Инструкция по установке и эксплуатации web-приложения "Электронная очередь ТРИЦ. Сириус"

Краткое описание

Система "Электронная очередь ТРИЦ. Сириус" представляет собой современное программное решение для автоматизации управления потоками посетителей в организациях различного профиля. Система обеспечивает упорядоченное обслуживание клиентов, снижает время ожидания и повышает качество сервиса за счет равномерного распределения нагрузки между операторами.

Основные функции системы:

- **Интерфейс терминала самообслуживания** - позволяет посетителям самостоятельно выбрать нужную услугу и получить электронный талон с номером очереди. Поддерживает печать талонов на термопринтере с указанием времени получения и типа услуги
- **Интерфейс оператора** - предоставляет сотрудникам удобные инструменты для управления очередью: вызов следующего клиента, просмотр количества ожидающих, настройка типов обслуживаемых услуг, управление паузами в работе
- **Информационные панели** - отображают актуальную информацию для посетителей: текущие вызываемые номера, указание окон обслуживания

Принцип работы системы

Система "Электронная очередь ТРИЦ. Сириус" построена по принципу распределенной архитектуры и состоит из нескольких взаимодействующих компонентов:

1. **Клиенты получают талоны** через терминалы самообслуживания
2. **Операторы вызывают клиентов** через интерфейс оператора
3. **Посетители видят информацию** на информационных панелях
4. **Система автоматически управляет очередью** и распределяет нагрузку

Основные преимущества

- **Гибкость настройки** - возможность настроить любое количество услуг и операторов под специфику организации. Система адаптируется к различным бизнес-процессам и может быть настроена как для небольшого офиса с 2-3 операторами, так и для крупного многофункционального центра
- **Масштабируемость** - легко добавлять новые терминалы самообслуживания и рабочие места операторов без изменения архитектуры системы. Поддерживается горизонтальное масштабирование для обработки большого количества одновременных запросов
- **Отказоустойчивость** - каждый компонент может работать независимо. При выходе из строя одного терминала или рабочего места остальные продолжают функционировать
- **Интеграция** - возможность подключения внешних систем через REST API: CRM-системы, системы бронирования, уведомления по SMS/email
- **Мультиплатформенность** - работа на любых устройствах с веб-браузером: компьютеры, планшеты, сенсорные киоски

Типовая схема развертывания

В типичной организации система развертывается следующим образом:

1. **Центральный сервер** - размещается в серверной комнате или облаке
2. **Терминалы** - устанавливаются в местах обслуживания клиентов
3. **Рабочие места операторов** - подключаются к центральному серверу
4. **Информационные панели** - размещаются в зонах ожидания

Основные сервисы

Система представляет собой набор микросервисов, которые взаимодействуют друг с другом через REST API и обеспечивают полный цикл работы электронной очереди.

Основные компоненты

- **apigateway** - единая точка входа и маршрутизации запросов. Обеспечивает централизованную обработку всех API-запросов
- **apiservice** - основной бизнес-сервис системы, отвечающий за:
    - Управление талонами (создание, обновление статуса, назначение операторам)
    - Управление услугами и их конфигурацией
    - Обработку действий операторов

    - Предоставление данных для интерфейсов

- **jobs** - сервис фоновых задач и автоматизации, выполняющий:
    - Закрытие рабочего дня и архивирование данных
    - Очистку устаревших записей и временных файлов
    - Проверку активности подключений операторов
- **cron** - планировщик заданий на основе cron-синтаксиса:
    - Запуск регулярных задач по расписанию
    - Выполнение скриптов обслуживания
- **webclient** - фронтенд-сервис для веб-интерфейсов:
    - Обслуживание статических файлов (HTML, CSS, JavaScript)
    - Рендеринг интерфейсов оператора, терминала и панелей
    - Обработка пользовательских сессий
    - Поддержка различных тем оформления

Дополнительные компоненты

- **printservice** - специализированный сервис печати талонов:
    - HTTP API для отправки заданий на печать
    - Интеграция с CUPS (Common Unix Printing System)
    - Поддержка шаблонов талонов в формате HTML/CSS
    - Конвертация в PDF для печати
- **electron-kiosk** - desktop-приложение для терминалов и панелей:
    - Полноэкранный режим киоска без возможности выхода
    - Автозапуск при включении системы
- **electron-operator** - desktop-приложение для операторов:
    - Иконка в системном трее
    - Удобный размер экрана
- **text-to-speech** - сервис озвучивания объявлений. Синтез речи для номеров талонов

Установка сервера

Сервер распространяется в виде набора Docker-контейнеров и compose-файла для

запуска приложения. Установка происходит в несколько этапов и требует базовых знаний работы с командной строкой Linux и Docker.

Этапы установки основного сервера

Требования

Перед установкой необходимо:

1. Установить Docker и Docker Compose
2. Скачать полученный дистрибутив и распаковать его в папку, откуда будет запускаться приложение

Команды рассчитаны на работу в Linux, но для других операционных систем они могут быть похожими или иметь аналоги.
Явных требований к основной операционной системе нет.

1. Загрузка образов приложения

В папке с проектом должна быть директория `.images`, которая содержит основные образы приложения. Нужно импортировать все образы из этой папки с помощью команд:

```
docker load -i .images/apigateway.tar.gz
docker load -i .images/apiservice.tar.gz
docker load -i .images/cron.tar.gz
docker load -i .images/jobs.tar.gz
docker load -i .images/webclient.tar.gz
```

2. Основные настройки БД

Настройка базы данных MongoDB является критически важным этапом установки системы. MongoDB используется для хранения всей информации о талонах, очередях, операторах и статистике.

Настройка учетных данных в docker-compose.yml:

В файле `docker-compose.yml` необходимо указать логин и пароль для подключения к MongoDB в переменных окружения контейнера `mongodb`:

```
environment:
  MONGO_INITDB_ROOT_USERNAME: user
  MONGO_INITDB_ROOT_PASSWORD: password
```

Важно: Не используйте стандартные `user` / `password` в продуктивной среде. Обязательно замените их на надежные учетные данные.

Создание директории для данных MongoDB:

Необходимо создать папку, в которой будет храниться база данных MongoDB на хост-машине:

```
mkdir -p data/mongodb
sudo chown 101 data/mongodb
```

Здесь `101` - это UID пользователя `mongodb`, от которого запускается MongoDB внутри контейнера. Это обеспечивает корректные права доступа к файлам базы данных.

Генерация ключа репликации:

MongoDB требует ключ репликации для работы в режиме replica set, который необходим для некоторых используемых функций. Ключ должен быть сгенерирован и сохранен в файле `config/mongodb/replset.key`. Для этого выполните следующие команды:

```
# Генерация ключа репликации (756 байт base64)
openssl rand -base64 756 > config/mongodb/replset.key

# Установка владельца и прав доступа для ключа
sudo chown 101 config/mongodb/replset.key
sudo chmod 400 config/mongodb/replset.key
```

Права `400` означают, что только владелец файла может его читать, что обеспечивает безопасность ключа.

Запуск контейнера MongoDB:

```
docker-compose up -d mongodb
```

Настройка файла setup.js:

Откройте файл `config/mongodb/setup.js` и настройте основные параметры системы:

`WINDOWS` - общее количество операторов очереди (окон обслуживания) Это определяет, сколько одновременно может работать операторов. Например, для МФЦ с 5 окнами укажите значение 5.

`BUTTONS` - массив объектов, описывающих доступные услуги

Каждая кнопка (услуга) имеет следующие обязательные параметры:

- **`name`** - внутреннее название услуги для системы (например, "consultation")
- **`short_name`** - краткое название для талона (например, "К" для консультации)
- **`display_name`** - полное название для отображения на экране терминала

Пример конфигурации услуг:

```
const BUTTONS = [
    {
        name: "consultation",
        short_name: "К",
        display_name: "Консультация"
    },
    {
        name: "documents",
        short_name: "Д",
        display_name: "Подача документов"
    },
    {
        name: "payment",
        short_name: "О",
        display_name: "Оплата услуг"
    }
];
```

Кроме этого можно указать дополнительные параметры:

`DATABASE` - название основной базы данных для оперативной информации

Важно: это имя должно точно совпадать с параметром `db` в секции `mongodb` конфигурационных файлов сервисов `API Service` и `Jobs`. По умолчанию используется `queue-tickets`.

Инициализация базы данных:

После настройки параметров выполните команды для инициализации:

```
# Инициализация replica set в MongoDB
docker-compose exec mongodb mongosh -u user -p password --quiet --eval '

# Выполнение скрипта настройки базы данных
docker-compose exec mongodb mongosh -u user -p password /data/config/set

# Проверка успешной инициализации
docker-compose exec mongodb mongosh -u user -p password --eval 'rs.statu
```

При успешной инициализации вы увидите информацию о статусе replica set и подтверждение создания базы данных.

3. Настройки приложения

Для каждого сервиса в папке `config` есть файл с настройками. В основном они не требуют дополнительных действий, но иногда требуется поменять параметры. Рассмотрим основные настройки подробно:

Общие настройки для всех сервисов

Каждый сервис умеет работать с сервисом для отслеживания ошибок и трейсинга запросов Sentry. Это полезно для мониторинга работы системы в продуктивной среде. Если у вас есть собственная инсталляция Sentry или облачный сервис и требуется следить за событиями, то нужно в каждом сервисе поменять параметры в блоке `"sentry"`:

```
"sentry": {
    "enable": false,
    "url": "https://setry.org",
    "environment": "production",
    "rate": 0.1
}
```

Параметры конфигурации Sentry:

- `enable` - включение/отключение отправки данных в Sentry
- `url` - URL вашего Sentry сервера
- `environment` - среда развертывания (production, staging, development)
- `rate` - доля событий для отправки (0.1 = 10% событий)

API Gateway

API Gateway является центральной точкой входа для всех запросов к системе. Основные настройки:

Подключение Text-to-Speech сервиса:

В секции `"api"` можно добавить параметр `"tts"` для подключения сервиса озвучивания талонов:

```
"api": {
    "tickets": "http://apiservice:8080",
    "tts": "https://tts.example.com"
}
```

Спецификацию API сервиса можно посмотреть в файле `.spec/tts.yml`, который находится в корне распакованного дистрибутива.

API Service

Это основной сервис системы, который управляет бизнес-логикой. Ключевые настройки:

Подключение к MongoDB: Если изменены имя БД или логин/пароль при настройке MongoDB, нужно поменять параметры в секции `"mongodb"`:

```
"mongodb": {
    "alias": "default",
    "host": "mongodb://mongodb",
    "username": "user",
    "password": "secret",
    "db": "queue-tickets"
}
```

Важно: Название БД в `db` должно соответствовать имени базы данных (`DATABASE`), указанному при настройке в файле `setup.js`.

Jobs

Для этого сервиса, если изменены имя БД или логин/пароль при настройке MongoDB, нужно поменять параметры в секции `"mongodb"`. Кроме того, сервис каждый день собирает отчет и данные в отдельную базу данных для аналитики и статистики. Для этого нужно указать параметры подключения к ней в секции `"mongodb"`.

```
"mongodb": {
    "default": {
        "alias": "default",
        "host": "mongodb://mongodb",
        "username": "user",
        "password": "secret",
        "db": "queue-tickets"
    },
    "stats": {
        "alias": "stats",
        "host": "mongodb://mongodb",
        "username": "user",
        "password": "secret",
        "db": "queue-statistics"
    }
}
```

- `default` - основная база данных для оперативной работы
- `stats` - база данных для аналитики и статистики

Web Client

Сервис веб-интерфейсов поддерживает различные настройки отображения:

Выбор темы оформления:

```
"app": {
    ...
    "theme": "single"
}
```

Cron

Данный сервис служит для запуска действий по расписанию. Расписание задается в файле `config/cron/crontab`. Сейчас там одна задача, которая закрывает день и переносит информацию в БД для аналитики, но можно добавить любые задания. В папке `scripts` можно положить нужные скрипты для вызова.

Формат записи задач:

```
# Минута Час День_месяца Месяц День_недели Команда
0 0 * * * /scripts/close_day.sh
*/15 * * * * /scripts/cleanup.sh
@daily /scripts/backup.sh
```

Поддерживаемые расширения:

- `@yearly` или `@annually` - раз в год
- `@monthly` - раз в месяц
- `@weekly` - раз в неделю
- `@daily` или `@midnight` - раз в день
- `@hourly` - раз в час

Настройка расписания выполняется в стандартном формате `crontab`, но поддерживает расширения через `@`. Подробнее можно посмотреть в документации к проекту cronn.

Примеры полезных задач:

```
# Очистка временных файлов каждые 4 часа
0 */4 * * * /scripts/cleanup_temp.sh

# Генерация отчетов каждый понедельник в 9:00
0 9 * * 1 /scripts/generate_reports.sh

# Резервное копирование каждую ночь в 2:00
0 2 * * * /scripts/backup_database.sh
```

4. Запуск приложения

Сервис запускается на порту 8080. Если требуется изменить порт, то нужно поменять его в файле `docker-compose.yml` в секции `webclient`:

```
ports:
  - "8080:8080"
```

После настройки всех сервисов можно запускать приложение. Для этого нужно выполнить команду:

```
docker-compose up -d
```

Это запустит все сервисы в фоновом режиме. Данные MongoDB будут сохраняться в папке `data/mongodb` на хост-машине, обеспечивая их сохранность между перезапусками контейнеров.

Для остановки приложения выполните команду:

```
docker-compose down
```

Если нужно посмотреть логи, то можно выполнить команду:

```
docker-compose logs -f
```

Далее сервис можно открывать в браузере по адресу `http://localhost:8080`, а точнее для каждого интерфейса доступен следующий адрес:

- Оператор: http://localhost:8080/operator/
- Терминал: http://localhost:8080/terminal/
- Информационная панель: http://localhost:8080/panel/

Дополнительные команды

Получить идентификаторы окон для настройки операторов

Чтобы настроить рабочие места операторов, необходимо получить уникальные ID окон из базы данных. Выполните команду:

```
docker-compose exec mongodb mongosh -u user --quiet --eval 'printjson(db
```

Замените `queue-tickets` на актуальное имя БД, если оно было изменено в `setup.js`

Установка системы печати талонов на терминал

Данный сервис позволяет использовать настроенный принтер для печати талонов на терминале. Он публикует API на локальном порту к которому подключается основной сервис

Этапы установки

Требования

Перед установкой необходимо:

1. Установить Linux на терминал
2. Установить Docker и Docker Compose
3. Установить CUPS и настроить в нем принтер. Включить общий доступ к принтеру
4. Скачать полученный дистрибутив и распаковать его в папку, откуда будет запускаться приложение

1. Загрузка образа приложения

В папке с проектом должна быть директория `.images`, которая содержит основные образы приложения. Нужно импортировать все образы из этой папки с помощью команд:

```
docker load -i .images/controlservice.tar.gz
```

2. Настройка приложения

В папке `config` есть файл с настройками. Необходимо в них указать следующие параметры:

```
"cups": {
    "host": "172.17.0.1",
    "printer": "Tickets",
    "template": "controlservice/ticket/template.html.j2"
}
```

- `host` - адрес сервера CUPS. В данном случае это внутренний адрес хоста с запущенным Docker
- `printer` - имя принтера в CUPS, по которому открыт доступ
- `template` - путь к шаблону печати. Подробнее про изменения шаблона можно почитать в документации

3. Запуск приложения

После настройки сервиса можно запускать приложение. Для этого нужно выполнить команду:

```
docker-compose up -d
```

Это запустит все сервисы в фоновом режиме. Если нужно посмотреть логи, то можно выполнить команду:

```
docker-compose logs -f
```

Далее сервис публикует API по адресу `http://localhost:5004`
Дальше система подключается к данному адресу и печатает талоны на принтере, который был настроен в CUPS

Изменение шаблона талонов

Чтобы использовать свой шаблон достаточно создать файл шаблона в папке `templates`, после чего прописать в `compose.yml` в секции `volumes` путь к папке:

```
volumes:
  - ./templates:/app/templates
```

Далее в настройках приложения указать путь к шаблону, например:

```
"cups": {
    ...
    "template": "/app/templates/ticket.html.j2"
}
```

Подробная настройка шаблонов талонов

Система использует шаблонизатор Jinja2 для создания HTML-разметки талонов, которая затем конвертируется в PDF для печати.

Доступные переменные в шаблоне:

- `ticket` - номер талона (например, "A001")
- `typename` - название услуги

Пример расширенного шаблона:

```
<!DOCTYPE html>
<html lang="ru">
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <style>
        @page {size: 80mm 80mm;}
        body {width: 80mm; height: 80mm; font-size: 8pt; margin-left: -5
        p {margin: 1px;}
        .content {margin: 5px; text-align: center;}
        .logo {font-size: 1.4em; padding-bottom: 10mm;}
        .ticket {font-size: 6em; padding-bottom: -7mm;}
        .type {font-size: 1.3em; padding-bottom: 15mm;}
        .info {font-size: 1.2em;}
        .info span {text-decoration: underline; font-size: 1.2em; font-w
    </style>
</head>
<body>
    <div id="content" class="content">
        <div class="ticket">{{ ticket }}</div>
        <div class="type">{{ typename }}</div>
    </div>
</body>
</html>
```

Ограничения библиотеки xhtml2pdf:

Для формирования используется библиотека xhtml2pdf.
Библиотека накладывает ограничения на шаблон, поэтому при редактировании стоит обратить внимание на ее документацию.

1. **CSS поддержка ограничена:**
    - Не поддерживается flexbox
    - Ограниченная поддержка позиционирования
    - Простые селекторы CSS
2. **Шрифты:**
    - Рекомендуется использовать базовые шрифты
    - Для кириллицы: "DejaVu Sans"
    - Можно подключать внешние шрифты через @font-face
3. **Изображения:**
    - Поддерживаются форматы: PNG, JPEG, GIF
    - Изображения должны быть доступны по абсолютному пути

Устранение проблем с печатью

Принтер не печатает

1. **Проверьте статус принтера:**

```
lpstat -p
lpstat -t
```

1. **Очистите очередь печати:**

```
cancel -a
```

1. **Перезапустите CUPS:**

```
sudo systemctl restart cups
```

Инструкция по использованию web-приложения операторами

Основные принципы работы

Интерфейс оператора - это центральный инструмент для управления потоком клиентов. Он позволяет:

1. **Видеть текущее состояние очереди** - количество ожидающих клиентов по каждой услуге
2. **Управлять своим рабочим местом** - выбирать услуги для обслуживания, ставить паузы
3. **Вызывать клиентов** - системно и последовательно обслуживать очередь
4. **Контролировать общую работу** - запускать и останавливать систему (с паролем)

Понимание логики очереди

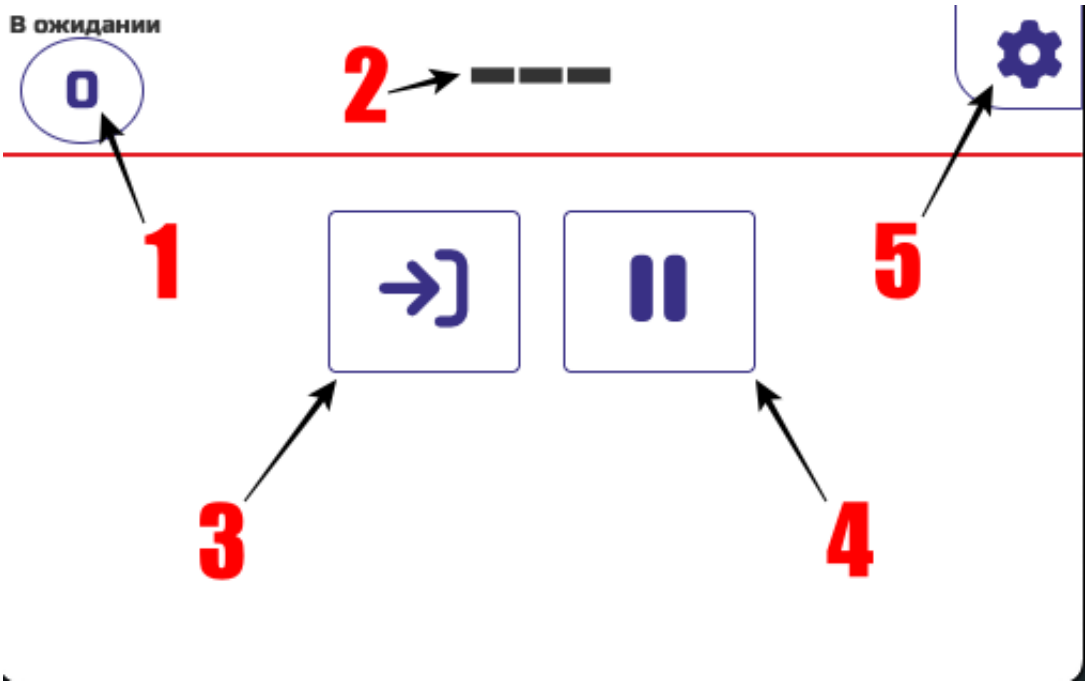
Система работает по следующим принципам:

- **Распределение по услугам** - оператор получает только те талоны, услуги которых он обслуживает
- **Автоматическое назначение** - система сама выбирает следующий талон для вызова

- **Учет пауз** - оператор на паузе не получает новых талонов

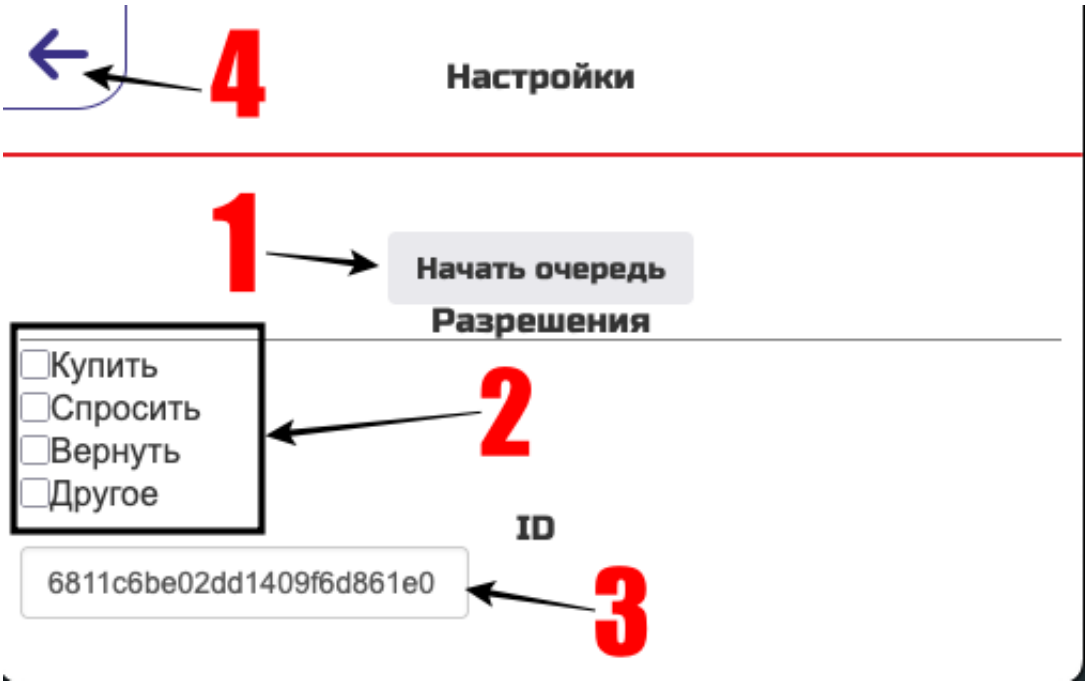
Общий интерфейс

Главный экран



1 - Счетчик талонов в очереди. Показывает общее количество талонов в очереди по выбранным услугам
2 - Текущий активный талон 3 - кнопка "Пригласить следующий талон". Вызывается следующий талон в очереди
4 - кнопка "Пауза". Приостанавливает выдачу талонов для этого оператора
5 - Настройки

Настройки



1 - Кнопка старта/остановки очереди. После нажатия требуется ввести пароль администратора для подтверждения действия
2 - Список услуг. Выбираются услуги, которые будут доступны для вызова талонов
3 - Уникальный идентификатор оператора. Выдается администратором очереди
4 - кнопка "Назад". Возвращает на главный экран

Первоначальная настройка

1. Получить уникальный идентификатор оператора у администратора очереди
2. Ввести его в поле "ID" и нажать `Enter`
3. Дождаться появления списка услуг и выбрать необходимые
4. Нажать кнопку "Начать очередь" если имеется пароль администратора
5. Нажать кнопку "Назад" и дожидаться появления талонов в очереди