УТВЕРЖДАЮ

Генеральный директор
ООО «Экодиджитал»

Л.П. Познанин

«10» ноября 2025 г.

СИСТЕМА ИНТЕГРАЦИИ ПРОДАЖ ДОПОЛНИТЕЛЬНЫХ ПРОДУКТОВ «ЭКОКРОСС»

руководство программиста

лист утверждения

RUS.1247700625787.00002-01 33-ЛУ

руководитель направления
разработки

«10» ноября 2025 г.

2025

Инв. № подл.	Подп. и дата	Взам.инв. №	Инв. № дубл.	Подп. и дата

УТВЕРЖДЁН
RUS.1247700625787.00002-01 33-ЛУ

СИСТЕМА ИНТЕГРАЦИИ ПРОДАЖ ДОПОЛНИТЕЛЬНЫХ ПРОДУКТОВ «ЭКОКРОСС»

руководство программиста

RUS.1247700625787.00002-01 33

листов 18

Подп. и дата	
Инв. № дубл.	
Взам. инв. №	
Подп. и дата	
Инв. № подл.	

2025

АННОТАЦИЯ

Настоящий эксплуатационный документ предназначен для предоставления общих сведений о программе персоналу, отвечающему за обеспечение функционирования специального программного обеспечения.

Документ содержит, в соответствии с ГОСТ 19.504-79, основные сведения о программном обеспечении, необходимом для функционирования программы, функциональном назначении, логической структуре и алгоритмах функционирования составных частей программы, входных и выходных данных.

СОДЕРЖАНИЕ

1. НАЗНАЧЕНИЕ И УСЛОВИЯ ПРИМЕНЕНИЯ ПРОГРАММЫ

Обозначение программы: RUS.1247700625787.00002-01.

Наименование программы: система интеграции продаж дополнительных продуктов «ЭкоКросс», далее – Программа.

Программа предназначена обеспечивать взаимодействие с внешним провайдером, обеспечивающим продажи дополнительных продуктов. В первую очередь, страхование микрокредитов.

Программа написана с использованием языков программирования PHP 8.4, JavaScript с использованием фреймворков Symfony 7.2, Nuxt 3, Vue и технологий PostgeSQL, RabbitMQ, Memcached, k8s.

1.1. УСЛОВИЯ, НЕОБХОДИМЫЕ ДЛЯ ВЫПОЛНЕНИЯ ПРОГРАММЫ

Программа размещается в среде виртуальных машин (ВМ), предоставляемой «Яндекс.Облаком». Функционирование Программы обеспечивается средой управления контейнерами k8s Kubernetes и программным обеспечением для автоматизации развёртывания и управления приложениями в средах с поддержкой контейнеризации Docker, позволяющим «упаковать» приложение со всем его окружением и зависимостями в контейнер.

Платформа «Яндекс.Облако» предоставляет ВМ с гарантированной или частично гарантируемой масштабируемой вычислительной мощностью по модели «инфраструктура как сервис» (IaaS).

В инфраструктуре «Яндекс.Облака» имеются:

- сервис для управления кластерами Kubernetes выполняет написанный код проекта, который доставляется средствами CI/CD (Bamboo);
- сервис для управления кластерами объектно-реляционной СУБД PostgreSQL, являющейся основным хранилищем данных;
- сервис NoSQL СУБД предназначен для обмена данными внутри проекта.
- сервис для управления кластерами Elasticsearch и Kibana предоставляет основное хранилище для всех логов проекта, как логов безопасности и доступа, так и логов приложения;
- сервис для управления в инфраструктуре «Яндекс.Облака» кластерами быстрой in-memory СУБД Redis;
- сервис Yandex Network Load Balancer, который распределяет сетевую нагрузку по «облачным» ресурсам, обеспечивая отказоустойчивость компонентов проекта;
- универсальное масштабируемое «облачное» объектное S3-совместимое хранилище Yandex Object Storage для логов, резервных копий и образов.

Администрирование осуществляется через веб-интерфейс и из консоли Cli из management-сети.

Мониторинг «облака» как провайдера услуг осуществляется самим провайдером услуг.

Мониторинг отдельных компонентов осуществляется в зависимости от самих компонентов, но общим правилом является единая точка хранения всех логов – ELK-кластер в самом «облаке».

Общий мониторинг чувствительных событий по системе и пользователям проводится через сервисы «облака» и дополнительные транспорты.

Все логи пишутся в единое хранилище, настроенное на долговременное хранение.

Работоспособность «облака» критична, его отказ ведёт к отказу всего и сразу.

Доступ к сети «Яндекс.Облака» и к техническим описаниям инфраструктуры, предоставляемой «Яндекс.Облаком», возможен только через VPN и предоставляется только группе администраторов и сотрудникам службы безопасности.

Сервисы программы используют внешние библиотеки. Описание зависимостей с описанием их назначения приведено в документе RUS.1247700625787.00002-01 32 «Система интеграции продаж дополнительных продуктов «ЭкоКросс». Руководство системного программиста».

2. ХАРАКТЕРИСТИКА ПРОГРАММЫ

Программа реализована как API-сервис.

2.1. ВОЗМОЖНЫЕ ВЗАИМОДЕЙСТВИЯ С ДРУГИМИ ПРОГРАММАМИ

Программа взаимодействует с внешними провайдерами дополнительных услуг (продуктов) и внутренней программой «система автоматизации кредитного конвейера микрофинансовой организации «ЭкоКор»».

2.2. ОБЩАЯ ИНФОРМАЦИЯ

Версия API: 1.0.0
Формат данных: JSON
Кодировка: UTF-8
Аутентификация: Bearer Token (JWT)
Базовый URL: https://ecocross.ecfn.ru/ .

2.3. АУТЕНТИФИКАЦИЯ

Для вызова методов InsuranceProvider требуется аутентификация с использованием JWT токена. Токен формируется по секретному ключу, с использованием алгоритма HS256.

Заголовок запроса:

```
Authorization: Bearer {ваш_jwt_токен}
```

2.4. КОДЫ ОТВЕТОВ

Коды ответов приведены в таблице ниже (таблица 1).

таблица 1 — коды ответов

код	описание
200	успешный запрос
400	неверный запрос
401	не авторизован
404	стратегия провайдера не найдена
500	внутренняя ошибка сервера

код	описание
503	сервис недоступен

2.5. МЕТОДЫ InsuranceProvider

2.5.1. POST /insurance-provider/obtain

Описание: выпуск дополнительной услуги (продукта) через внешнего провайдера.
Требуется аутентификация: да (Bearer Token).
Описание тела запроса приведено в таблице ниже (таблица 2).

таблица 2 — тело запроса

параметр	тип	обязательный	описание	пример
companyAlias	string	да	алиас компании для запроса	"ecofinance"
providerName	string	да	название страхового провайдера допустимые значения: "d2", "qrget", "fake"	"d2"
providerProductId	string	да	идентификатор продукта провайдера	"product-123"
orderDate	string (date-time)	да	дата заказа в формате ISO 8601	"2023-10-01T12:00:00Z"
orderId	string	да	уникальный идентификатор заказа	"order-123"
insurancePremium	integer	да	сумма страховой премии	1000
insuranceAmount	integer	да	сумма страхового покрытия	50000
loanAmount	integer	нет	сумма кредита для расчета страхового покрытия и премии	10000
clientFirstName	string	да	имя клиента	"Иван"
clientMiddleName	string	нет	отчество клиента	"Иванович"
clientLastName	string	да	фамилия клиента	"Иванов"

параметр	тип	обязательный	описание	пример
clientBirthday	string (date-time)	да	дата рождения клиента в формате ISO 8601	"1990-01-01T00:00:00Z"
clientAddress	string	да	адрес клиента	"ул. Пушкина д. 10"
clientEmail	string	нет	e-mail клиента	"ivanov@example.com"
clientPhone	string	да	телефон клиента	"+71234567890"
clientPassportSerial	string	да	серия паспорта клиента	"1234"
clientPassportNumber	string	да	номер паспорта клиента	"567890"
clientPassportIssuer	string	да	кем выдан паспорт	"УФМС России"
clientPassportIssueDate	string (date-time)	да	дата выдачи паспорта в формате ISO 8601	"2010-05-15T00:00:00Z"
clientCardPan	string	нет	маскированный номер карты	"555555******5555"

Пример запроса:

```
http
POST /insurance-provider/obtain
Authorization: Bearer ваш_токен
Content-Type: application/json

{
  "companyAlias": "ecofinance",
  "providerName": "d2",
  "providerProductId": "product-123",
  "orderDate": "2023-10-01T12:00:00Z",
  "orderId": "order-123",
  "insurancePremium": 1000,
  "insuranceAmount": 50000,
  "loanAmount": 10000,
  "clientFirstName": "Иван",
  "clientMiddleName": "Иванович",
  "clientLastName": "Иванов",
  "clientBirthday": "1990-01-01T00:00:00Z",
  "clientAddress": "ул. Пушкина д. 10",
  "clientEmail": "ivanov@example.com",
  "clientPhone": "+71234567890",
  "clientPassportSerial": "1234",
  "clientPassportNumber": "567890",
```

```
  "clientPassportIssuer": "УФМС России",
  "clientPassportIssueDate": "2010-05-15T00:00:00Z",
  "clientCardPan": "555555******5555"
}
```

Пример успешного ответа (200):

```
json
{
  "success": true,
  "data": {
    "rawResponse": "{\"status\": \"success\", \"policyNumber\":
\"12345\"}",
    "number": "12345",
    "additionalData": {
      "insurancePremium": 53,
      "realInsurancePremium": null,
      "insuranceAmount": 100000
    }
  },
  "errorCode": null,
  "errorMessage": null,
  "errors": {},
  "notification": ""
}
```

Пример ошибки (400):

```
json
{
  "success": false,
  "data": {},
  "errorCode": 400,
  "errorMessage": "Invalid request to insurance provider.",
  "errors": {},
  "notification": ""
}
```

2.5.2. POST /insurance-provider/revoke

Описание: Аннулирование существующего страхового полиса.
Требуется аутентификация: Да (Bearer Token).

Описание тела запроса приведено в таблице ниже (
таблица 3).

таблица 3 — тело запроса

параметр	тип	обязательный	описание	пример
companyAlias	string	да	алиас компании	"ecofinance"
providerName	string	да	название страхового провайдера допустимые значения: "d2", "qrget", "fake"	"d2"
number	string	да	номер дополнительной услуги для аннулирования	"12345"
revokeDate	string (date-time)	да	дата аннулирования в формате ISO 8601	"2023-10-01T12:00:00Z"

Пример запроса:

```
http
POST /insurance-provider/revoke
Authorization: Bearer ваш_токен
Content-Type: application/json

{
  "companyAlias": "ecofinance",
  "providerName": "d2",
  "number": "12345",
  "revokeDate": "2023-10-01T12:00:00Z"
}
```

Пример успешного ответа (200):

```
json
{
  "success": true,
  "data": {
    "rawResponse": "{\"status\": \"success\", \"revoked\": true}",
    "isSuccess": true
  },
  "errorCode": null,
  "errorMessage": null,
  "errors": {},
  "notification": ""
}
```

Пример ошибки (404):

```
json
{
  "success": false,
  "data": {},
  "errorCode": 404,
  "errorMessage": "Insurance provider strategy not found.",
  "errors": {},
  "notification": ""
}
```

2.6. МОДЕЛИ ДАННЫХ

Структура ответа:

```
json
{
  "success": "boolean",
  "data": "object",
  "errorCode": "integer | null",
  "errorMessage": "string | null",
  "errors": "object",
  "notification": "string"
}
```

Дополнительные данные для obtain:

```
json
{
  "rawResponse": "string (сырой ответ от провайдера)",
  "number": "string (номер полиса)",
  "additionalData": {
    "insurancePremium": "number",
    "realInsurancePremium": "number | null",
    "insuranceAmount": "number"
  }
}
```

Дополнительные данные для revoke:

```
json
{
  "rawResponse": "string (сырой ответ от провайдера)",
  "isSuccess": "boolean (успешно ли аннулирование)"
}
```

2.7. ПРИМЕРЫ ИСПОЛЬЗОВАНИЯ

пример 1: создание дополнительной услуги

```
bash
curl -X POST "https://api.example.com/insurance-provider/obtain" \
  -H "Authorization: Bearer ваш_jwt_токен" \
  -H "Content-Type: application/json" \
  -d '{
    "providerName": "d2",
    "providerProductId": "product-123",
    "orderDate": "2023-10-01T12:00:00Z",
    "orderId": "order-123",
    "insurancePremium": 1000,
    "insuranceAmount": 50000,
    "clientFirstName": "Иван",
    "clientLastName": "Иванов",
    "clientBirthday": "1990-01-01T00:00:00Z",
    "clientAddress": "ул. Пушкина д. 10",
    "clientPhone": "+71234567890",
    "clientPassportSerial": "1234",
    "clientPassportNumber": "567890",
    "clientPassportIssuer": "УФМС России",
    "clientPassportIssueDate": "2010-05-15T00:00:00Z"
  }'
```

пример 2: аннулирование дополнительной услуги

```
bash
curl -X POST "https://api.example.com/insurance-provider/revoke" \
  -H "Authorization: Bearer ваш_jwt_токен" \
  -H "Content-Type: application/json" \
  -d '{
    "providerName": "d2",
    "number": "12345",
    "revokeDate": "2023-10-01T12:00:00Z"
  }'
```

2.8. ОБРАБОТКА ОШИБОК

Коды ошибок и их значения:

400 Bad Request:

- неверный формат запроса;
- отсутствуют обязательные параметры;
- невалидные значения параметров.

404 Not Found:

- стратегия провайдера не найдена;
- провайдер не поддерживается.

500 Internal Server Error:

- внутренняя ошибка сервера;
- проблемы с подключением к БД.

503 Service Unavailable:

- провайдер недоступен;
- проблемы с сетевым подключением.

2.9. ПРИМЕЧАНИЯ

1. Поддерживаемые провайдеры:
- d2 – провайдер D2;
- qrget – провайдер QRGet;
- fake – тестовый провайдер.
2. Формат дат: ISO 8601 (YYYY-MM-DDTHH:MM:SSZ).
3. Валидация: проверяйте обязательные поля перед отправкой запроса.

2.10. ПРОВЕРКА НАЛИЧИЯ НЕПРЕДУСМОТРЕННЫХ ДАННЫХ

Наличие непредусмотренных данных не проверяется.

При появлении в запросе или ответе непредусмотренных данных, они игнорируются.

2.11. РЕЖИМЫ РАБОТЫ ПРОГРАММЫ

Возможные режимы функционирования Программы: штатный и автономный.

В штатном режиме функционирования должна обеспечиваться работа Программы в автоматическом режиме 24 × 7 × 365 (24 часа в день, 7 дней в неделю, 365 дней в году):

- исправно работать обеспечивающая вычислительная и сетевая инфраструктура;
- исправно функционировать системное, базовое и прикладное программное обеспечение;
- выполняться все функции.

Для обеспечения штатного режима функционирования Программы необходимо соблюдение условий эксплуатации программного обеспечения и ВСИ, изложенных в эксплуатационных документах.

В автономном режиме осуществляются техническое обслуживание, реконфигурация, модернизация и совершенствование компонентов Программы, резервное копирование. Все или отдельные функции Программы становятся недоступными и решение задачи автоматизированным способом прекращается.

3. ВХОДНЫЕ И ВЫХОДНЫЕ ДАННЫЕ

Входные и выходные данные организованы в формате JSON.

Описания входных и выходных данных приведены в разделе «характеристика программы».

4. СООБЩЕНИЯ

Сообщения собираются средствами k8s Kubernetes и отправляются в Kibana в «Яндекс.Облаке». Все логи доступны для чтения только из закрытого контура.

Помимо сообщений, логи содержат теги k8s, включающие данные об имени сервиса, позволяющие фильтрацию. В остальном логи не структурированы и не имеют описаний.

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ

В настоящем документе используются следующие обозначения и сокращения:

API – программный интерфейс приложения (Application Programming Interface)
HTTP – протокол передачи гипертекста (HyperText Transfer Protocol)
JSON – текстовый формат обмена данными, основанный на JavaScript (JavaScript Object Notation)
URL – адрес ресурса в сети Интернет (Uniform Resource Locator)
UTF – система кодирования символов стандарта «Юникод» (Unicode Transformation Format)
XML – расширяемый язык разметки (eXtensible Markup Language)
БД – база данных
ВМ – виртуальная машина
СУБД – система управления базами данных

лист регистрации изменений									
номера листов (страниц)					*всего листов (страниц) в док-те*	*№ документа*	*входящий номер сопроводительного документа и дата*	*подпись*	*дата*
изм.	*изме-ненных*	*заме-ненных*	*новых*	*аннули-рован-ных*					