mongo:4.2
 ports:
  - "27017:27017"
```

```
  environment:
    MONGO_INITDB_DATABASE: prospacepromo
    MONGO_INITDB_ROOT_USERNAME: admin
    MONGO_INITDB_ROOT_PASSWORD: <НАДЁЖНЫЙ_ПАРОЛЬ>
  volumes:
    - mongo-data:/data/db
```

Формат строки подключения:

```
mongodb://<ПОЛЬЗОВАТЕЛЬ>:<ПАРОЛЬ>@<ХОСТ>:27017/<ИМЯ_БД>?connectTimeoutMS=60000&ssl=false
```

Компонент используется сервисами Promo API и FileStorage API.

4.3. Redis 7.0

Пример настройки через Docker:

```
prospace-promo-redis:
  image: redis:7.0.0-alpine
  ports:
    - "6379:6379"
  command: redis-server --requirepass <НАДЁЖНЫЙ_ПАРОЛЬ> --appendonly yes
  volumes:
    - redis-data:/data
```

Формат строки подключения:

```
<ХОСТ>:6379,defaultDatabase=0,abortConnect=false,allowAdmin=true,
syncTimeout=10000,password=<ПАРОЛЬ>
```

Компонент используется сервисами Promo API, Security API, FileStorage API, Gateway и ETL Service для кеширования, публикации/подписки межсервисных событий и распределённых блокировок.

4.4. RabbitMQ 4.0.8

Пример настройки через Docker:

```
prospace-promo-rabbitmq:
  image: rabbitmq:4.0.8-management
```

```
ports:
  - "5672:5672"    # AMQP
  - "15672:15672"  # веб-интерфейс управления
environment:
  RABBITMQ_DEFAULT_USER: <ПОЛЬЗОВАТЕЛЬ>
  RABBITMQ_DEFAULT_PASS: <НАДЁЖНЫЙ_ПАРОЛЬ>
volumes:
  - rabbitmq-data:/var/lib/rabbitmq
```

Параметры подключения задаются в конфигурации сервисов в секции RabbitMQOptions (хост, порт, пользователь, пароль). Компонент используется сервисами Promo API, TaskManager API и ETL Service. При развёртывании в Kubernetes RabbitMQ запускается как отдельный под в пространстве имён приложения, данные размещаются в постоянном томе.

4.5. HashiCorp Vault (опционально)

Хранилище секретов Vault применяется для централизованного управления секретами в окружениях, отличных от среды разработки. При установке переменной окружения IGNORE_VAULT=true сервисы используют локальные файлы appsettings.<ОКРУЖЕНИЕ>.json, и Vault не требуется.

Метод аутентификации сервисов в Vault — AppRole. Параметры подключения задаются переменными окружения, приведёнными в таблице.

Переменная окружения	**Описание**
VAULT_ADDR	URL-адрес сервера Vault
VAULT_MOUNT_POINT	Точка монтирования хранилища KV v2
VAULT_ROLE_ID	Идентификатор роли AppRole
VAULT_SECRET_ID	Секрет роли AppRole
IGNORE_VAULT	Значение true отключает Vault и включает использование локальных файлов конфигурации

Секреты в Vault организованы по структуре: <MOUNT_POINT>/Common/AppSettings — общая конфигурация (строки подключения, параметры RabbitMQ, межсервисный токен); <MOUNT_POINT>/Backend/<ИМЯ_СЕРВИСА> — переопределения для конкретного

сервиса. При запуске каждый сервис получает адрес Vault, точку монтирования и реквизиты роли, после чего считывает значения секретов напрямую из соответствующих путей.

5. УПРАВЛЕНИЕ СЕРТИФИКАТАМИ

Для работы Программы используются четыре вида сертификатов. Ниже приведены их назначение и команды генерации самоподписанных сертификатов для проверочного развёртывания. Для промышленной эксплуатации применяются сертификаты, выданные доверенным удостоверяющим центром.

5.1. HTTPS-сертификат Kestrel

Используется всеми сервисами .NET для защищённых конечных точек HTTPS. Путь к файлу сертификата задаётся переменной окружения:

```
ASPNETCORE_Kestrel__Certificates__Default__Path=/etc/ssl/certs/prospace-promo-app.pfx
```

Генерация самоподписанного сертификата:

```
openssl req -x509 -newkey rsa:4096 -keyout key.pem -out cert.pem -days 365 -nodes \
  -subj "/CN=prospace-promo-app"
openssl pkcs12 -export -out prospace-promo-app.pfx -inkey key.pem -in cert.pem \
  -password pass:
```

5.2. Сертификат подписи (signing)

Используется для подписи журнала аудита и проверки целостности данных. Генерация:

```
openssl req -x509 -newkey rsa:2048 -keyout signing-key.pem -out signing-cert.pem \
  -days 3650 -nodes -subj "/CN=ProSpaceDevCertSigning"
openssl pkcs12 -export -out signing.pfx -inkey signing-key.pem -in signing-cert.pem
```

5.3. Сертификат шифрования (encryption)

Используется для шифрования данных (Data Protection API). Генерация:

```
openssl req -x509 -newkey rsa:2048 -keyout enc-key.pem -out enc-cert.pem \
  -days 3650 -nodes -subj "/CN=ProSpaceDevCertEnc"
openssl pkcs12 -export -out enc.pfx -inkey enc-key.pem -in enc-cert.pem
```

5.4. TLS-сертификат клиентской части

Используется веб-сервером Nginx для защищённого соединения с клиентами. Генерация самоподписанного сертификата:

```
openssl req -x509 -newkey rsa:4096 -keyout prospace.tech.key -out prospace.tech.crt \
  -days 365 -nodes -subj "/CN=*.prospace.tech"
```

6. КОНФИГУРИРОВАНИЕ ПРОГРАММЫ

Конфигурация сервисов задаётся переменными окружения и файлами appsettings.<ОКРУЖЕНИЕ>.json, либо значениями из Vault. Ниже приведены основные параметры конфигурации.

6.1. Переменные окружения

Переменная	Обязательная	По умолчанию	Описание
ASPNETCORE_ENVIRONMENT	Да	Development	Имя окружения; определяет загружаемый файл appsettings
IGNORE_VAULT	Нет	false	Значение true отключает Vault
VAULT_ADDR / VAULT_MOUNT_POINT / VAULT_ROLE_ID / VAULT_SECRET_ID	При Vault	—	Параметры подключения к Vault
ADMIN_LOGIN	Да	—	Логин начального администратора
ADMIN_PASS	Да	—	Пароль начального администратора
ASPNETCORE_URLS	Только K8s	—	Адреса прослушивания Kestrel

6.2. Строки подключения

Ключ	Используется сервисами	База / компонент
DbContext-postgres	Promo, Security, FileStorage	PostgreSQL, БД Promo
TaskManager-postgres	TaskManager	PostgreSQL, БД TaskManager

Ключ	Используется сервисами	База / компонент
Integration-postgres	ETL Service	PostgreSQL, БД Integration
Hangfire	TaskManager	PostgreSQL, БД TaskManager
Redis	Promo, Security, Gateway, FileStorage, ETL	Redis
MongoDbUrl	Promo, FileStorage	MongoDB

6.3. Конфигурация сервисов

Общие параметры для всех сервисов задаются в файле appsettings:

```
{
 "ServiceSecretOptions": { "Token": "<ОБЩИЙ_МЕЖСЕРВИСНЫЙ_ТОКЕН>" },
 "Origins": ["https://<URL_ФРОНТЕНДА>"],
 "AllowedHosts": "*"
}
```

ВНИМАНИЕ. Значение межсервисного токена (ServiceSecretOptions.Token) должно быть одинаковым для всех сервисов — оно используется для аутентификации межсервисных вызовов через заголовок X-Service-Token.

Дополнительные параметры Promo API включают имя базы MongoDB, адреса сервисов TaskManager и FileStorage, параметры RabbitMQ, системного пользователя и провайдеров аутентификации. Дополнительные параметры Security API включают параметры почтового сервера (EmailSettings), адреса проверки и сброса, адрес клиентской части. Дополнительные параметры TaskManager API включают перечень конечных точек выполнения задач (ServiceEndpointsOptions).

Шлюз Gateway содержит назначения кластеров обратного прокси YARP — адреса прикладных сервисов, на которые маршрутизируются запросы:

```
{
 "ReverseProxy": {
  "Clusters": {
   "promoCluster":    { "Destinations": { "s": { "Address": "https://<ХОСТ_PROMO>/" } } },
```

```
    "securityCluster": { "Destinations": { "s": { "Address": "https://<XOCT_SECURITY>/" } }
},
    "taskManagerCluster": { "Destinations": { "s": { "Address":
"https://<XOCT_TASKMANAGER>/" } } }
   }
  }
}
```

6.4. Конфигурация журналирования (Serilog)

В среде разработки журналирование выполняется только в консоль. В промышленной эксплуатации к консоли добавляется приёмник Grafana Loki с метками app (имя сервиса) и env (окружение):

```
{
 "Serilog": {
  "Using": ["Serilog.Sinks.Console", "Serilog.Sinks.Grafana.Loki"],
  "WriteTo": [
    { "Name": "Console" },
    { "Name": "GrafanaLoki", "Args": {
       "uri": "http://<XOCT_LOKI>:3100",
       "labels": [ { "key": "app", "value": "<ИМЯ_СЕРВИСА>" },
                { "key": "env", "value": "<ОКРУЖЕНИЕ>" } ] } }
  ]
 }
}
```

7. МИГРАЦИИ БАЗЫ ДАННЫХ

Перед первым запуском сервисов приложения необходимо выполнить миграции схемы базы данных. Миграции выполняются строго в следующем порядке:

1. Promo — основная бизнес-база данных.
2. TaskManager — база данных оркестрации задач.
3. Integration — база данных ETL-интеграции.

База данных Security заполняется автоматически при первом запуске Security API.

Предварительно устанавливается инструмент dotnet-ef:

```
dotnet tool install --global dotnet-ef --version 8.0.11
export PATH="$PATH:$HOME/.dotnet/tools"
```

Вариант А. Через инструменты миграции (рекомендуется)

Для каждой базы данных запускается соответствующий инструмент миграции:

```
cd Tools/ProSpace.Promo.Migrate.Tool        && dotnet run && cd ../..
cd Tools/ProSpace.TaskManager.Migrate.Tool  && dotnet run && cd ../..
cd Tools/ProSpace.Integration.Migrate.Tool  && dotnet run && cd ../..
```

Строка подключения для инструмента миграции задаётся в его файле appsettings.json (ключ DbContext-postgres).

Вариант Б. В составе развёртывания Kubernetes

Инструменты миграции имеют файлы Dockerfile и запускаются как задания (jobs) Kubernetes. В этом случае миграции выполняются после инициализации конфигурации (раздел 8.4, создание пространств имён и секретов) и до запуска прикладных сервисов.

После выполнения миграций каждый аудируемый сервис при запуске автоматически создаёт и синхронизирует триггеры аудита в схеме audit. Дополнительное ручное вмешательство не требуется.

8. РАЗВЁРТЫВАНИЕ СЕРВЕРНОЙ ЧАСТИ В KUBERNETES

8.1. Назначение и предварительные требования

Раздел описывает развёртывание серверной части Программы в кластере Kubernetes средствами пакетного менеджера Helm. Развёртывание выполняется вручную из репозитория исходного кода. На момент запуска развёртывания требуется наличие:

— Подготовленных инфраструктурных компонентов (раздел 4) и сертификатов (раздел 5).

— Действующего кластера Kubernetes версии 1.25 и выше с настроенным файлом подключения kubeconfig.

— Установленных инструментов Helm 3.0+, kubectl, Docker.

— Доступа к реестру контейнеров для публикации и получения образов.

— Выполненной (или включённой в состав развёртывания) миграции базы данных (раздел 7).

8.2. Сборка образов контейнеров

Каждый сервис серверной части имеет файл Dockerfile. Базовые образы: для сборки — mcr.microsoft.com/dotnet/sdk:8.0, для среды выполнения — mcr.microsoft.com/dotnet/aspnet:8.0. Сборка решения и образов выполняется командами:

```
dotnet publish ProSpace.Api.sln -c Release -o ./publish


docker build -t <РЕЕСТР>/promo:<ВЕРСИЯ>       -f
Application/ProSpace.Promo.Api/Dockerfile .
docker build -t <РЕЕСТР>/security:<ВЕРСИЯ>    -f
Application/ProSpace.Security.Api/Dockerfile .
docker build -t <РЕЕСТР>/gateway:<ВЕРСИЯ>     -f
Application/ProSpace.Gateway/Dockerfile .
docker build -t <РЕЕСТР>/taskmanager:<ВЕРСИЯ> -f
Application/ProSpace.TaskManager.Api/Dockerfile .
docker build -t <РЕЕСТР>/filestorage:<ВЕРСИЯ> -f
Application/ProSpace.FileStorage.Api/Dockerfile .
docker build -t <РЕЕСТР>/import:<ВЕРСИЯ>      -f
Application/ProSpace.Import.Api/Dockerfile .
```

Собранные образы публикуются в реестр контейнеров (docker push).

8.3. Структура пространств имён

Пространство имён	Назначение
prospace-promo-web	Gateway, клиентская часть (Nginx), правила ингресса
prospace-promo-app	Микросервисы серверной части, RabbitMQ
prospace-promo-config	Vault (при размещении внутри кластера)

8.4. Шаг 1. Инициализация конфигурации (helm-init)

Инициализирующий чарт helm-init создаёт пространства имён, словари настроек (ConfigMap), секреты, развёртывание RabbitMQ и правила ингресса. Установка выполняется командой:

```
cd Deployment/helm-init


helm install prospace-config . \
  -f values-<ОКРУЖЕНИЕ>.yaml \
  --set appurl=<ВАШ_ДОМЕН> \
  --set ASPNETCORE_ENVIRONMENT=<ИМЯ_ОКРУЖЕНИЯ>
```

В результате создаются:

— Пространства имён prospace-promo-web и prospace-promo-app.

— Словари настроек env-config (имя окружения, конфигурация клиентской части config.js, конфигурация Nginx).

— Секреты: vault-secret (реквизиты Vault), rabbitmq-secret, certs (signing.pfx, enc.pfx, prospace-promo-app.pfx), tls-pfx, admin-credentials (логин и пароль администратора), ingress-tls (TLS-сертификат ингресса).

— Развёртывание RabbitMQ как пода в пространстве имён приложения.

— Правила ингресса: /api и /static — на gateway-service:443, / — на frontend-service:4443.

8.5. Шаг 2. Развёртывание сервисов (helm)

Развёртывание прикладных сервисов выполняется основным чартом helm с файлами значений для выбранного окружения:

```
cd Deployment/helm
```

```
helm install prospace . \
  -f env/<ОКРУЖЕНИЕ>/values-promo.yaml \
  -f env/<ОКРУЖЕНИЕ>/values-gateway.yaml \
  -f env/<ОКРУЖЕНИЕ>/values-securityadmin.yaml \
  -f env/<ОКРУЖЕНИЕ>/values-taskmanager.yaml \
  -f env/<ОКРУЖЕНИЕ>/values-filestorage.yaml \
  -f env/<ОКРУЖЕНИЕ>/values-import.yaml \
  -f env/<ОКРУЖЕНИЕ>/values-mars-promo.yaml \
  -f env/<ОКРУЖЕНИЕ>/values-mars-integration.yaml
```

Пример файла значений сервиса (values-promo.yaml):

```
namespace: prospace-promo-app
port: 80
portssl: 443
image: "<РЕЕСТР>/promo:<ВЕРСИЯ>"
resources:
  requests: { cpu: 500m, memory: 3.5Gi }
  limits:   { cpu: 1500m, memory: 5Gi }
```

Все сервисы монтируют сертификаты signing.pfx и enc.pfx из секрета certs по пути /etc/ssl/certs, а также общее файловое хранилище (постоянный том app-storage-pvc) по пути /mnt/storage. Сервисам задаются дополнительные переменные окружения адресов прослушивания и пути к сертификату Kestrel; сервисы Security API, TaskManager API и Mars получают реквизиты администратора из секрета admin-credentials.

8.6. Хранилище и ингресс

Постоянное файловое хранилище подключается через запрос постоянного тома app-storage-pvc, монтируемый в /mnt/storage. Класс хранилища определяется платформой (например, csi-s3 для облачного хранилища или локальный том для bare-metal-кластера).

Маршрутизация внешнего трафика выполняется контроллером ингресса. Запросы пути /api и /static направляются на gateway-service (порт 443), запросы пути / — на frontend-service (порт 4443). Защита соединения обеспечивается секретом ingress-tls либо средствами cert-manager.

9. РАЗВЁРТЫВАНИЕ СЕРВЕРНОЙ ЧАСТИ НА ВЫДЕЛЕННОМ СЕРВЕРЕ

Раздел описывает развёртывание серверной части на выделенном сервере (физическом или виртуальном) под управлением операционной системы семейства Linux (Ubuntu 22.04/24.04 LTS или совместимый отечественный дистрибутив) с запуском сервисов в виде служб системного менеджера systemd. Команды приведены для дистрибутивов семейства Debian/Ubuntu; для дистрибутивов семейства RHEL пакетный менеджер apt заменяется на dnf, межсетевой экран ufw — на firewalld.

9.1. Подготовка операционной системы

Выполнить обновление системы и установку базовых пакетов:

```
sudo apt update && sudo apt upgrade -y
sudo apt install -y curl wget gnupg2 ca-certificates apt-transport-https \
  unzip jq net-tools fail2ban ufw auditd chrony logrotate nginx
```

Установить среду выполнения .NET 8 и средства сборки клиентской части:

```
wget https://packages.microsoft.com/config/ubuntu/$(lsb_release -rs)/packages-microsoft-prod.deb
sudo dpkg -i packages-microsoft-prod.deb && sudo apt update
sudo apt install -y dotnet-runtime-8.0 aspnetcore-runtime-8.0

curl -fsSL https://deb.nodesource.com/setup_22.x | sudo -E bash -
sudo apt install -y nodejs && sudo npm install -g pnpm@9.12.3
```

Создать системную группу и сервисного пользователя (без интерактивной оболочки) для запуска сервисов по принципу наименьших привилегий:

```
sudo groupadd --system prospace
sudo useradd --system --gid prospace --shell /usr/sbin/nologin \
  --comment "ProSpace .NET services" prospace
```

Создать структуру каталогов и назначить права доступа:

```
sudo mkdir -p /opt/prospace/{promo-api,security-api,gateway,taskmanager-api,\
  filestorage-api,import-api,etl-service,mars-promo}
sudo mkdir -p /etc/prospace/{certs,env}
sudo mkdir -p /mnt/storage/{static,uploads,temp}
sudo mkdir -p /var/log/prospace /run/prospace
```

```
sudo chown -R prospace:prospace /opt/prospace /mnt/storage /var/log/prospace /run/prospace
sudo chown -R root:prospace /etc/prospace
sudo chmod 750 /opt/prospace /mnt/storage /var/log/prospace /run/prospace /etc/prospace
sudo chmod 700 /etc/prospace/certs
sudo chmod 640 /etc/prospace/env/*
```

9.2. Настройка сетевой безопасности

Настроить межсетевой экран: запретить входящие соединения по умолчанию, разрешить только необходимые порты:

```
sudo ufw default deny incoming
sudo ufw default allow outgoing
sudo ufw allow from <IP_АДМИНИСТРАТОРА>/32 to any port 22 proto tcp  # SSH
sudo ufw allow 443/tcp   # HTTPS
sudo ufw allow 80/tcp    # HTTP (перенаправление на HTTPS)
sudo ufw enable
```

Инфраструктурные компоненты (PostgreSQL, Redis, RabbitMQ, MongoDB) при размещении на том же узле должны прослушивать только адрес 127.0.0.1 и не открываться во внешнюю сеть. Рекомендуется дополнительно применить меры усиления защиты согласно практикам CIS Benchmark и OWASP: запрет входа суперпользователя по протоколу SSH и вход только по ключам, защита параметров ядра (sysctl), защита от подбора паролей (fail2ban), аудит системных событий (auditd), синхронизация времени (chrony) и ротация журналов (logrotate).

9.3. Размещение файлов сервисов и переменные окружения

Опубликованные файлы каждого сервиса размещаются в соответствующем подкаталоге /opt/prospace. Параметры запуска и секреты сервиса задаются в файле переменных окружения /etc/prospace/env/<сервис>.env (права доступа 640, владелец root:prospace). Пример файла для Promo API:

```
ASPNETCORE_ENVIRONMENT=Production
ASPNETCORE_URLS=https://127.0.0.1:7053
ASPNETCORE_Kestrel__Certificates__Default__Path=/etc/prospace/certs/prospace-promo-app.pfx
ASPNETCORE_Kestrel__Certificates__Default__Password=<ПАРОЛЬ_СЕРТИФИКАТА>
IGNORE_VAULT=true
ADMIN_LOGIN=superadmin
ADMIN_PASS=<НАДЁЖНЫЙ_ПАРОЛЬ>
```

```
DOTNET_CLI_TELEMETRY_OPTOUT=1
```

Каждый сервис прослушивает собственный порт на локальном адресе (Security API — 7010, Promo API — 7053, TaskManager API — 7018 и так далее) по протоколу HTTPS с сертификатом Kestrel (раздел 5.1). Шлюз Gateway и веб-сервер Nginx обращаются к сервисам по HTTPS, что согласуется с маршрутизацией Gateway (раздел 6.3).

ВНИМАНИЕ. Файл переменных окружения содержит секреты. Он должен иметь права доступа 640 и владельца root:prospace; запрещается размещать его в каталогах, доступных на чтение посторонним пользователям.

9.4. Настройка служб systemd

Для каждого сервиса создаётся юнит systemd с включёнными мерами изоляции (sandboxing). Пример юнита для Promo API (/etc/systemd/system/prospace-promo.service):

```
[Unit]
Description=ProSpace Promo API
After=network.target postgresql.service redis.service
Wants=network-online.target

[Service]
Type=notify
User=prospace
Group=prospace
WorkingDirectory=/opt/prospace/promo-api
ExecStart=/usr/bin/dotnet ProSpace.Promo.Api.dll
EnvironmentFile=/etc/prospace/env/promo.env
Restart=always
RestartSec=10
LimitNOFILE=65535
NoNewPrivileges=yes
ProtectSystem=strict
ReadWritePaths=/mnt/storage /var/log/prospace /run/prospace /tmp
PrivateTmp=yes
ProtectHome=yes

[Install]
```

```
WantedBy=multi-user.target
```

Аналогичные юниты создаются для остальных сервисов: prospace-security, prospace-gateway, prospace-taskmanager, prospace-filestorage, prospace-import, prospace-etl, prospace-mars. Активация и запуск служб:

```
sudo systemctl daemon-reload
sudo systemctl enable --now prospace-security prospace-promo prospace-taskmanager \
  prospace-filestorage prospace-import prospace-etl prospace-mars prospace-gateway
```

9.5. Файловое хранилище

Путь к файловому хранилищу задаётся в конфигурации сервисов параметром StorageUrl (например, /mnt/storage). Каталог должен существовать и быть доступен сервисному пользователю на запись:

```
mkdir -p /mnt/storage/static
chown -R prospace:prospace /mnt/storage
```

9.6. Настройка обратного прокси Nginx

Рекомендуемый вариант — запуск шлюза Gateway как отдельной службы systemd с маршрутизацией всего трафика API через него; Nginx при этом раздаёт клиентскую часть и проксирует запросы на Gateway:

```
server {
    listen 443 ssl;
    server_name <URL_ПРИЛОЖЕНИЯ>;
    ssl_certificate     /etc/ssl/certs/prospace.tech.crt;
    ssl_certificate_key /etc/ssl/certs/prospace.tech.key;
    client_max_body_size 10M;

    location /api/    { proxy_pass https://localhost:7032/api/; }
    location /static/ { proxy_pass https://localhost:7032/static/; }
    location / {
        root /var/www/prospace/dist;
        try_files $uri $uri/ /index.html;
    }
}
```

ПРИМЕЧАНИЕ. Допускается альтернативный вариант с прямым проксированием Nginx на каждый сервис (минуя Gateway), однако при этом не выполняются преобразования авторизации шлюза. Рекомендуется использовать вариант с Gateway.

10. РАЗВЁРТЫВАНИЕ КЛИЕНТСКОЙ ЧАСТИ

Клиентская часть представляет собой одностраничное веб-приложение на платформе Vue 3. Образ или каталог статических файлов клиентской части собирается из исходного кода и раздаётся веб-сервером Nginx (в сценарии Kubernetes — в составе развёртывания, в сценарии выделенного сервера — из каталога публикации).

10.1. Сборка

Сборка выполняется пакетным менеджером pnpm с указанием режима, соответствующего целевому окружению:

```
cd Application/Frontends/ProSpace/ProSpace
pnpm install
pnpm build --mode <ИМЯ_ОКРУЖЕНИЯ>
```

10.2. Переменные окружения

Переменные времени сборки задаются в файлах .env.<окружение>. Основные переменные приведены в таблице.

Переменная	Описание
VITE_API_BASE_URL	URL-адрес шлюза API (например, https://<URL_ПРИЛОЖЕНИЯ>/api/)
VITE_ROOT_BASE_URL	Корневой URL-адрес приложения (необязательная)
VITE_API_REDIRECT_URL	URL-адрес перенаправления при авторизации
VITE_DEBUG	Включение режима отладки (true/false)
VITE_INTERNAL_MODE	Признак внутреннего режима (необязательная)

10.3. Конфигурация времени выполнения

При развёртывании в Kubernetes значения параметров клиентской части могут переопределяться во время выполнения через файл config.js, подставляемый из словаря настроек (ConfigMap). Это позволяет использовать единый образ для нескольких окружений, задавая адреса приложения непосредственно при развёртывании.

10.4. Раздача клиентской части

Собранные статические файлы размещаются в каталоге веб-сервера Nginx. Маршрутизация одностраничного приложения настраивается директивой try_files с перенаправлением неизвестных путей на index.html; запросы пути /api проксируются на шлюз Gateway (см. раздел 9.6).

11. ПОРЯДОК ЗАПУСКА СЕРВИСОВ

Ввод сервисов в работу выполняется в строго определённой последовательности фаз. Переход к следующей фазе допускается только после успешного прохождения контрольных проверок предыдущей.

1. Фаза 1. Инфраструктура. Запуск в порядке: PostgreSQL, Redis, RabbitMQ, MongoDB. Дождаться прохождения проверок работоспособности.

2. Фаза 2. Миграции базы данных. Выполнение миграций в порядке: Promo, TaskManager, Integration (см. раздел 7).

3. Фаза 3. Security API. Запускается первым среди сервисов приложения, так как остальные сервисы зависят от него для аутентификации.

4. Фаза 4. Основные сервисы приложения. Запуск (допускается параллельно): Promo API, FileStorage API, TaskManager API, Import API, ETL Service, Mars.Promo. Import API использует Promo API, поэтому запускается после него.

5. Фаза 5. Gateway. Запускается после готовности сервисов приложения для корректного обнаружения маршрутов.

6. Фаза 6. Клиентская часть. Запуск веб-сервера Nginx с раздачей клиентской части.

12. ПРОВЕРКА РАБОТОСПОСОБНОСТИ

После завершения развёртывания выполняется проверка работоспособности компонентов.

12.1. Конечные точки контроля состояния

Все сервисы предоставляют конечную точку контроля состояния вида /<базовый_путь>/health, возвращающую признаки доступности сервиса и базы данных. Перечень приведён в таблице.

Сервис	Конечная точка	Ожидаемый ответ
Promo API	GET /api/promo/health	{"alive": true, "hasDb": true, "dbAlive": true}
Security API	GET /api/security/health	{"alive": true, "hasDb": true, "dbAlive": true}
TaskManager API	GET /api/tasks/health	{"alive": true, "hasDb": true, "dbAlive": true}
FileStorage API	GET /api/fileStorage/health	{"alive": true, "hasDb": true, "dbAlive": true}
ETL Service	GET /api/integration/health	{"alive": true, "hasDb": true, "dbAlive": true}
Mars.Promo	GET /api/mars/promo/health	{"alive": true, ...}

12.2. Метрики и интерфейс описания API

Каждый сервис предоставляет метрики в формате Prometheus по адресу GET /metrics, а также интерактивный интерфейс описания программного интерфейса (Swagger) по адресу /<базовый_путь>/swagger (например, /api/promo/swagger, /api/security/swagger).

12.3. Контрольная проверка (smoke-тест)

Рекомендуемая последовательность контрольной проверки:

1. Инфраструктура. PostgreSQL — pg_isready; Redis — redis-cli ping (ответ PONG); RabbitMQ — запрос состояния узла; MongoDB — команда ping (ответ ok: 1).

2. Миграции. Наличие записей в таблице "__EFMigrationsHistory" в базах Promo, TaskManager, Integration.

3. Сервисы. Запросы к конечным точкам контроля состояния через Gateway возвращают код 200.

4. Аутентификация. Запрос POST /api/authorization/signin с реквизитами администратора возвращает маркер доступа (JWT).

5. Клиентская часть. По публичному адресу приложения загружается одностраничное приложение и отображается страница входа; вход администратора выполняется успешно.

13. ОТКАТ ВЕРСИИ

Откат к предыдущей рабочей версии выполняется повторным развёртыванием ранее опубликованной сборки. Перед обновлением рекомендуется зафиксировать текущую метку версии (тег образа или релизную метку) для возможной операции отката.

Сценарий Kubernetes

Откат выполняется средствами Helm — возвратом к предыдущей ревизии выпуска либо повторным развёртыванием с указанием предыдущей метки образа:

```
helm history prospace -n prospace-promo-app
helm rollback prospace <НОМЕР_РЕВИЗИИ> -n prospace-promo-app
```

Управляемый кластер Kubernetes выполняет последовательную замену подов. После отката выполняется проверка работоспособности (раздел 12).

Сценарий выделенного сервера

Откат выполняется заменой опубликованных файлов сервиса на файлы предыдущей рабочей сборки и перезапуском соответствующих служб:

```
sudo systemctl stop prospace-promo
# заменить содержимое /opt/prospace/promo-api на предыдущую сборку
sudo systemctl start prospace-promo
```

ВНИМАНИЕ. Откат версии не выполняет обратную миграцию схемы базы данных. При наличии несовместимых изменений схемы откат прикладной части согласуется с порядком отката миграций, определённым эксплуатационной документацией.

ПРИЛОЖЕНИЕ А. ПЕРЕЧЕНЬ КОМПОНЕНТОВ СЕРВЕРНОЙ ЧАСТИ

Компонент	Назначение
ProSpace.Gateway	Шлюз (обратный прокси YARP), маршрутизация и трансформация авторизации
ProSpace.Promo.Api	Основная бизнес-логика управления промоакциями
ProSpace.Security.Api	Аутентификация, авторизация, управление пользователями
ProSpace.TaskManager.Api	Оркестрация фоновых задач
ProSpace.FileStorage.Api	Загрузка и выдача файлов, раздача статики
ProSpace.Import.Api	Импорт данных
ETL Service	Обработка интеграционных потоков и файлового обмена
Mars.Promo	Интеграционный сервис обмена данными
ProSpace.Promo.Migrate.Tool	Инструмент миграции базы данных Promo
ProSpace.TaskManager.Migrate.Tool	Инструмент миграции базы данных TaskManager
ProSpace.Integration.Migrate.Tool	Инструмент миграции базы данных Integration
Frontend (Vue 3)	Клиентская часть — пользовательский веб-интерфейс

ПРИЛОЖЕНИЕ Б. СПРАВОЧНИК ПО ОКРУЖЕНИЯМ

Программа поддерживает несколько окружений эксплуатации, различающихся источником конфигурации и параметрами инфраструктуры. Имя окружения задаётся переменной ASPNETCORE_ENVIRONMENT и определяет загружаемый файл конфигурации. Ключевые различия приведены в таблице.

Параметр	**Среда разработки**	**Промышленная эксплуатация**
Источник конфигурации	appsettings.<ОКРУЖЕНИЕ>.json	Vault (или appsettings при IGNORE_VAULT=true)
Журналирование	Только консоль	Консоль и Grafana Loki
Сертификаты	Самоподписанные	Файлы PFX в секретах
Миграции	Автоматически при запуске	Отдельные задания миграции

ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ

Изм.	Номера листов (страниц)				Всего листов (страниц) в докум.	Номер докум.	Вход. номер сопров од. докум.	Подпись	Дата
	Изме-ненных	Заменен ных	Новых	Аннулиро ванных					