# 📌 Описание банковского приложения (простым языком)

Это мини-банковское приложение, которое позволяет:

✅ Регистрировать клиентов (ФИО, телефон, логин, пароль).

✅ Авторизоваться и работать в системе.

✅ Создавать счета и пополнять баланс.

✅ Переводить деньги внутри банка по номеру телефона или счета.

✅ Собирать метрики (количество запросов, время выполнения).

Приложение написано на Spring Boot, хранит данные в памяти (без БД) и имеет REST API для работы с пользователями, счетами и переводами.

📌 Что делает каждый класс?

## 1️. Регистрация и авторизация

| **Класс** | **Описание** |
| --- | --- |
| AuthController | Обрабатывает запросы регистрации (/auth/register), входа (/auth/login) и выхода (/auth/logout). |
| ClientService | Логика регистрации и авторизации пользователей. |
| ClientRepository | Хранит список клиентов в памяти, позволяет их искать по username. |
| SessionManager | Запоминает, какой клиент сейчас авторизован, чтобы все действия выполнялись от его имени. |

## 2️. Работа со счетами

| **Класс** | **Описание** |
| --- | --- |
| Account | Описывает банковский счет: номер счета, номер карты, баланс. |
| AccountService | Позволяет создавать счета для клиентов. |
| AccountRepository | Хранит счета в памяти. |

## 3️. Переводы денег

| **Класс** | **Описание** |
| --- | --- |
| TransactionController | Обрабатывает переводы: список клиентов (/transactions/clients), выбор получателя (/transactions/select-recipient), перевод (/transactions/transfer). |
| MetricsInterceptor | Подсчитывает количество запросов и время их обработки (нужно для мониторинга). |

## 4️. Генерация тестовых данных

| **Класс** | **Описание** |
| --- | --- |
| TestDataInitializer | Создает 10 случайных клиентов с логинами, паролями и счетами при запуске приложения. |

## 5️. (Не реализовано) Мониторинг (Prometheus, метрики) – не реализовано

| **Класс** | **Описание** |
| --- | --- |
| MetricsInterceptor | Записывает количество запросов и время их выполнения. |
| WebConfig | Подключает MetricsInterceptor ко всем HTTP-запросам. |

📌 Итог

* Приложение позволяет регистрироваться, создавать счета и переводить деньги.
* Каждый клиент должен сначала авторизоваться, затем он может работать со своими счетами.
* При переводе можно выбрать получателя и счет, с которого списать деньги.
* Система автоматически считает запросы и замеряет их время выполнения.

# 📌 Подробное описание методов в каждом классе

Здесь кратко описаны все методы, их назначение и входные/выходные данные.

## 1️. AuthController (Регистрация и авторизация)

📍 Файл: com.bankapp.controller.AuthController

| **Метод** | **Описание** | **Запрос** | **Ответ** |
| --- | --- | --- | --- |
| register(String fullName, String phone, String username, String password) | Регистрирует нового клиента. | POST /auth/register  Параметры: fullName, phone, username, password | JSON-объект клиента |
| login(String username, String password) | Авторизует клиента, запоминая его в SessionManager. | POST /auth/login  Параметры: username, password | ✅ Успешный вход: username или ❌ Ошибка: Неверный логин или пароль |
| logout() | Завершает сессию клиента. | POST /auth/logout | ✅ Успешный выход |

## 2️. ClientService (Логика работы с клиентами)

📍 Файл: com.bankapp.service.ClientService

| **Метод** | **Описание** | **Вход** | **Выход** |
| --- | --- | --- | --- |
| register(String fullName, String phone, String username, String password) | Создаёт нового клиента и сохраняет его в ClientRepository. | fullName, phone, username, password | Объект Client |
| login(String username, String password) | Ищет клиента по username, проверяет пароль. | username, password | Optional<Client> (если клиент найден) |

## 3️. ClientRepository (Хранилище клиентов)

📍 Файл: com.bankapp.repository.ClientRepository

| **Метод** | **Описание** | **Вход** | **Выход** |
| --- | --- | --- | --- |
| save(Client client) | Сохраняет клиента в память. | Client | Client |
| findByUsername(String username) | Ищет клиента по логину. | username | Optional<Client> |
| findById(String id) | Ищет клиента по id. | id | Optional<Client> |
| getAllClients() | Возвращает всех клиентов. | – | Collection<Client> |

## 4️. SessionManager (Хранение текущего пользователя)

📍 Файл: com.bankapp.util.SessionManager

| **Метод** | **Описание** | **Вход** | **Выход** |
| --- | --- | --- | --- |
| login(Client client) | Запоминает авторизованного клиента. | Client | – |
| getLoggedInClient() | Возвращает текущего авторизованного клиента. | – | Client |
| logout() | Очищает авторизованного клиента. | – | – |
| isLoggedIn() | Проверяет, есть ли активный пользователь. | – | true/false |

## 5️. Account (Модель банковского счета)

📍 Файл: com.bankapp.model.Account

| **Поле** | **Описание** |
| --- | --- |
| id | Уникальный идентификатор счета |
| accountNumber | Номер счета (генерируется автоматически) |
| cardNumber | Номер карты (генерируется автоматически) |
| balance | Баланс счета |

## 6️. AccountService (Логика работы со счетами)

📍 Файл: com.bankapp.service.AccountService

| **Метод** | **Описание** | **Вход** | **Выход** |
| --- | --- | --- | --- |
| createAccount(String clientId) | Создаёт новый счёт для клиента. | clientId | Объект Account |
| findByNumber(String number) | Ищет счёт по номеру. | accountNumber | Optional<Account> |

## 7️. AccountRepository (Хранилище счетов)

📍 Файл: com.bankapp.repository.AccountRepository

| **Метод** | **Описание** | **Вход** | **Выход** |
| --- | --- | --- | --- |
| save(Account account) | Сохраняет счет в память. | Account | Account |
| findByNumber(String number) | Ищет счет по номеру. | accountNumber | Optional<Account> |

## 8️. (Не реализовано) TransactionController (Переводы)

📍 Файл: com.bankapp.controller.TransactionController

| **Метод** | **Описание** | **Запрос** | **Ответ** |
| --- | --- | --- | --- |
| getAllClients() | Возвращает всех клиентов перед переводом. | GET /transactions/clients | JSON-массив клиентов |
| selectRecipient(String phone, String accountNumber) | Выбирает клиента-получателя перевода. | POST /transactions/select-recipient  Параметры: phone, accountNumber | ✅ Получатель выбран: username или ❌ Ошибка |
| transfer(double amount) | Переводит деньги от авторизованного клиента к выбранному получателю. | POST /transactions/transfer  Параметры: amount | ✅ Перевод завершен! или ❌ Ошибка: Недостаточно средств |

## 9️. (Не реализовано) MetricsInterceptor (Мониторинг запросов)

📍 Файл: com.bankapp.config.MetricsInterceptor

| **Метод** | **Описание** |
| --- | --- |
| preHandle() | Запоминает время начала запроса |
| afterCompletion() | Считает количество запросов и время выполнения |

## 🔟 WebConfig (Подключение MetricsInterceptor)

📍 Файл: com.bankapp.config.WebConfig

| **Метод** | **Описание** |
| --- | --- |
| addInterceptors() | Добавляет MetricsInterceptor во все запросы |

## 1️1️. TestDataInitializer (Генерация тестовых данных)

📍 Файл: com.bankapp.util.TestDataInitializer

| **Метод** | **Описание** |
| --- | --- |
| run(String... args) | Создаёт 10 тестовых клиентов с логинами, паролями и случайными счетами |

📌 Итог

1. Клиенты могут регистрироваться, входить в систему, создавать счета и переводить деньги.
2. Все данные хранятся в памяти (ClientRepository, AccountRepository).
3. Все переводы происходят внутри приложения.
4. Метрики собираются через MetricsInterceptor.
5. Тестовые пользователи создаются автоматически (TestDataInitializer).