АКЦИОНЕРНОЕ ОБЩЕСТВО «ТРАНССЕТЬ» (АО «ТРАНССЕТЬ»)

СИСТЕМА АВТОМАТИЗАЦИИ ПРОПУСКНОГО РЕЖИМА В ЦЕЛЯХ ОБЕСПЕЧЕНИЯ ТРАНСПОРТНОЙ БЕЗОПАСНОСТИ ТРС.ПРОПУСК

РУКОВОДСТВО АДМИНИСТРАТОРА СИСТЕМЫ

Листов 21

Москва, 2026

АННОТАЦИЯ

Документ представляет собой руководство, предназначенное для администраторов, работающих с комплексным программным обеспечением «Система автоматизации пропускного режима в целях обеспечения транспортной безопасности (ТРС.Пропуск)». Документ предоставляет описание сервисов и основных компонентов. В частности, описываются действия по установке и настройке ТРС.Пропуск, а также лицензированию.

СОДЕРЖАНИЕ

Термины, определения и сокращения 4
1 Общие сведения 7
1.1 Наименование системы 7
1.2 Назначение системы 7
1.3 Функциональные возможности системы 7
2 Инсталляция системы 9
2.1 Программное обеспечение, необходимое для функционирования Системы 9
2.2 Сервисы системы 9
2.3 Первичный запуск системы 11
2.4 Офлайн установка 12
2.5 Защищенный протокол HTTPS 13
2.6 Краткий список команд для управления проектом 13
3 Системные настройки 14
3.1 Основные настройки 14
4 Фоновые процессы 19
5 Модуль лицензирования 20
5.1 Описание модуля 20
5.2 Компиляция приложения генерации лицензии 20
5.3 Генерация лицензии 21
5.4 Компиляция приложения контролирующего лицензии 21
5.5 Шифрование лицензии 21

ТЕРМИНЫ, ОПРЕДЕЛЕНИЯ И СОКРАЩЕНИЯ

Термин	Значение
Application Programming Interface, API	Набор способов и правил, по которым различные программы общаются между собой и обмениваются данными
Docker	Платформа с открытым исходным кодом для автоматизации разработки, доставки и развёртывания приложений, которая позволяет упаковать в контейнер приложение со всем окружением и зависимостями, а затем доставить и запустить его в целевой системе
Docker Swarm	Встроенная система оркестрации контейнеров, предоставляемая Docker, которая позволяет создавать, управлять и масштабировать кластер из Docker-хостов, превращая их в единый логический хост, на котором можно запускать контейнеры
Fastcgi Process Manager, PHP-FPM	Программный пакет, позволяющий выполнить обработку скриптов, написанных на языке PHP
Hyper Text Transfer Protocol Secure, HTTPS	Расширение протокола HTTP для поддержки шифрования в целях повышения безопасности
Hypertext PreProcessor, PHP	Скриптовый язык программирования с открытым исходным кодом
HyperText Transfer Protocol, HTTP	Протокол передачи гипертекста, реализующий набор правил, по которым данные в Интернет передаются между разными источниками, обычно между компьютерами и серверами
JSON Web Token, JWT-токен	Специальный формат токена, который позволяет безопасно передавать данные между клиентом и сервером
Laravel	Бесплатный фреймворк с открытым исходным кодом для веб-разработки на языке программирования PHP
MinIO	Высокопроизводительная объектная система хранения данных с открытым исходным кодом, которая предназначена для облачных и контейнерных приложений

Термин	Значение
Nginx	Программное обеспечение с открытым исходным кодом для создания лёгкого и мощного веб-сервера (также его используют в качестве почтового сервера)
Node.js	Бесплатная среда с открытым исходным кодом, предназначенная для выполнения кода JavaScript на сервере, позволяющая писать серверный код для веб-приложений и динамических веб-страниц, а также для программ командной строки
Object-Relational Mapping, Eloquent ORM	Инструмент, встроенный в Laravel, который позволяет разработчикам работать с базами данных, используя объектно-ориентированный подход
PostgreSQL	Свободно распространяемая объектно-реляционная система управления базами данных с открытым исходным кодом
Python	Высокоуровневый язык программирования общего назначения
Simple Storage Service, S3	Сервис, где хранятся цифровые данные большого объёма, реализующий вариант «плоского» (не иерархического) хранилища, в котором все объекты равнозначны, в котором удобно долго хранить разнородную информацию и быстро получать к ней доступ
TCP/IP	Стек протоколов, которые задают правила передачи данных по сети
Transport Layer Security, TLS/ Secure Sockets Layer, SSL	Криптографические протоколы, обеспечивающие защищённую передачу данных в компьютерной сети
Авторизация	Процедура проверки прав пользователя на доступ к информации. Она носит вторичный характер и происходит после успешной аутентификации пользователя в системе. Таким образом, аутентификация направлена на установление подлинности личности пользователя, а авторизация — на проверку прав пользователя на доступ к информации
Аутентификация	Процедура установления подлинности личности пользователя. В ходе аутентификации пользователь представляет доказательства своей личности, такие как пароль, а система проверяет их

Термин	Значение
База данных, БД	Упорядоченный набор структурированной информации или данных, которые хранятся в электронном виде в компьютерной системе
Браузер	Программное обеспечение, которое позволяет пользователю взаимодействовать с интернетом и служит посредником между пользователем и веб-сайтами, предоставляя доступ к информации, приложениям и медиафайлам через удобный интерфейс
Микросервисная архитектура	Вариант сервис-ориентированной архитектуры программного обеспечения, направленный на взаимодействие небольших, слабо связанных и легко изменяемых модулей — микросервисов
Система	Система автоматизации пропускного режима в целях обеспечения транспортной безопасности (ТРС.Пропуск)
Система управления базами данных, СУБД	Совокупность программных и лингвистических средств общего или специального назначения, обеспечивающих управление созданием и использованием БД
Токен	Идентификатор, который используется для представления доступа к ресурсам или выполнения определённых операций

1 ОБЩИЕ СВЕДЕНИЯ

1.1 Наименование системы

Полное наименование системы: Система автоматизации пропускного режима в целях обеспечения транспортной безопасности.

Сокращённое наименование системы: ТРС.Пропуск

1.2 Назначение системы

Комплексное программное обеспечение «Система автоматизации пропускного режима в целях обеспечения транспортной безопасности (ТРС.Пропуск)» (далее по тексту — Система) предназначено для организации полного цикла работы, связанного с управлением доступом посетителей и транспорта на охраняемые объекты, территории или в помещения.

Основная цель Системы — обеспечение безопасности инфраструктурных объектов с помощью контроля и учёта доступа, а также предотвращение несанкционированного проникновения.

1.3 Функциональные возможности системы

Функциональные возможности Системы:

- Автоматизация рабочего места для создания заявок на выпуск пропусков.
- Автоматизация процесса согласования заявок для выпуска пропусков.
- Автоматизация рабочего места для выпуска, печати и управлением пропусками.
- Автоматизация рабочего места для сотрудников охраны для организации процесса контроля пропусков и фиксирования событий на контрольно-пропускном пункте.
- Автоматизация рабочего места для контроля текущего состояния охраняемого объекта и формирование отчетов по историческим данным событий контрольно-пропускного пункта.

- Аутентификация пользователя для доступа к Системе и предоставление контента аутентифицированному пользователю (авторизация) в соответствии с группой, в которую он включен.
- Обеспечение оперативного информирования пользователей, в том числе посредством электронной почты.
- Предоставление административного интерфейса для управления Системой.
- Обеспечение возможности взаимодействия с другими программами через API, обеспечивая интеграцию с инфраструктурой и возможность обмена данными с другими программами через стандартный интерфейс.

2 ИНСТАЛЛЯЦИЯ СИСТЕМЫ

2.1 Программное обеспечение, необходимое для функционирования Системы

Клиентская часть:

- Операционная система с поддержкой сетевых протоколов TCP/IP.
- Современный веб-браузер одной из следующих версий или выше: Google Chrome (версия 49.0 и выше), Mozilla Firefox (версия 45.0 и выше), Microsoft Edge (версия 12.0 и выше), Opera (версия 36.0 и выше), Safari (версия 10.0 и выше).

  Примечание. Рекомендуется использовать последние стабильные версии браузеров для обеспечения максимальной совместимости, безопасности и производительности. Internet Explorer (версия 11.0) использование не рекомендуется.

Серверная часть:

- Docker версии не ниже 26.0.0.

2.2 Сервисы системы

Система представляет собой приложение, реализованное с использованием принципов микросервисной архитектуры и развернутое в среде Docker Swarm. Компоненты Системы развернуты в Docker-контейнерах и объединены в масштабируемый кластер.

Сервис в микросервисной архитектуре (микросервис) — это небольшой автономный модуль, который отвечает за конкретный элемент логики приложения. Каждый микросервис выполняет одну конкретную функцию и имеет свою базу данных. Разрабатывать, развёртывать и масштабировать такие сервисы можно отдельно друг от друга. Микросервисы взаимодействуют между собой с помощью программного интерфейса приложения (API). Такой набор инструкций и функций в виде интерфейса есть у каждого из них и определяет, какие запросы может получать сервис и как будет на них отвечать.

Сервисы Системы приведены в таблице (Таблица 2.1).

Таблица 2.1 – Сервисы Системы

Сервис	Описание
Redis	Система управления базами данных (далее по тексту — СУБД), которая работает со структурами данных типа «ключ — значение». Это технологический сервис, предназначенный для хранения очереди фоновых заданий, создаваемых ядром Системы для автоматического обновления данных Системы. Реализован с помощью сетевого журналируемого хранилища Redis.
Postgres (17.0)	Объектно-реляционная СУБД, которая обеспечивает надёжность, масштабируемость и широкие возможности для разработки сложных приложений. СУБД реализована по классической клиент-серверной модели, причём и сервер, и клиент могут находиться как на одной машине, так и на разных хостах, взаимодействуя по сети TCP/IP. Также сервер может быть размещен во внешнем дата-центре или облаке, тогда связь обеспечивается по сети интернет.
Core (laravel)	Основной сервис Системы представляет собой веб-приложение, реализованное на PHP фреймворке Laravel, развернутое в среде PHP-FPM.
Core-api (nginx:1.26.2)	Независимая от формата модель объектов документа для представления веб-API. С помощью Core API можно взаимодействовать с API на уровне интерфейса приложения, а не сетевого интерфейса. Это шлюзовой сервис, точка входа сервиса Core для обработки входящих HTTP-запросов. Реализован с помощью приложения Nginx и работает в режиме HTTP-Proxy.
Front (node:22.9.0)	Платформа для обслуживания клиентов, которая помогает пользователям работать эффективнее и предоставлять услуги высочайшего качества. Она объединяет все каналы связи в одном месте, обеспечивая быстрый доступ к информации и совместную работу в режиме реального времени. Веб-сервер, предназначенный для загрузки клиентского веб-приложения в браузер пользователя, реализован на базе среды Node.js. Клиентское веб-приложение реализовано на реактивном фреймворке Vue.js версии 3.
Swagger-ui	Компонент инструмента Swagger, который визуализирует документацию для API. Он преобразует спецификацию OpenAPI в удобный для использования веб-интерфейс. С помощью Swagger UI можно изучать структуру API и полноценно тестировать его, задавая параметры и отправляя запросы в режиме реального времени. Он создаёт интерактивные шаблоны, для работы с которыми не нужно использование кода. Все манипуляции выполняют посредством интерфейса. Это технологический сервис для создания интерактивной документации методов API Системы.
Storage (minio)	Позволяет управлять файлами проекта: загружать, просматривать, скачивать и выполнять другие операции. С помощью контейнеров хранилища можно решать, какие файлы и какого размера допускать в хранилище, шифровать ли их, сканировать антивирусом и так далее. Сервис представляет собой масштабируемое файловое S3-хранилище, которое реализовано с помощью приложения MinIO.

Сервис	Описание
Storagecreatebucket	Служебный сервис для создания контейнеров хранилища (bucket — это сущность для организации хранения в хранилище) для сервиса Storage. S3 — это не иерархическая файловая система, а всего лишь хранилище ключей и значений, хотя ключ часто используется как путь к файлу для организации данных. Чтобы разместить файл в объектном хранилище S3, его необходимо поместить в контейнер хранилища. Использует стандартный образ minio/mc:RELEASE.2024-11-17T19-35-25Z-cpuv1.
Traefik	Шлюзовой сервис реализующий основную точку входа Системы, который предназначен для маршрутизации входящих HTTP-запросов внутри Системы. Это современный обратный прокси-сервер и балансировщик нагрузки, предназначенный для упрощения развёртывания микросервисов и управления ими. Сервис реализован с помощью приложения Traefik.

2.3 Первичный запуск системы

Действия для первичного запуска

1) Установить Docker.
2) Инициализировать инструмент Docker для управления кластерами контейнеров Docker Swarm.
3) Разместить проект в необходимой директории.
4) Создать файл настроек проекта.env и отредактировать параметры. Отдельно нужно обратить внимание на параметры чувствительные к доменному имени по которому предоставляется доступ к Системе. Данные параметры отмечены звездочкой (*) в таблице «Основные настройки» (п. 3.1). Образец настроек файла находится в корне проекта и называется.env.example.
5) Перед началом сборки проекта необходимо обеспечить доступ к репозиториям образов Docker (выход в сеть Интернет). Если нет возможности обеспечить доступ в Интернет необходимо провести процедуру офлайн установки (п. 2.4).
6) Находясь в одной директории с файлом docker-compose-swarm.yml выполнить команду:

```
export $(grep -v '^#' .env | xargs) && docker stack deploy -c docker-compose-swarm.yml <название проекта>
```

7) Начнётся установка всех необходимых сервисов. Первичное наполнение базы запустится автоматически после сборки сервиса core.

2.4 Офлайн установка

Список образов необходимых для работы Системы:

- ais-kpp.core:1.0.1;
- ais-kpp.core-api:1.0.1;
- ais-kpp.front:1.0.1;
- postgres:17.0;
- redis:7.2.7;
- minio/minio:RELEASE.2024-11-07T00-52-20Z-cpuv1;
- minio/mc:RELEASE.2024-11-17T19-35-25Z-cpuv1;
- ais-kpp.api-doc:1.0.1;
- traefik:v2.10.

Действия для офлайн установки:

1) Провести первичный запуск Системы в среде, из которой есть доступ к сети Интернет (п. 2.3).
2) Вывести список всех образов собранных в процессе установки Системы командой:

```
docker service ls
```

3) Выгрузить каждый образ командой вида:

```
docker save -o <имя архива.tar> <имя образа>
```

где <имя образа> указано в колонке IMAGE листинга команды docker service ls.

4) Перенести архивы на сервер на котором будет запускаться Система и загрузить каждый выгруженный образ в локальное хранилище docker-командой:

```
docker load -i <filename>.tar
```

5) Создать каталог проекта и разместить в нём следующие файлы:
   - docker-compose-swarm.yml;
   - docker-compose-swarm.add-ssl-le.yml (для ssl);
   - настроенный файл .env.
6) Находясь в директории проекта выполнить команду:

```
export $(grep -v '^#' .env | xargs) && docker stack deploy -c docker-compose-swarm.yml <название проекта>
```

2.5 Защищенный протокол HTTPS

Для работы по защищенному протоколу HTTPS с применением TLS/SSL настраивается Traefik для работы с сертификатами, согласно официальной документации (https://doc.traefik.io/traefik/https/tls/).

В проекте предусмотрен запуск с использованием SSL (Let's Encrypt). Для этого необходимо выполнить сборку проекта используя команду:

```
export $(grep -v '^#' .env | xargs) && docker stack deploy -c docker-compose-swarm.yml -c docker-compose-swarm.add-ssl-le.yml <название проекта>
```

2.6 Краткий список команд для управления проектом

Краткий список команд для управления проектом приведён ниже.

Остановка работы и удаление контейнеров и связанных сетей, созданных для конкретного стека:

```
docker stack rm <название проекта>
```

Просмотреть список сервисов:

```
docker service ls
```

Остановка конкретного сервиса:

```
docker service rm <service_name>
```

Запустить сервисы (данная команда выполняется из директории, где находится файл docker-compose-swarm.yml):

```
docker stack deploy -c docker-compose-swarm.yml <название проекта>
```

Полный список команд описан в официальной документации (https://docs.docker.com/engine/swarm/?spm=5aebb161.2ef5001f.0.0.2dbec921pS7Jc9).

3 СИСТЕМНЫЕ НАСТРОЙКИ

В данном разделе приведена информация о системных настройках (настройки параметров окружения).

3.1 Основные настройки

Конфигурационный файл (.env), обеспечивающий работы сервисов и параметры файла docker-compose-swarm.yml находится в корне директории проекта. Для того что бы изменения в настройках вступили в силу никаких дополнительных команд не требуется. Описание ключей приведено в таблице (Таблица 3.1).

Таблица 3.1 – Основные настройки

Ключ	Описание
IMAGE_TAG	Версия контейнера сервиса
Настройки базы данных Postgres	
POSTGRES_HOST	Хост базы данных
POSTGRES_INNER_PORT	Внутренний порт базы данных
POSTGRES_OUTER_PORT	Внешний порт базы данных
POSTGRES_USER	Имя root пользователя
POSTGRES_PASSWORD	Пароль root пользователя
Настройки Redis	
REDIS_PASSWORD	Пароль Redis
REDIS_HOST	Адрес хоста Redis
REDIS_INNER_PORT	Внутренний порт Redis
REDIS_OUTER_PORT	Внешний порт Redis
Minio	
MINIO_STORAGE_HOSTNAME	Адрес хоста Minio
MINIO_STORAGE_HOST_PORT	Внутренний порт Minio
MINIO_CONSOLE_OUTER_PORT	Внешний порт Minio
MINIO_USER	Имя пользователя
MINIO_PASSWORD	Пароль
MINIO_BUCKET	Название контейнера хранилища

Ключ	Описание
Common	
COMMON_APP_ENV	Среда выполнения приложения
COMMON_APP_DEBUG	Флаг режима отладки
COMMON_LOCALE	Язык приложения
COMMON_TIMEZONE	Часовой пояс приложения
COMMON_FAKER_LOCALE	Язык и регион для генератора фейковых данных Faker
COMMON_LOCALES	Поддерживаемые языки приложения
COMMON_FORWARDED_HEADERS_TRUSTED_IPS	Диапазон доверительных IP-адресов
Core	
CORE_APP_URL*	Адрес приложения
CORE_APP_NAME	Имя приложения
CORE_HOST	Адрес docker-контейнера
CORE_PORT	Порт
CORE_APP_KEY	Ключ приложения
CORE_DB_CONNECTION	Тип подключенной базы данных
CORE_DB_DATABASE	Имя базы данных
CORE_DB_USERNAME	Имя пользователя базы данных
CORE_DB_PASSWORD	Пароль базы данных
CORE_LOGGING_CHANNELS	Канал логирования
CORE_LOGGING_LEVEL	Уровень логирования
CORE_LOGGING_DOCKER_STDOUT_PROCESS_ID	Это идентификатор процесса (Process ID, PID), в контексте которого выполняется запись логов
CORE_LOGGING_DOCKER_STDOUT_FILE_DESCRIPTOR	Это числовой идентификатор файлового дескриптора, который указывает, в какой поток вывода будут записываться логи: - 1 – стандартный поток вывода (stdout). - 2 – стандартный поток ошибок (stderr).
CORE_LOGGING_DAILY_DAYS	Времени хранения ежедневных файлов журнала
CORE_JWT_SECRET	Строка для проверки подлинности JWT-токена
CORE_JWT_TTL	Время жизни JWT-токена
CORE_JWT_REFRESH_TTL	Время жизни refresh-токена JWT
CORE_PASSWORD_TOKEN_EXPIRE	Время жизни токена для смены пароля
CORE_PASSWORD_RESET_URL	Адрес для сброса пароля пользователя

Ключ	Описание
CORE_MAIL_MAILER	Транспорт для отправки электронной почты
CORE_MAIL_HOST	Имя сервера исходящей почты
CORE_MAIL_PORT	Порт сервера исходящей почты
CORE_MAIL_USERNAME	Имя учетной записи
CORE_MAIL_PASSWORD	Пароль учетной записи
CORE_MAIL_ENCRYPTION	Безопасное подключение: SSL
CORE_MAIL_FROM_ADDRESS	Адрес электронной почты отправителя
CORE_DEFAULT_ADMIN_EMAIL	Адрес электронной почты администратора системы
CORE_S3_AWS_ENDPOINT	Адрес к сервису хранения данных S3
CORE_S3_AWS_URL	Адрес к сервису хранения данных S3
CORE_S3_AWS_DEFAULT_REGION	Регион по умолчанию для S3
CORE_S3_AWS_ACCESS_KEY_ID	Имя пользователя S3
CORE_S3_AWS_SECRET_ACCESS_KEY	Пароль доступа к S3
CORE_S3_AWS_BUCKET	Имя контейнера хранения данных S3
CORE_S3_AWS_USE_PATH_STYLE_ENDPOINT	Параметр, который определяет, стиль URL-адресации для доступа к объектам в S3
CORE_S3_AWS_FILESYSTEM_DISK	Название типа файлового хранилища
CORE_QUEUE_CONNECTION	Брокер сообщений
CORE_REDIS_CLIENT	Параметр, который указывает тип соединения или драйвер, используемый для работы с очередями (queues) в приложении
CORE_CACHE_DRIVER	Параметр, который определяет драйвер, используемый для кэширования данных в приложении
QUEUE_CONNECTION_PASSES	Тип соединения для очереди запуска проверки даты выданных пропусков
QUEUE_QUEUE_PASSES	Название очереди проверки даты выданных пропусков
QUEUE_CONNECTION_NOTIFICATION	Тип соединения для очереди уведомлений
QUEUE_QUEUE_NOTIFICATION	Название очереди уведомлений
CORE_QR_CODE_URL	Шаблон для генерации QR-кода пропуска
CORE_HORIZON_BASIC_AUTH_USERNAME	Имя пользователя для входа в интерфейс приложения для мониторинга и управления очередями задач

Ключ	Описание
CORE_HORIZON_BASIC_AUTH_PASSWORD	Пароль пользователя для входа в интерфейс приложения для мониторинга и управления очередями задач
CORE_HORIZON_EMAIL	Адрес электронной почты для уведомлений из приложения для мониторинга и управления очередями задач
FILE_LICENSE	Путь к файлу лицензии
Core-api	
CORE_API_INNER_PORT	Внутренний порт для соединения с API
CORE_API_OUTER_PORT	Внешний порт для соединения с API
CORE_API_DOMAIN*	Домен API приложения
CORE_API_URL*	Адрес API приложения
Front	
FRONT_APP_DOMAIN*	Домен frontend контейнера
FRONT_APP_URL*	Адрес frontend контейнера
FRONT_APP_OUTER_PORT*	Внешний порт контейнера frontend
FRONT_IP_WHITELIST*	Список разрешенных IP-адресов
FRONT_APP_API_URL*	Адрес frontend контейнера для доступа к API
FRONT_APP_TIME_SETINTERVAL	Времени повторяющегося действия на стороне клиента, в миллисекундах для функции setInterval()
FRONT_APP_TIME_SETTIMEOUT	Задержки в миллисекундах перед выполнением определенного действия на стороне клиента с помощью функции setTimeout()
FRONT_CORE_QR_CODE_URL*	URL указывает на сервис, который обрабатывает сканирование и валидацию QR-кодов
Swagger	
SWAGGER_UI_PORT	Порт сервиса документации API
SWAGGER_EDITOR_PORT	Порт сервиса редактора API
SWAGGER_API_TEMPLATE	Имя основного шаблона
SWAGGER_API_SERVER_URL*	Адрес для маршрутизации запросов к API основного приложения
SWAGGER_UI_DOMAIN*	Адрес сервиса документации API
SWAGGER_UI_IP_WHITELIST*	Список разрешенных IP-адресов
Traefik	
TRAEFIK_DOMAIN*	Домен реверс-прокси (traefik)
TRAEFIK_IP_WHITELIST*	Список разрешенных IP-адресов

Ключ	Описание
TRAEFIK_LOG_LEVEL	Уровень логирования
TRAEFIK_BASICAUTH	Логин и пароль для доступа к traefik

4 ФОНОВЫЕ ПРОЦЕССЫ

После старта Системы и при дальнейшей работе Системы стартуют фоновые процессы (Таблица 4.1). Запуск фоновых процессов осуществляется по расписанию.

Таблица 4.1 – Фоновые процессы

Процесс	Источник	Повторение
Очистка истекших токенов сброса пароля	Система	5 минут
Проверка окончания срока действия пропусков	Система	1 минута
Удаление неподтвержденных шаблонов пропусков	Система	1 час

5 МОДУЛЬ ЛИЦЕНЗИРОВАНИЯ

5.1 Описание модуля

Модуль предназначен для обеспечения контроля за лицензией и генерации ключа лицензии. Реализован на языке Python в виде скомпилированного бинарного файла недоступного для редактирования.

Модуль интегрирован в имидж-файл контейнера core и инициируется при его старте.

Модуль считывает количество пользователей из файла лицензии и сравнивает его с количеством пользователей в базе данных (таблица users). Если превышено количество пользователей, доступных по лицензии или не найден файл лицензии или нет доступа к базе данных, процесс запускает скрипт остановки контейнера (сервиса) core. Перед остановкой контейнера Системой будет отправлено письмо о соответствующей ошибке на адрес электронной почты указанный в файле .env в переменной CORE_HORIZON_EMAIL (п. 3.1). Дополнительно файле лога (/usr/local/src/core/storage/logs/license_error.log) создаётся запись с меткой времени и текстом ошибки.

Для предотвращения случайного превышения количества пользователей в системе предусмотрена валидация при создании пользователя. Проверка лицензионного ключа выполняется при запуске контейнера и затем повторяется каждые 10 (десять) секунд.

5.2 Компиляция приложения генерации лицензии

Для компиляции приложения необходимо перейти в каталог ais_kpp_license/licensing_v1 и выполнить последовательно следующие команды:

```
pyinstaller --onefile --name=<название для файла> <путь к скрипту>/license_generation.py
staticx <путь и имя к файлу сгенерированному первой командой> <путь и имя для нового файла>
```

5.3 Генерация лицензии

Для генерации лицензии необходимо из командной строки запустить приложение, которое было скомпилировано в п. 5.2. В консоли ответить на все вопросы для генерации файла лицензии.

При успешной генерации файл появится в той же директории, где расположено приложение генерации.

Файл необходимо поместить в каталог /core/docker/app/license/ или в тот каталог, который указан в файле .env.

Файл будет добавлен в образ при следующей сборке образа контейнера.

5.4 Компиляция приложения контролирующего лицензии

Для компиляции приложения необходимо перейти в каталог ais_kpp_license/licensing_v1 и выполнить последовательно следующие команды:

```
pyinstaller --onefile --name=<название для файла> <путь к скрипту>/main.py
staticx <путь и имя к файлу сгенерированному первой командой> <путь и имя для нового файла>
```

Готовые файлы для проверки и генерации лицензии расположены в папке dist/staticx.

5.5 Шифрование лицензии

Принцип шифрования лицензии основан на побитовом исключающем «ИЛИ» (XOR). Каждый символ преобразуется в числовое значение с помощью функции ord(), затем к нему применяется операция «ИЛИ» (XOR) с соответствующим символом ключа. После этого результат обратно преобразуется в символ с помощью функции chr() и добавляется в результирующий список. Если длина ключа меньше длины незашифрованного текста, ключ повторяется циклически.