## Техническое задание на разработку Telegram Online Bank на базе API BinCentric

### 1. Введение и цели проекта

**Цель**: Создание полноценного онлайн-банка в Telegram (Telegram Mini App + Bot) для физических лиц с возможностями выпуска и управления виртуальными картами, пополнения, контроля баланса и истории транзакций. fileciteturn2file0L1-L8

### 2. Сфера применения (Scope)

* Клиенты, владеющие аккаунтом Telegram.
* Использование API BinCentric как основного внешнего платёжного сервиса.
* Встроенный KYC-функционал для идентификации пользователей. fileciteturn2file1L3-L10

### 3. Требования пользователей (User Stories)

1. **Регистрация и верификация**: пользователь вводит ФИО, передаёт контакт, проходит KYC и биометрию лица, вводит промокод, оплачивает выпуск карты (мин. 235 USDT). fileciteturn2file1L3-L10
2. **Выпуск карты**: пользователь нажимает «Выпустить карту», получает реквизиты и баланс. fileciteturn2file0L28-L33
3. **Пополнение**: пользователь выбирает метод (наличные, карта, криптовалюта, P2P) и пополняет карту. fileciteturn2file1L69-L77
4. **Просмотр баланса и транзакций**: на главном экране отображается баланс, список карт, последние операции и кнопка «Посмотреть все». fileciteturn2file1L21-L28
5. **Управление картами**: переноминовать, блокировать/разблокировать, закрывать, устанавливать лимиты. fileciteturn2file1L49-L57
6. **Уведомления**: Telegram Bot отправляет уведомления о транзакциях, ошибках и напоминания. fileciteturn2file0L17-L20

### 4. Функциональные требования

#### 4.1 Модуль регистрации и KYC

* **API**: POST /users, GET /users, PUT/DELETE для учётных записей. fileciteturn2file2L53-L56
* Загрузка документов (удостоверение личности).
* Биометрия лица и ввод промокода.

#### 4.2 Модуль карт

* **Выпуск**: POST /cards (виртуальные/физические) fileciteturn2file0L36-L40
* **Информация**: GET /cards/{id} (статус, реквизиты, баланс) fileciteturn2file0L41-L44
* **Блокировка/разблокировка**: POST /cards/{id}/block и /unblock fileciteturn2file0L44-L46
* **Удаление**: DELETE /cards/{id} fileciteturn2file0L48-L50
* **Лимиты**: POST /cards/{id}/limits, GET /cards/{id}/limits fileciteturn2file2L41-L47
* **Tokenize для Apple/Google Pay**: POST /cards/{id}/tokenize. fileciteturn2file2L49-L53

#### 4.3 Модуль транзакций

* **Пополнение**: POST /cards/{id}/fund; карта, наличные, крипта, P2P. fileciteturn2file0L52-L56
* **Снятие**: POST /cards/{id}/withdraw. fileciteturn2file0L56-L59
* **История**: GET /transactions, GET /transactions/{id}. fileciteturn2file0L60-L66

#### 4.4 Webhooks и уведомления

* **Настройка**: POST /webhooks для событий транзакций, блокировок, отказов. fileciteturn2file2L61-L64
* **Рассылка**: Telegram Bot для real-time уведомлений.

#### 4.5 Крипто-кошелек и обмен

* **Депозит**: генерация адреса, QR (USDT, BTC), история депозитов. fileciteturn2file4L29-L34
* **Вывод**: 2FA, комиссия, история выводов. fileciteturn2file4L37-L42
* **Swap**: крипто–крипто, обмен баланса на фиат. fileciteturn2file3L65-L68
* **P2P**: выбор банка, выдача реквизитов, чек, таймаут 15 мин. fileciteturn2file3L59-L66

### 5. Нефункциональные требования

* **Производительность**: время отклика API ≤ 2 с
* **Надежность**: доступность ≥ 99.9%
* **Безопасность**: соответствие KYC/PдDSS, шифрование данных at-rest и in-transit
* **Логирование и мониторинг**: Elastic Stack + Prometheus
* **Масштабируемость**: микросервисная архитектура, Docker/Kubernetes

### 6. Архитектурная схема и технологии

* **Компоненты**:
  + Telegram Mini App (JS/TS)
  + Telegram Bot (Python/Node.js)
  + Backend-сервисы (Go/Python)
  + PostgreSQL + Redis
  + API Gateway
  + CI/CD (GitLab CI)
* **Взаимодействие**: диаграмма потоков (схема будет приложена).

### 7. Внешние интерфейсы

* **BinCentric API** (OpenAPI 3.0) для всего карточного и транзакционного функционала
* **Telegram Bot API** для нотификаций и команд
* **KYC-сервис**: сторонний провайдер через REST

### 8. UI/UX

* Чистый и единообразный дизайн: Quick Actions, Progressive Disclosure
* Контекстные подсказки и валидация форм
* Адаптивность под мобильные устройства

### 9. Критерии приёмки и тест-кейсы

* Все user stories реализованы и подтверждены ручным тестированием
* API-эндпоинты проходят интеграционные тесты ≥ 90% покрытия
* Нагрузочное тестирование: поддержка 100 транзакций/с

### 10. Перечень модулей системы

* Модуль регистрации и верификации (KYC и биометрия)
* Модуль управления картами (выпуск, блокировка, лимиты)
* Модуль транзакций (пополнение, вывод, история)
* Модуль уведомлений (Webhooks и Telegram Bot)
* Модуль крипто-кошелька и обмена (депозит, swap)
* Модуль пользовательского интерфейса (Telegram Mini App)
* Модуль Telegram Bot (команды и общение)
* Модуль API Gateway (маршрутизация запросов)
* Модуль администрирования и мониторинга (панель, логирование)

### 11. Структура проекта

project-root/  
├── backend/  
│ ├── user-service/ # Сервис регистрации и KYC  
│ ├── card-service/ # Сервис управления картами  
│ ├── transaction-service/ # Сервис транзакций  
│ ├── notification-service/ # Сервис уведомлений и webhook  
│ └── crypto-service/ # Сервис крипто-кошелька и обмена  
├── frontend/  
│ ├── telegram-mini-app/ # Telegram Mini App (JS/TS)  
│ └── telegram-bot-ui/ # UI для Telegram Bot (React)  
├── gateway/  
│ └── api-gateway/ # API Gateway (Go/Node.js)  
├── infra/  
│ ├── docker/ # Docker-файлы и образы  
│ ├── kubernetes/ # Манифесты для Kubernetes  
│ └── ci-cd/ # Конфигурации CI/CD (GitLab CI)  
├── docs/  
│ ├── openapi/ # OpenAPI-спецификация BinCentric и собственная  
│ └── diagrams/ # Архитектурные схемы и пользовательские потоки  
└── scripts/  
 ├── setup.sh # Скрипт установки окружения  
 └── deploy.sh # Скрипт деплоя

### 12. Диаграммы взаимодействия модулей

sequenceDiagram  
 participant UI as Telegram Mini App  
 participant Bot as Telegram Bot  
 participant Gateway as API Gateway  
 participant Auth as User Service  
 participant Cards as Card Service  
 participant Tx as Transaction Service  
 participant Crypto as Crypto Service  
 participant Notif as Notification Service  
  
 UI->>Gateway: Выполнение API-запроса (регистрация/операция)  
 Gateway->>Auth: Проверка KYC и авторизация  
 Auth-->>Gateway: Токен/статус  
 Gateway-->>UI: Ответ пользователю  
  
 UI->>Gateway: Запрос выпуска карты  
 Gateway->>Cards: POST /cards  
 Cards-->>Gateway: Детали карты (номер, срок)  
 Gateway-->>UI: Отображение реквизитов карты  
  
 UI->>Gateway: Пополнение карты  
 Gateway->>Tx: POST /cards/{id}/fund  
 Tx-->>Gateway: Подтверждение транзакции  
 Gateway-->>UI: Оповещение об успехе/ошибке  
  
 Tx->>Notif: Webhook событие транзакции  
 Notif-->>Bot: Отправка уведомления в Telegram  
  
 UI->>Gateway: Запрос крипто-адреса  
 Gateway->>Crypto: GET /crypto/address  
 Crypto-->>Gateway: Адрес и QR  
 Gateway-->>UI: Отображение адреса депозита

### 13. Пользовательские сценарии кратко

1. **Регистрация и верификация**
   * **Участники**: Новый пользователь
   * **Описание**: Пользователь запускает бот, вводит ФИО и контакт, загружает документы для KYC и проходит биометрию. После успешного KYC получает уведомление.

* flowchart LR  
   A[Запуск бота] --> B{Введены данные?}  
   B -- Да --> C[Загрузка документов]  
   C --> D[Биометрия]  
   D --> E{KYC успешен?}  
   E -- Да --> F[Уведомление об успешной регистрации]  
   E -- Нет --> G[Запрос повторной верификации]

1. **Выпуск виртуальной карты**
   * **Участники**: Верифицированный пользователь
   * **Описание**: Пользователь выбирает «Выпустить карту», вводит промокод и оплачивает выпуск (235 USDT). Система создаёт карту через API BinCentric и возвращает реквизиты.

* flowchart TD  
   A[Выбор «Выпустить карту»] --> B[Ввод промокода]  
   B --> C[Оплата выпуска]  
   C --> D[Запрос к BinCentric API]  
   D --> E[Получение реквизитов]  
   E --> F[Отображение реквизитов пользователю]

1. **Пополнение карты**
   * **Участники**: Пользователь с активной картой
   * **Описание**: Пользователь нажимает «Пополнить», выбирает метод (Crypto/Card/P2P), вводит сумму и подтверждает транзакцию.

* flowchart LR  
   A[Нажать «Пополнить»] --> B[Выбор метода]  
   B --> C[Ввод суммы]  
   C --> D[Подтверждение]  
   D --> E[Запрос Fund API]  
   E --> F{Успешно?}  
   F -- Да --> G[Уведомление об успешном пополнении]  
   F -- Нет --> H[Оповещение об ошибке]

1. **Просмотр баланса и истории транзакций**
   * **Участники**: Любой пользователь
   * **Описание**: При запуске Mini App/бота отображаются баланс и последние 5 операций. При выборе «Посмотреть все» происходит запрос полного списка транзакций.

* flowchart TD  
   A[Главный экран] --> B[Баланс]  
   A --> C[5 последних операций]  
   C --> D[Нажать «Посмотреть все»]  
   D --> E[GET /transactions]  
   E --> F[Показать полный список]

### 14. Подробные пользовательские сценарии

#### 14.1 Регистрация и верификация

**Основной сценарий:**

1. Пользователь открывает Telegram и запускает Bot.
2. Bot отображает форму ввода ФИО и контактных данных.
3. Пользователь вводит данные и подтверждает.
4. Bot -> User Service: POST /users с данными регистрации.
5. User Service сохраняет данные и возвращает статус 201.
6. Bot уведомляет пользователя об успешном вводе и запрашивает загрузку документа.
7. Пользователь загружает фото удостоверения.
8. Bot -> User Service: PUT /users/{id}/documents.
9. User Service -> Внешний KYC-API: POST /verify-id.
10. KYC-API возвращает результат проверки.
11. User Service обновляет статус KYC и возвращает результат.
12. Bot уведомляет пользователя об успешном прохождении KYC.

**Альтернативные шаги (KYC неуспешен):**

* Если на шаге 9 KYC-API отклоняет документ, Bot отображает сообщение и предлагает повторную загрузку.

#### 14.2 Выпуск виртуальной карты

**Основной сценарий:**

1. Верифицированный пользователь в Mini App нажимает «Выпустить карту».
2. UI отображает форму ввода промокода.
3. Пользователь вводит промокод и нажимает «Подтвердить».
4. Mini App -> API Gateway: POST /cards с промокодом и userId.
5. API Gateway -> Card Service: обработка запроса.
6. Card Service -> BinCentric API: POST /cards.
7. BinCentric API возвращает данные карты (номер, срок действия).
8. Card Service сохраняет данные и возвращает их через Gateway.
9. Mini App отображает реквизиты карты.

**Альтернативные шаги:**

* Некорректный промокод: UI отображает ошибку и возвращается к вводу.
* Ошибка API: Card Service возвращает ошибку, Mini App показывает сообщение об ошибке.

#### 14.3 Пополнение карты

**Основной сценарий:**

1. Пользователь выбирает «Пополнить карту».
2. UI предлагает выбрать метод (Crypto/Card/P2P).
3. Пользователь выбирает метод и вводит сумму.
4. UI -> API Gateway: POST /cards/{id}/fund.
5. API Gateway -> Transaction Service: создание транзакции.
6. Transaction Service -> BinCentric API: POST /cards/{id}/fund.
7. BinCentric API возвращает статус транзакции.
8. Transaction Service сохраняет результат и генерирует webhook событие.
9. Notification Service через webhook отправляет уведомление через Bot.
10. API Gateway возвращает результат UI.
11. UI отображает уведомление об успешном пополнении.

**Альтернативные шаги:**

* Ошибка транзакции: Transaction Service сохраняет ошибку, Notification Service уведомляет пользователя о сбое.

#### 14.4 Просмотр баланса и истории транзакций

**Основной сценарий:**

1. Пользователь открывает Mini App или Bot.
2. UI -> API Gateway: GET /cards.
3. API Gateway -> Card Service: получение списка карт и балансов.
4. Card Service возвращает данные.
5. API Gateway возвращает результаты UI.
6. UI отображает баланс и последние 5 операций.
7. Пользователь нажимает «Посмотреть все».
8. UI -> API Gateway: GET /transactions.
9. API Gateway -> Transaction Service: получение полного списка.
10. Transaction Service возвращает данные.
11. UI отображает полную историю транзакций.

#### 14.5 Управление картами

**Основной сценарий:**

1. Пользователь выбирает карту и нажимает «Управление».
2. UI отображает опции: Переименовать, Блокировать, Установить лимиты. 3a. **Переименование:** 4a. UI запрашивает новое имя. 5a. UI -> API Gateway: PUT /cards/{id} с новым именем. 6a. API Gateway -> Card Service: обновление имени. 7a. Card Service сохраняет изменения и возвращает обновлённые данные. 8a. UI отображает новое имя карты. 3b. **Блокировка/разблокировка:** 4b. UI -> API Gateway: POST /cards/{id}/block или /unblock. 5b. API Gateway -> Card Service: вызов BinCentric API блокировки. 6b. Card Service возвращает статус операции. 7b. UI отображает результат. 3c. **Установка лимитов:** 4c. UI запрашивает лимиты (дневной, месячный). 5c. UI -> API Gateway: POST /cards/{id}/limits. 6c. API Gateway -> Card Service: сохранение лимитов. 7c. Card Service возвращает новые лимиты. 8c. UI отображает подтверждение.

### 14.6 Альтернативные сценарии и ошибки

* Сбой сети: UI показывает сообщение об отсутствии соединения.
* Превышение лимитов: API возвращает ошибку, UI уведомляет пользователя.
* Некорректный ввод: формы показывают валидацию и подсказки.

$1 предоставленных материалов и требований.\*