1. Общая структура

Предполагается, что система будет работать с одним центральным сервером и сетью экоматов.

1.1 Состав прикладного ПО сервера

1.1.1 Функции сервера

1. Хранение базы данных:
    a. исходная таблица штрих-кодов
    b. таблица пользователей системы
    c. таблица хранения транзакций, баллов
    d. таблица сети экоматов, их состояния
2. Провайдер авторизации пользователей для экоматов
3. Взаимодействие с SMS-шлюзом
4. Интерфейс администратора для просмотра статистики, состояния и управления системой

1.1.2 Описание системного ПО сервера

1. Ubuntu 22.04 LTS
2. PostgreSQL 14
3. Docker CLI
4. Nginx

1.1.3 Описание прикладного ПО сервера

На сервере размещено админское ПО, которое включает в себя веб-интерфейс админки и эндпоинты API, экомат использует это API для работы.
https://admin.ecomats.ru/
Тестовый стенд: https://dev-admin.ecomats.ru/

Таблица штрих-кодов по замыслу должна вестись в оригинальном виде на сервере и автоматически обновляться на всех экоматах в сети. Функцию можно реализовать стандартными средствами PostgreSQL, либо отдельным скриптом.

На сервере реализована отправка СМС пользователям с использованием API сервиса https://smsc.ru.

Обновление приложения

1. Загрузить новый jar в папку `/var/www/apps/prod-ecomat-admin-v1/`
2. Выполнить сборку образа и запустить контейнер:

```
cd /var/www/apps/prod-ecomat-admin-v1
docker compose build
docker compose up -d
```

1.1.4 Список методов API сервера

Адрес сервера
https://admin.ecomats.ru/

Во всех запросах ожидается наличие заголовка `X-Secret-Code` - это уникальный секрет, связанный с экоматом. Нужен для проверки корректности клиента, по нему определяется экомат, с которого делается запрос.
В случае, если экомат с указанным секретом не найден в БД, отправляется ответ `401 Not Authorized`.
В случае любой ошибки (4xx, 5xx) тело ответа выглядит следующим образом:
`{“error”: “тут описание ошибки”}`

Проверка существования пользователя

`GET /api/1.0/user/check/{phone}`
Параметры:
{phone} (string)— номер телефона клиента в формате 7********** (11 цифр)
Варианты ответов:
302 Found — пользователь с таким номером существует
400 Bad Request — неверный запрос
404 Not Found — пользователь с таким номером не существует
Тело ответа:
{ “user_id": "INT" } — идентификатор клиента, только в случае ответа 302

Регистрация пользователя

`GET /api/1.0/user/register/{phone}`
Параметры:
{phone} (string) — номер телефона клиента в формате 7********** (11 цифр)
Варианты ответов:
200 OK — пользователь зарегистрирован, в этот момент пользователю отправляется SMS с кодом
400 Bad Request — неверный запрос
409 Conflict — пользователь уже зарегистрирован

Тело ответа:

{ "pin_length": "INT" } — длина кода, отправленного клиенту в SMS, только в случае ответа 200

Активация пользователя

GET /api/1.0/user/activate/{phone}?pin=…

Параметры:

{phone} (string) — номер телефона клиента в формате 7********** (11 цифр)

pin (string) — код активации из SMS

Варианты ответов:

200 OK — пользователь активирован

400 Bad Request — неверный запрос или не удалось отправить СМС

404 Not Found — пользователь с таким номером и кодом не существует

Тело ответа:

{ “user_id”: "INT" } — идентификатор клиента, только в случае ответа 200

Внесение баллов на счет клиента после сдачи тары

Баллы связываются с конкретным партнёром.

Начисление баллов в партнёрской программе предполагается делать в момент завершения сеанса (клиент).

Также баллы будут отправлены партнеру автоматически спустя час неактивности пользователя (сервер).

GET /api/1.0/deposit/add/{user_id}?barcode_id=…&weight=…&partner=...&uuid=…

Параметры:

{user_id} (int) — идентификатор клиента

barcode_id (int) — идентификатор штрих-кода

weight (int) — вес принятой тары, гр

partner (string)— код партнёра (строка, н-р krasyar)

uuid (string, uuid)— уникальный id запроса

Варианты ответов:

200 OK — баланс пополнен

400 Bad Request — указанная тара не принимается

404 Not Found — не найден пользователь, тара или партнёр

Тело ответа, только в случае ответа 200:

{

“points”: “FLOAT", — количество начисленных баллов

“tare_type”: “STR”, — тип тары

“pending”: “BOOL” — true если баллы начислены за непроверенную тару

}

Получение данных о таре

`GET /api/1.0/tare/get/{code}`
Параметры:
{code} (string) — штрих-код
partner (string)— код партнёра (строка, н-р krasyar)
Варианты ответов:
200 OK — возвращает данные о таре
400 Bad Request — код пустой
404 Not Found — код или партнёр не найден
406 Not Acceptable — тара не принимается (и при этом **не** на проверке)

Тело ответа, только в случае ответа 200:
{
“id”: “INT", — идентификатор тары
“tare_type”: “STR”, — тип тары
“barcode”: “STR” — штрих-код
“acceptable”: “BOOL” — принимается ли эта тара
“points”: “FLOAT", — сколько баллов полагается за данную тару у этого партнёра
“pending”: “BOOL” — true если тара ещё не проверена
}

Добавление данных о таре

`GET /api/1.0/tare/add/{code}?user_id=…&type=…&volume=…&weight=…&name=…`
Параметры:
{code} (string)— штрих-код
user_id (int) — идентификатор клиента, от лица кого делается запрос
type (string) — тип тары
volume (int) — примерный объем тары, мл
weight (int) — вес тары с датчика веса, гр
name (string) — наименование тары, может быть пустым
Варианты ответов:
200 OK — тара добавлена сейчас или была добавлена ранее и уже проверена
400 Bad Request — штрих-код пустой
404 Not Found — не найден указанный клиент
Тело ответа, только в случае ответа 200:
{
“id”: “INT", — идентификатор тары
“tare_type”: “STR”, — тип тары
“code”: “STR” — штрих-код
}

Проверка доступности партнёра

`GET /api/1.0/partner/health/{partnerCode}`
Параметры:
{partnerCode} (string) — строковый идентификатор партнёра, н-р “krasyar”
Варианты ответов:
200 OK — возвращаем данные
Тело ответа, только в случае ответа 200:
{ “ready”: "BOOL" } — доступен ли партнёр для начисления баллов (true) или нет (false)

Получение сведений о партнёре

`GET /api/1.0/partner/get/{partnerCode}`
Параметры:
{partnerCode} (string) — строковый идентификатор партнёра, н-р “krasyar”
Варианты ответов:
200 OK — возвращаем данные
Тело ответа, только в случае ответа 200:
{
“code”: "STR" — код партнёра
“name”: ““STR” — его название
“active”: ”BOOL” — активен ли он
“register_url”: “STR” — ссылка для регистрации в бонусной программе
“logo”: { — логотип
“name”: “STR” —название картинки
“content”: “STR” — картинка в Base64-формате
}
}

Получение списка доступных партнёров

`GET /api/1.0/partner/list`
Параметры:
нет
Варианты ответов:
200 OK — возвращаем данные
Тело ответа, только в случае ответа 200:
{
“partners”: [OBJ_ARRAY] — список партнёров
}
Каждый партнёр:

смотри описание ответа для запроса `/partner/get`

Начисление баллов клиенту для партнёра

Баллы начисляются за все сданные-и-не-начисленные тары, которые сдали в рамках указанного партнёра и с указанного экомата.

`GET /api/1.0/partner/deposit/add/{partnerCode}?user_id=…`
Параметры:
{partnerCode} (string) — строковый идентификатор партнёра, н-р “krasyar”
user_id (int) — идентификатор клиента, которому начисляются баллы
Варианты ответов:
200 OK — возвращаем данные
400 Bad Request — при отправке данных в партнерку возникла ошибка
404 Not Found — не найден указанный партнёр или клиент
Тело ответа, только в случае ответа 200:
{
“points”: “FLOAT” — суммарное количество баллов, отправленных в бонусную систему
“pending_points”: “FLOAT” — суммарное количество непроверенных баллов

Получение состояния экомата, отправляемое периодически (н-р, раз в минуту)

`POST /api/1.0/diagnostic/periodic`
В теле запроса передаётся json, соответствующий классу CurrentState (см. ниже).
Варианты ответов:
200 OK — данные обработаны
400 Bad Request — невалидный экомат

```
class CurrentState {
   private Collect aluCollect, petCollect;
   private State state;
   private Collection<ErrorsHolder.ErrorItem> errors;
   private HwStatus hwStatus;
   private CurrentUser currentUser;
   private CurrentBarcode currentBarcode;
   private SessionData sessionData;
   private ScreenSaverData screenSaverData;
   private List<NativeCmd> nativeCommands;
   private int activeThreads;
}
```

```
class CurrentUser {
    private Long userId;
    private String phone;
    private Balance balance;
    private Partner partner;
    private boolean readyToSkipBonuses;
    private boolean offlineMode;
}
class Balance implements Cloneable {
    private int petCount;
    private float petPoints;
    private int aluCount;
    private float aluPoints;
    private int pendingCount;
    private float pendingPoints;
}
HwSensorsState {
    private State state; // OK, ERROR
    private OkFail tareAlu; // OK, FAIL
    private OkFail tarePet;
    private YesNo transporterD1; // YES, NO
    private YesNo transporterD2;
    private YesNo tarePetFull;
    private YesNo tareAluFull;
    private YesNo topDoorOpened;
    private YesNo bottomDoorOpened;
    private YesNo curtainD1;
    private YesNo curtainD2;
    private YesNo serviceButtonD1;
    private YesNo serviceButtonD2;
    private int weight;
}
class ErrorItem {
    private final String message;
    private Object extra;
    private LocalDateTime time;
}
record Collect(int collected, int capacity, LocalDateTime lastResetAt)
record SessionData(String id, int elapsedMillis)
record ScreenSaverData(int elapsedMillis, int elapsedMillisInSaverMode)
record NativeCmd(String cmd, String output)
```

1.2 Состав прикладного ПО экомата

1.2.1 Функции экомата

1. Авторизация пользователей по номеру телефона (API)
2. Хранение локальной базы штрих-кодов
3. Прием тары, подсчет баллов, отправка данных о транзакции на сервер (API)
4. Взаимодействие с устройствами экомата

1.2.2 Состав системного ПО экомата

1. Debian GNU/Linux 11 (bullseye)
2. PostgreSQL 13

1.2.3 Состав устройств экомата

1. В качестве вычислительного устройства установлена плата x86/amd64 с процессором Intel со встроенным сетевым интерфейсом или USB-модемом
2. Дисплей с тач-функцией, имитирующей нажатие левой клавиши мыши
3. Весы Штрих-Slim, подключены по USB
4. Контроллер приводов, шторки и сминателя*, подключен по USB
5. Контроллер подсветки, подключен по USB
6. Сканеры штрих-кодов, от 1 до 6 штук, подключены по USB

** **сминатель** является доп. оборудованием, не входит в стандартную поставку*

1.2.4 Ecomat Client APP

Интерфейс экомата реализован на базе Web-приложения на Java.
Запускается на адресе: `http://localhost:5001/`
Исполняемый jar-файл находится в папке `/home/ecomat/app/`
Настройки приложения расположены в папке `/home/ecomat/app/config`

Запуск приложения происходит не мгновенно, поэтому экран загрузки сделан в виде html-страницы, которая отображается сразу: `/home/ecomat/app/loader/index.html`
На этой странице делается проверка доступности приложения, за счёт отправки запроса на http://localhost:5001, как только страница начинает отвечать, делается переход по этому адресу, и дальнейшая работа происходит уже в интерфейсе приложения.

Инициализация /

Эта страница открывается при запуске экомата и производит инициализацию и опрос состояния контроллера, в случае успеха переадресует на Главную страницу, иначе на страницу /err/, /maintenance/ или /full/ в зависимости от состояния датчиков.

Главная страница /start

На этой странице экомат находится в режиме ожидания. Здесь у пользователя два выбора — сдать тару /auth или перейти на условия использования /rules.

Условия использования /rules

Текст с правилами работы с экоматом. Ссылка назад, для возврата на предыдущий экран.

Авторизация пользователя /auth

Страница авторизации пользователя, на которой он вводит номер телефона, после проверки по API сервера, если пользователь существует, то переадресует на страницу /reception, иначе на страницу /register.

Регистрация пользователя /register

Страница регистрации, при переходе на нее пользователю сообщается, что он не зарегистрирован, предлагается зарегистрироваться и перейти на /register/complete или отказаться и перейти на главную.

Завершение регистрации и активация пользователя /register/complete

Страница вызывает метод регистрации API сервера и ожидает от пользователя ввода кода из SMS. В случае ввода кода отправляет запрос активации пользователя в API сервера, если успешно — переадресует на /reception, в противном случае просит ввести код повторно. По истечении срока действия кода предлагает отправить его повторно, для этого снова вызывается метод регистрации API сервера.

Ошибка экомата /err

На эту страницу интерфейс переключается в случае некорректного ответа от контроллера экомата, либо отсутствия связи с ним. На этой странице происходит постоянная проверка статуса контроллера, если он становится успешным — экомат переходит на /maintenance.

Экомат заполнен /full

На эту страницу интерфейс переключается в случае, если оба контейнера для приема заполнены. На этой странице происходит постоянная проверка статуса контейнеров, если он становится незаполненным — экомат переходит на главную страницу.

Экомат на обслуживании /maintenance

На эту страницу интерфейс переключается в случае если какая-либо дверь экомата открыта, либо произошел переход из другой страницы. На этой

странице происходит постоянная проверка статуса дверей, если они закрываются — экомат переходит на главную страницу.

Прием тары /reception

Главный экран работы экомата по приему тары.

Чтение штрих-кодов

Чтение штрих-кода происходит при помощи 1-6 сканеров штрих-кодов, которые подключены как Generic HID устройства к системе, то есть имитируют клавиатуру. Чтобы прочитать ввод с клавиатуры в интерфейсе предусмотрено скрытое поле input с установленным фокусом, при вводе какой-либо строки в него делается проверка соответствия строки штрих-коду, если строка распознана как штрих-код, то это значение далее будет использовано для поиска тары в БД.
Этот процесс происходит параллельно общей работе интерфейса на странице приема.

Основной цикл

Основной цикл работы интерфейса в режиме приема основан на состояниях, которые отражают текущий этап приема тары и взаимодействия с пользователем.
Цикл бесконечно выполняется в асинхронной функции, которая меняет интерфейс, сообщения, цвет в зависимости от текущего статуса.

Суть работы статусов заключается в том, чтобы реализовать алгоритм приема тары с соблюдением условий. Например, находясь в статусе готовности к приему ready, существует три условия перехода из этого статуса в другие:
- датчик Д1 занят
- датчик Д2 открыт
- присутствует штрих-код, считанный сканерами.

Если условия выполняются, тогда переходим к статусу detect, и т.п.

По ходу взаимодействия с экоматом пользователь будет видеть информацию, соответствующую текущей фазе.
Предусмотрены ошибки:

- Перевес
- Тара не распознана - если штрих-код корректный но отсутствует в БД, то пользователю доступна кнопка “Добавить тару”, позволяющая отправить свою тару на проверку. Баллы за такую тару будут начислены после успешной проверки модератором.

Получение веса тары

Реализовано через вызов нативного приложения `/home/ecomat/app/native/weightscale`

1.2.4.1 Параметры/настройки приложения

Расположены в папке `/home/ecomat/app/config`.
Часть настроек применены по умолчанию, часть указана явно. Все настройки-по-умолчанию можно переопределить явно, добавив их в файл `/home/ecomat/app/config/application.yml`

При изменении параметров требуется перезапуск приложения, выполнить в консоли: `systemctl restart ecomat`

```
ecomat:
  ecomat-id: 1  # идентификатор (порядковый номер)
  timezone: "Asia/Krasnoyarsk" # таймзона экомата
  debug: false  # режим вывода в консоль служебной информации
  country-code: "7"  # код для номеров телефонов
  phone-length: 10  # длина номеров телефонов без учета кода
  sms-timeout-seconds: 60  # через сколько секунд давать повторно отправить код авторизации
  session-timeout-seconds: 600  # через сколько секунд бездействия возвращать на стартовый экран
  check-barcode-via-api: true  # проверять ли штрих-коды также на сервере, true - да, false - нет (использовать только локальную БД)
  multi-partner: false # доступен ли выбор из нескольких партнёров
  partner: "krasyar" # партнер по умолчанию
  possible-tare-volumes: 250, 300, 330, [...], 2000, 2250 # объемы тары для окна добавления тары

  screen-saver: # настройки скринсейвера
    timeout-seconds: 1200 # через сколько секунд бездействия запускать заставку
    auto-awake-seconds: 7200 # через сколько секунд автоматически прерывать заставку

  api:
    local-url: "http://localhost:5001"  # адрес, на котором запущено приложение
    remote-url: "https://admin.ecomats.ru"  # адрес сервера с API
    secret-code: "тут секретный код"  # служит для проверки подлинности в запросах к API (доступен в админке экомата)
    request-timeout: 15s # таймаут для запросов к АПИ

  fullness:  # количество тары, после которого экомат будет считаться заполненным
    alu: 100  # для алюминия
    pet: 300  # для пластика

  diagnostic: # настройки диагностики, отправляемой на сервер
    periodic:
      enabled: true # включено
      send-period-milles: 60000 # отправка раз в минуту
      native-commands: # нативные команды, результат которых хотим отправить на сервер
        - free -m
```

```
      - journalctl -l -e --no-pager -n 100 -u ecomat

  deposit: # параметры отправки баллов
    pending-resend-period-seconds: 600 # если не удалось отправить, то повторить через это
время

  devices:  # настройки устройств экомата
    controller:  # контроллер, управляющий процессом приёма тары
      debugMode: false  # выводить доп. информацию в консоль (true) или нет (false)
      search:  # список слов, по которым будет определяться это устройство для коннекта
        - "stm32"

    topper:  # светодиодная подсветка
      search:  # список слов, по которым будет определяться это устройство для коннекта
        - "cp2102"

    scales:  # весы
      search:  # список слов, по которым будет определяться это устройство для коннекта
        - "weight"
      zero-weight: 5  # в пределах скольки грамм считается, что тара отсутствует
      max-weight: 100  # более скольки грамм считается “перевес”
      native-app-path: "./native/weightscale"  # путь до приложения, из которого “достаётся” вес

  diagnostic:
    send-period-millis: 60000  # период ms для отправки состояния экомата на сервер
```

1.2.4.2 Настройки подключения к БД

Расположены в файле `/home/ecomat/app/db/application.yml`

Примечание: название БД, имя пользователя и пароль должны совпадать с настроенными в **пункте 2.1.1** данного документа.

```
spring:
  datasource:
    url: "jdbc:postgresql://localhost:5432/ecomat" # адрес для подключения к БД
    username: "ecomat" # имя пользователя
    password: "password-here" # пароль
```

1.2.4.3 Настройка автостарта

Старт браузера: /home/ecomat/.config/autostart/ecomat-chromium.desktop

```
[Desktop Entry]
Type=Application
Exec=chromium --disk-cache-dir=/dev/null --disk-cache-size=1 --kiosk --incognito
--disable-pinch --overscroll-history-navigation=0 --noerrdialogs
--enable-features=OverlayScrollbar --force-device-scale-factor=1
file:///home/ecomat/app/loader/index.html
Hidden=false
NoDisplay=false
X-GNOME-Autostart-enabled=true
Name[en_US]=Ecomat
Name=Экомат
Comment[en_US]=Ecomat kiosk mode (chromium)
Comment=Экомат в режиме киоска (chromium)
```

Сервис приложения: /etc/systemd/system/ecomat.service

```
[Unit]
Description=Ecomat application
After=network.target

[Service]
User=ecomat
Group=www-data
WorkingDirectory=/home/ecomat/app/
ExecStart=java -jar /home/ecomat/app/ecomat.jar

[Install]
WantedBy=multi-user.target
```

1.2.4.4 Ручной запуск экомата

Экомат запускается вместе со стартом системы (см. “Настройка автостарта”)

и работает в виде сервиса.

Запуск: `systemctl start ecomat`

Перезапуск: `systemctl restart ecomat`

Остановка:: `systemctl stop ecomat`

Просмотр логов: `journalctl -l -e --no-pager -f -n 20 -u ecomat` (-f -- follow, -n 20 -- количество строк с конца лога)

2. Установка программного обеспечения

2.1 ПО экомата

2.1.1 Системное ПО

Экомат работает в ОС Debian под пользователем ecomat - домашний каталог /home/ecomat/.

Для работы требуется установить PostgreSQL 13 или новее из официального дистрибутива (репозитория).
Рекомендуемое имя пользователя для подключения к БД - ecomat.
Рекомендуемое название БД - ecomat.

Настройка БД:
```
create database ecomat;
create schema barcode;
create schema stat;
create schema deposit;
```

Также необходимо установить Java 17 из официального дистрибутива (репозитория), чтобы команда java была доступна глобально в системе.
Для проверки выполнить java --version, результат будет примерно такой:
```
openjdk 17.0.6 2023-01-17
```

2.1.2 Прикладное ПО

ПО поставляется в виде дистрибутива.

2.1.2.1 Состав дистрибутива

- /loader - папка содержит начальную страницу загрузки, отображаемую в момент старта ПО
- /native/weightscale - нативная утилита для получения веса с весов
- /splash.png - заставка, которая отображается в момент загрузки ОС
- /install-app.sh - скрипт для скачивания указанной версии приложения;
  получает на вход версию, которую требуется скачать
- /update-to-v1.0.0.sh - скрипт обновляет ПО экомата до версии 1.0.0; также запускает скачанную версию
- /restart-app.sh - скрипт для перезапуска приложения

- /restart-app-ui.sh - скрипт для перезапуска приложения; также запускает браузер с ПО экомата
- /rotate.sh - скрипт для поворота экрана на 90 градусов влево
- /ui.sh - Скрипт для запуска браузера с ПО экомата;
  также запускается терминал с логами, расположенный на заднем плане и не видимый пользователю
- /reset-chromium.sh - скрипт для удаления кэша браузера

2.1.2.2 Процесс установки и запуска ПО

Чтобы подготовить к запуску ПО экомата, необходимо запустить установочный скрипт `install-components.sh` из установочного дистрибутива (архива).
Архив нужно распаковать - можно на флешку либо скопировать на диск экомата.

После успешного выполнения скрипта нужно настроить параметры подключения к БД - см. **пункт 1.2.4.2** данного документа.

Далее можно запускать ПО с помощью скрипта `/home/ecomat/app/restart-app-ui.sh`.

Если браузер с ПО уже запущен, то для перезапуска нужно выполнить скрипт `/home/ecomat/app/restart-app.sh`.

При открытии браузера также запускается терминал с логами, расположенный на заднем плане и не видимый пользователю.

Для обновления можно запустить скрипт, скачивающий новую версию из репозитория ситисенс: `sh install-app.sh 1.0.5`, либо заменить файл `/home/ecomat/app/ecomat.jar`. на новую версию вручную.

2.2 ПО сервера

ПО поставляется в виде дистрибутива.
Серверное ПО работает в контейнерах докера.

Соответственно, на сервере должен быть установлен Docker CLI. Установка производится согласно официальной документации https://docs.docker.com/engine/install/.

2.2.1 Минимальные требования к железу

Процессор: 4 ядра
Оперативная память: 2 Гб

Жесткий диск: 10 Гб

2.2.2 Состав дистрибутива

/database - содержит настройки для контейнера БД
/api - содержит настройки для контейнера серверного ПО (API) экомата
/admin - содержит настройки для контейнера серверного ПО (Админка) экомата
/nginx - содержит настройки для контейнера nginx, направляющего http-трафик в ПО

2.2.3 Процесс установки и запуска ПО

Установка ПО производится путём копирования файлов конфигурации докера из предоставленного дистрибутива на сервер и последующей настройки некоторых параметров.
Предлагаемая папка для размещения файлов дистрибутива: `/var/www/`.

Перед запуском контейнера БД необходимо

- настроить пароль суперпользователя в файле `/database/docker-compose.yml`
- настроить имя пользователя и пароль в файле `/database/init-db/0-init.sql`

Перед запуском контейнера с ПО API экомата необходимо

- запустить контейнер с БД
- авторизоваться в докер-хабе для загрузки образа с ПО:
  `docker login docker.citysense.ru` - для получения логина и пароля обратитесь к разработчикам из СитиСенс
- настроить подключение к БД в файле `/api/config/db/application.yml`:

```
spring:
  datasource:
    url: "jdbc:postgresql://db:5432/ecomats" # изменить при необходимости
    username: "db_user"
    password: "replace-with-password"
```

username и password должны совпадать с настроенными в файле `/database/init-db/0-init.sql`

Для обновления ПО нужно

- актуализировать (увеличить) версию образа в файле `docker-compose.yml`
- загрузить новый образ: `docker compose pull`
- перезапустить контейнер: `docker compose up -d`

Веб-интерфейс администратора/оператора (Админка)

Данная часть функционала является независимой, она позволяет настраивать количество баллов за тару для конкретного партнёра, а также управлять базой данных штрих-кодов, на которую опирается процесс приёма тары.

Процедура “перед запуском” аналогична описанной выше для ПО API экомата.

3. Эксплуатация и устранение неисправностей

Для корректной работы ПО требуется постоянное наличие интернета.

Также **требуется актуализировать базу данных штрих-кодов** - подтверждать или отклонять непроверенную тару, которую сдают пользователи.
Этим занимается специальный сотрудник (оператор в админке), предполагается делать это раз в сутки в рабочее время.

Авторизация в системе реализована посредством отправки смс с кодом через сервис https://smsc.ru, это платная услуга, соответственно нужно следить за балансом в данном сервисе.

3.1 ПО экомата

При возникновении неисправностей необходимо выяснить причину - программная ошибка или аппаратная (физическая).

Если экомат сообщает “Контроллер не отвечает более 15 секунд”, то требуется перезагрузить всю систему (выключить и включить).

При работе с неисправностями и при настройке параметров нужно подключить клавиатуру в свободный usb-порт, открыв верхнюю крышку экомата, чтобы иметь доступ к терминалу.
Переключение на терминал делается с помощью комбинации клавиш Alt+Tab (обычно терминал с отображением журнала событий уже запущен на заднем фоне).
В терминале можно создать новое окно: File – New window.

В случае неправильной логики работы ПО нужно смотреть журнал сервиса `journalctl -l -e --no-pager -f -n 200 -u ecomat`.
Если данных в журнале недостаточно для понимания проблемы, то нужно включить режим отладки: файл `/home/ecomat/app/config/application.yml`, параметр

```
ecomat:
  debug: true
```

и перезапустить приложение: `systemctl restart ecomat`.

В случае выявления ошибок требуется доработка ПО с последующим обновлением.

3.2 ПО сервера

Проверить состояние контейнеров: `docker ps`, `docker stats`

Посмотреть логи: `docker logs ecomat-api -f -n 100`
Перезапустить приложение: `docker restart ecomat-api`

В случае выявления ошибок требуется доработка ПО с последующим обновлением.

3.3 Устранение физических дефектов

Требуется знание физического устройства экоматов, понимание процесса сборки и разборки его частей, чтобы в случае необходимости диагностировать неисправность и устранить её, например путём замены вышедшей из строя детали.

Что может выйти из строя:

- датчики определения тары, в том числе может понадобиться их более точная настройка для лучшего распознавания тары
- весы
- датчики контроля состояния задней “шторки”
- моторчики, вращающие ленту для забора тары
- моторчик, двигающий шторку
- датчик распознавания штрих-кодов
- датчики открытия дверей
- датчики обнаружения контейнеров для сбора тары (прижимаются обручами контейнера)

3.4 Техническая поддержка

Для отслеживания состояния экоматов предусмотрен веб-интерфейс администратора/оператора (админка), расположенный по адресу **https://admin.ecomats.ru**.

В экоматах реализована активная система мониторинга состояния, когда каждый экомат с определённой периодичностью (1 раз в минуту) сообщает серверу своё состояние.

Если экомат сообщил об ошибке или не отвечает какое-то время (5 минут), то админка отправляет почтовые уведомления об этом всем заинтересованным лицам. Адреса для уведомлений указываются в файле `/admin/config/application.yml` в параметре

```
ecomat:
  notifications:
    ecomat-state:
      attention:
        emails:
          - “your-email@address1.ru"
```

На данные уведомления реагирует ответственный сотрудник, который анализирует последнюю полученную от экомата информацию и решает - к чему относится данная проблема. Далее он ставит в известность либо разработчика, либо специалиста по аппаратной части, чтобы те приняли меры по устранению неисправности.
При необходимости, осуществляется выезд к проблемному экомату, чтобы изучить его состояние “на месте”.

4. Разработка и доработка системы

Текущий адрес разработчиков системы: г. Красноярск, пр-т Мира 91, оф. 207.

4.1 Доработка ПО экомата и сервера

Требуется специалист со знанием Java и веб-технологий, фреймворков Spring и Vaadin.
Если он же будет заниматься обновлением на сервере, то нужны минимальные знания Docker.

Существующие проекты ПО рассчитаны на сборку в системе Maven с использованием Java 17 в формат JAR с последующим заворачиванием в докер-образ, основанный на `bellsoft/liberica-openjdk-alpine:17`

Исходные коды расположены в репозитории https://gitlab.citysense.ru/ecomat, это self-hosted репозиторий, расположенный на сервере платформы Beget.

Доступы запрашиваются индивидуально через обращение к менеджеру проекта.

Для тестирования новых версий ПО рекомендуется иметь тестовый экомат, на котором можно будет проверять работоспособность и новое поведение системы.

Также рекомендуется иметь отдельную тестовую точку входа для API, к которой будет подключаться тестовый экомат в процессе доработки и тестирования.

4.2 Доработка конструкции

Требуется специалист, разбирающийся в электротехнике - углубленные знания не требуются, как правило не требуется что-то паять, достаточно понимать целые элементы-блоки, описанные в **пункте 3.3** данного документа.

Если потребуется менять компоненты на что-то иное, тогда потребуется доработка прошивки контроллера, и это скорее всего должен быть отдельный специалист.