Документация к проекту Secure Bank System

Бойко Руслан и Осипов Илья

2 декабря 2025 г.

Аннотация

Документация описывает архитектуру, функциональные и нефункциональные требования, а также краткое пользовательское руководство для Secure Bank System – многопользовательской банковской системы с системой безопасности, верификации и шифрования операций.

Содержание

1 Функциональные требования

1.1 Управление клиентами

Система обеспечивает полный цикл управления клиентскими учетными записями:

- Регистрация новых клиентов с обязательной проверкой паспортных данных на уникальность;
- Аутентификация пользователей по уникальному идентификатору учетной записи и паролю;
- Процедура верификации клиентов уполномоченными сотрудниками службы безопасности

1.2 Управление банковскими счетами

Реализована система управления счетами различных категорий:

- Создание и ведение счетов следующих типов:
  - Сберегательные счета (Savings)
  - Расчётные счета (Checking)
  - Кредитные счета (Credit)
  - Депозитные счета (Deposit)
- Предоставление информации о текущем балансе и полной истории транзакций по каждому счёту
- Установка и управление кредитными лимитами для счетов кредитного типа

1.3 Банковские операции

Система поддерживает проведение стандартных банковских операций:

- Операции пополнения счетов (deposit)
- Операции снятия денежных средств (withdraw) с автоматической проверкой доступных лимитов
- Переводы денежных средств между счетами (transfer) в рамках клиентов системы;
- Механизм одобрения/отклонения крупных операций службой безопасности банка.

1.4 Система безопасности

Реализованы многоуровневые механизмы защиты:

- Шифрование всех данных, хранимых в базе данных системы;
- Двухуровневая система аутентификации (клиенты банка / сотрудники банка);
- Система очередей для одобрения операций, превышающих установленные лимиты;
- Специальные ограничения для новых и неверифицированных пользователей.

1.5 Административные функции

Для сотрудников банка предусмотрены административные функции:

- Просмотр и обработка операций, ожидающих одобрения;
- Верификация новых клиентов банка;
- Управление процентными ставками по кредитным и депозитным продуктам (в процессе реализации);

2 Системные требования

2.1 Требования к производительности

- Время отклика системы не должно превышать 100 мс для стандартных банковских операций зачисления, перевода и снятия;
- Система должна поддерживать несколько одновременных клиентских подключений (и параллельную обработку запросов);
- Эффективное использование оперативной памяти и ресурсов процессора.

2.2 Требования к безопасности

- Обязательное шифрование всех конфиденциальных данных при хранении
- Применение алгоритмов хеширования для хранения паролей пользователей
- Комплексная валидация всех входных данных и параметров операций
- Реализация механизма управления сессиями с автоматическим завершением по тайм-ауту

2.3 Требования к надёжности

- Автоматическое сохранение состояния системы при каждом изменении данных;
- Возможность восстановления работоспособности после аварийных сбоев;
- Регулярное резервное копирование базы данных с возможностью восстановления.

2.4 Требования к масштабируемости

- Модульная архитектура системы, позволяющая добавлять новые функциональные компоненты
- Четкое разделение клиентской и серверной частей системы
- Поддержка многопоточности для обработки параллельных запросов

2.5 Требования к совместимости

- Кроссплатформенная работа на операционных системах семейства macOS (Linux и Windows в процессе отладки);
- Использование стандартных сетевых протоколов TCP/IP для клиент-серверного взаимодействия;
- Независимость от внешних систем управления базами данных.

2.6 Технические требования

Тестирование программы производилось на следующем сетапе:

- Язык программирования: C++17 или выше
- Компилятор: GCC 9.0+, Clang 10.0+, CMake 4.2+ или MSVC 2019+
- Минимальный объем оперативной памяти: 8 ГБ
- Минимальное дисковое пространство: 128 ГБ для установки и работы
- Сетевой интерфейс для клиент-серверного взаимодействия

3 Архитектурная документация

Secure Bank System реализована по многоуровневой клиент-серверной архитектуре с четким разделением ответственности между компонентами. Архитектура следует принципам модульности и инкапсуляции, что обеспечивает высокую сопровождаемость и масштабируемость системы.

Ключевые архитектурные решения:

- Трехуровневая архитектура: клиентский уровень, серверный уровень, уровень данных;
- Событийно-ориентированная обработка: асинхронная обработка клиентских подключений;
- Многопоточность: изолированные потоки для каждого клиентского соединения;
- Пассивный объект базы данных: инкапсуляция логики хранения и сериализации.

Дерево проекта:

```
├── CMakeLists.txt
├── Makefile
├── data
│   ├── accounts.dat
│   ├── accounts.dat.settings
│   └── verification_queue.dat
├── src
│   ├── account.cpp
│   ├── account.h
│   ├── client.cpp
│   ├── client.h
│   ├── crypto.cpp
│   ├── crypto.h
│   ├── database.cpp
│   ├── database.h
│   ├── init_database.cpp
│   ├── main_client.cpp
│   ├── main_server.cpp
│   ├── server.cpp
│   ├── server.h
│   └── view_database.cpp
└── tests
    └── test_bank_system.cpp
```

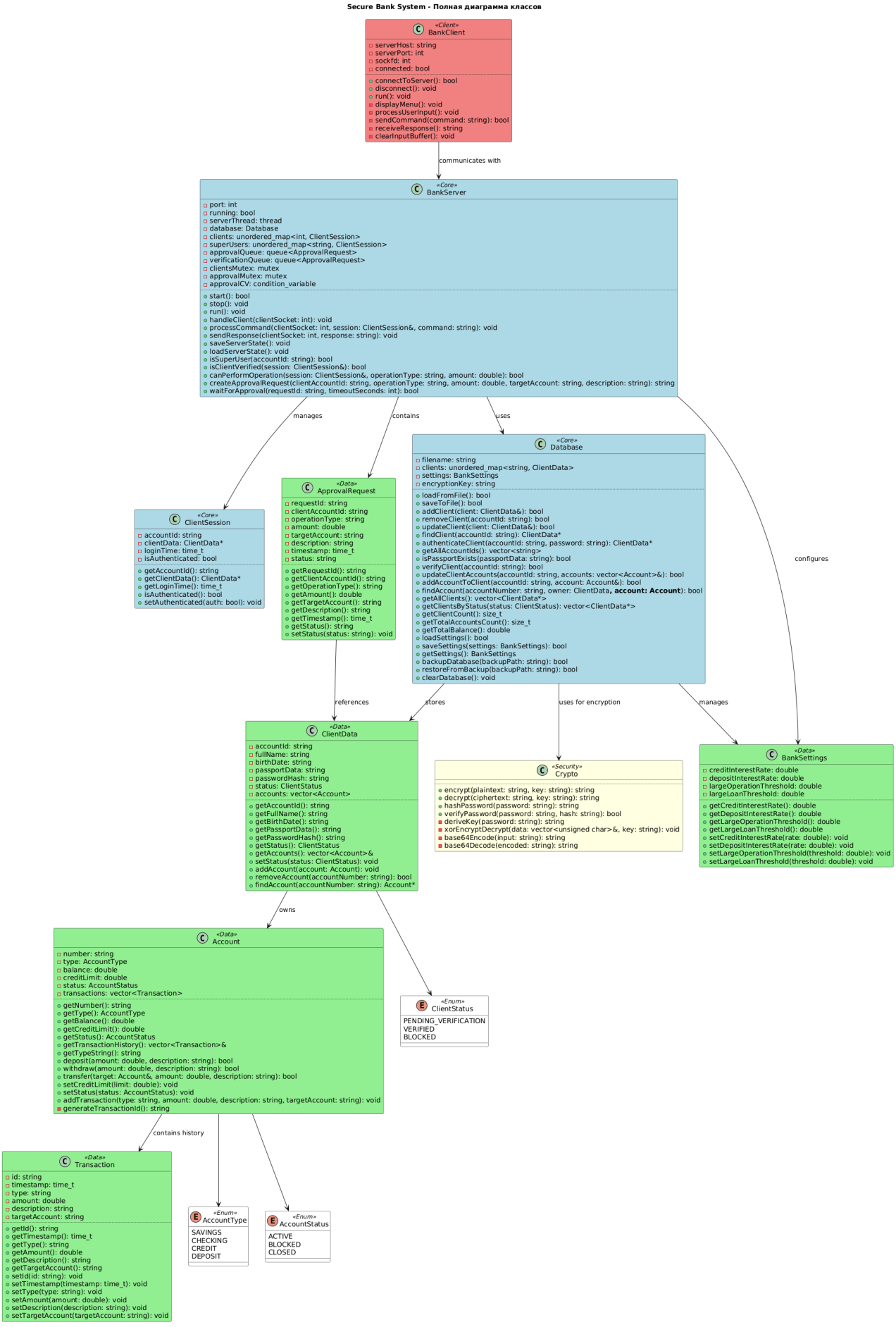

Рис. 1: Диаграмма классов системы

3.1 Компонентная архитектура

Программный код можно разделить на следующие ключевые компоненты:

- **Client Application** – клиентское приложение;
- **Bank Server** – многопоточный сервер с системой сессий;
- **Database Layer** – слой работы с данными;
- **Security Module** – модуль безопасности и шифрования.

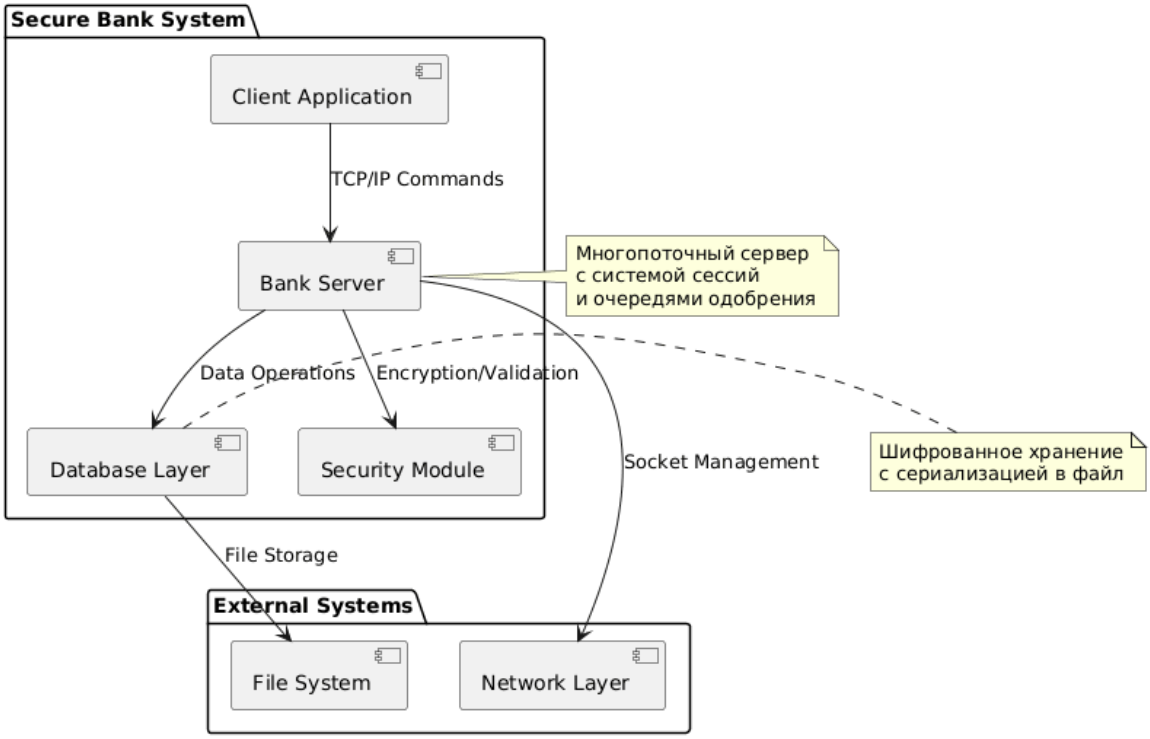

Рис. 2: Архитектурная диаграмма системы

3.2 Логика обработки запросов

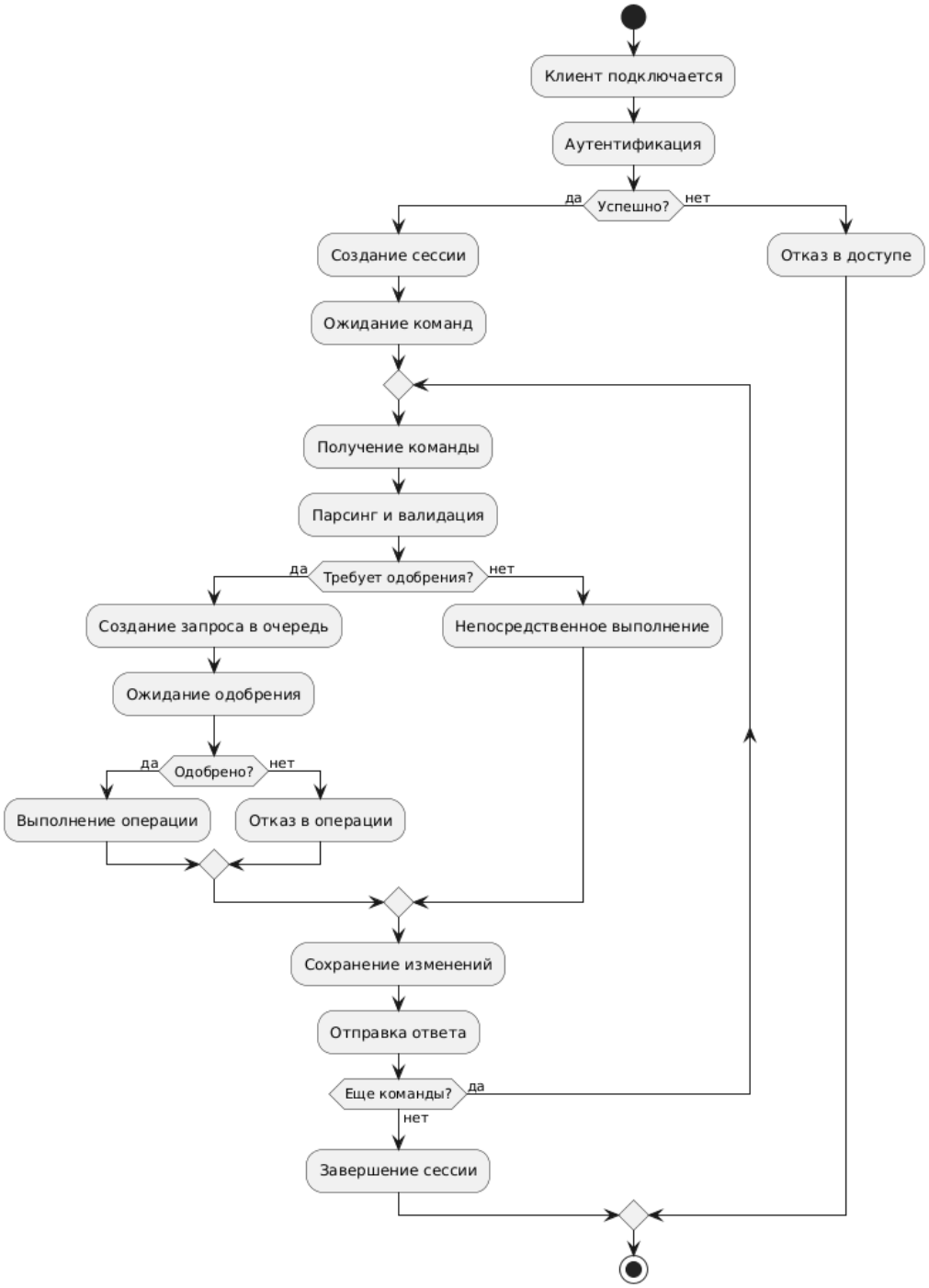

Рис. 3: Логика обработки клиентских запросов

3.3 Архитектурная диаграмма компонент

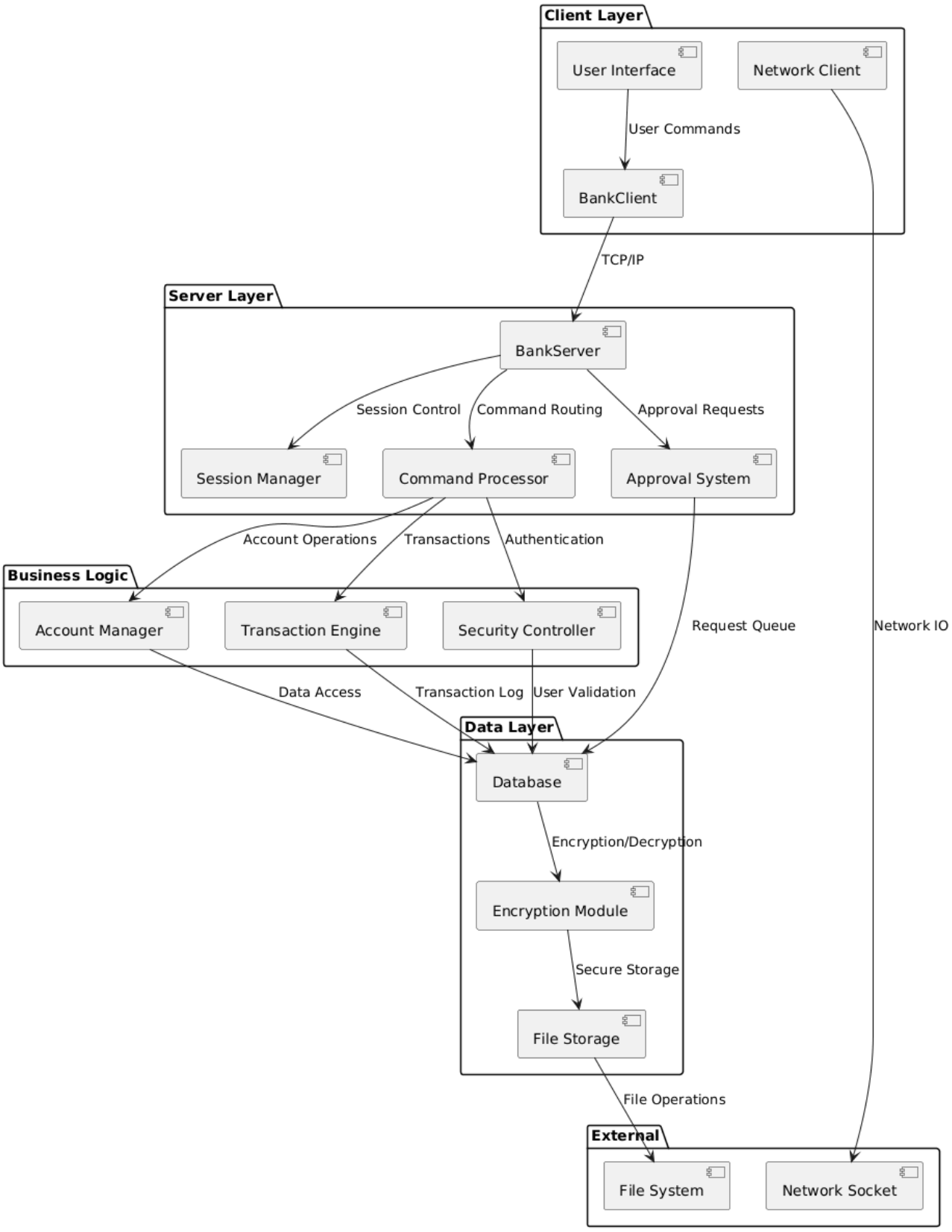

Рис. 4: Диаграмма компонент системы

3.4 Диаграмма последовательности операций

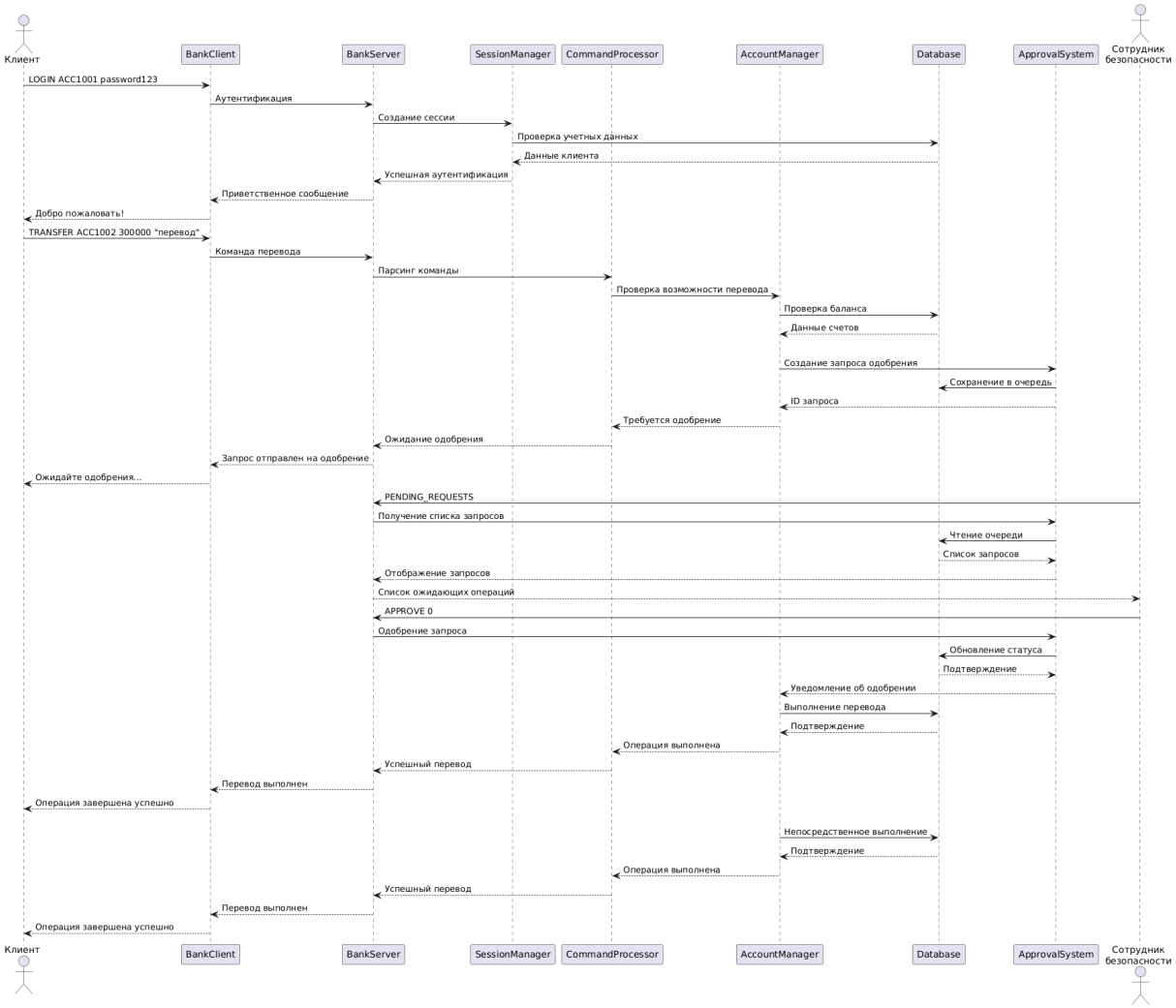

Рис. 5: Последовательность выполнения операции перевода средств

4 Пользовательская документация

Инициализация системы

```
# Компиляция и настройка
make setup
# Запуск сервера
make run_server
# Запуск клиента в( другом терминале)
make run_client
```

Тестовые учетные записи

```
Обычный клиент:          ACC1001 / password123
Обычный клиент:          ACC1002 / qwerty456
Неверифицированный:      ACC1003 / test789
Сотрудник безопасности: SUPER001 / superpass123
```

4.1 Основные команды

4.1.1 Команды без авторизации:

```
RATES                                                  # Просмотр ставок
REGISTER ФИО 1990-05-15 4510123456 password123 # Регистрация нового пользователя
LOGIN ACC1001 password123                              # Войти в аккаунт
SUPERLOGIN SUPER001 superpass123                       # Войти в акккаунт банковского
    сотрудника
HELP                                                   # Справка
```

4.1.2 Команды после авторизации клиента:

```
ACCOUNTS                 # Список счетов
DEPOSIT 1000             # Пополнение на первый счёт 1000 тугриков
WITHDRAW 500             # Снятие с первого счёта 500 тугриков
TRANSFER ACC1002 200     # Перевод пользователю ACC1002 200 тугриков
HISTORY 0                # История операций
CREATE_ACCOUNT 0         # 0=Savings, 1=Checking, 2=Credit, 3=Deposit
INFO                     # Информация о клиенте
LOGOUT                   # Выход из аккаунта
```

4.1.3 Команды сотрудника безопасности:

```
PENDING_REQUESTS        # Ожидающие операции
PENDING_VERIFICATIONS   # Ожидающие верификации
APPROVE 0               # Одобрить запрос 0
REJECT 1                # Отклонить запрос 1
VERIFY 0                # Верифицировать клиента 0
SET_RATES 12.0 6.5      # Установить ставки
SETTINGS                # Текущие настройки
```

5 Пример пайплайна работы

5.1 Стандартная операция (пополнение счета)

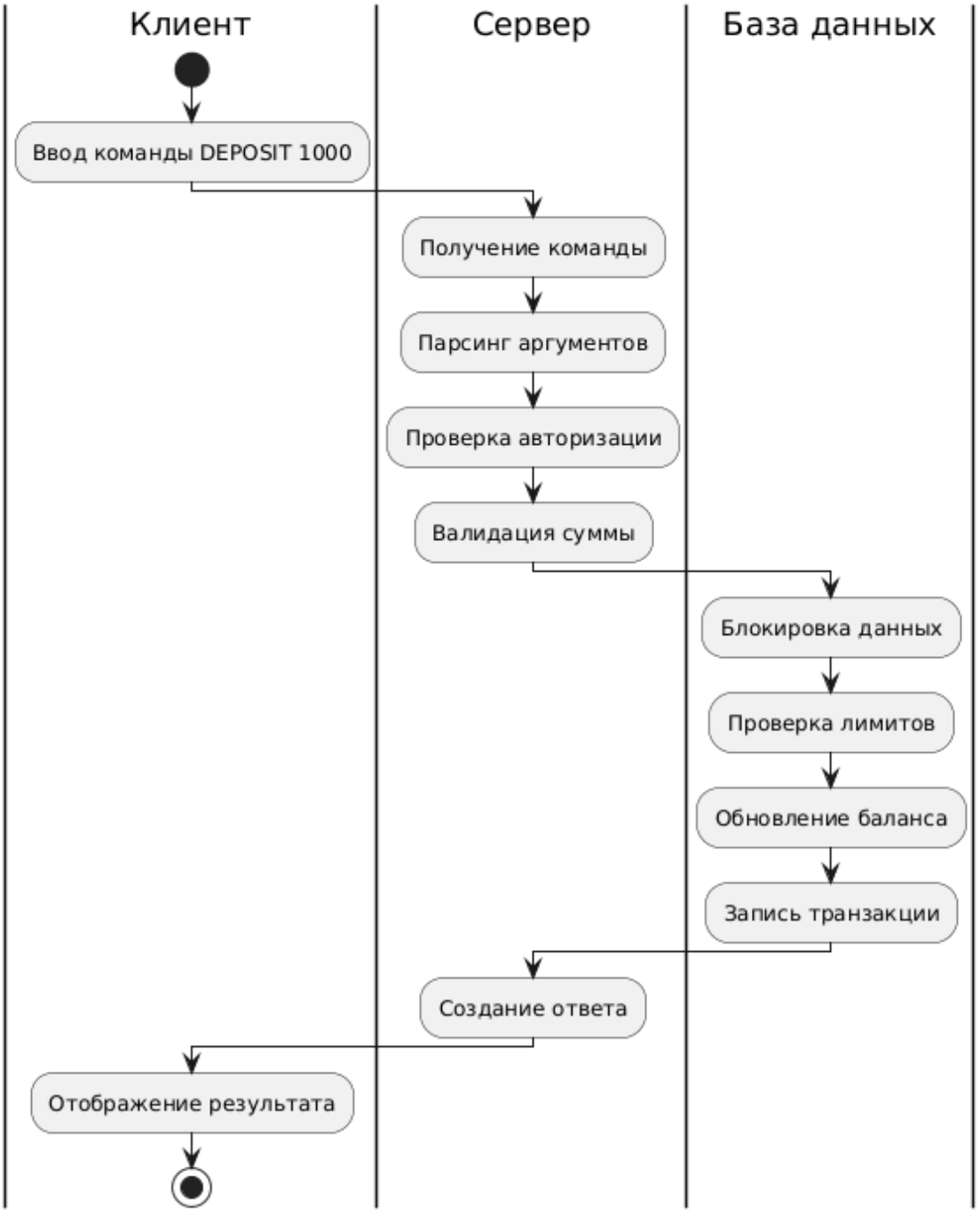

Рис. 6: Пайплайн выполнения стандартной операции пополнения счета

5.2 Операция с одобрением (крупный перевод)

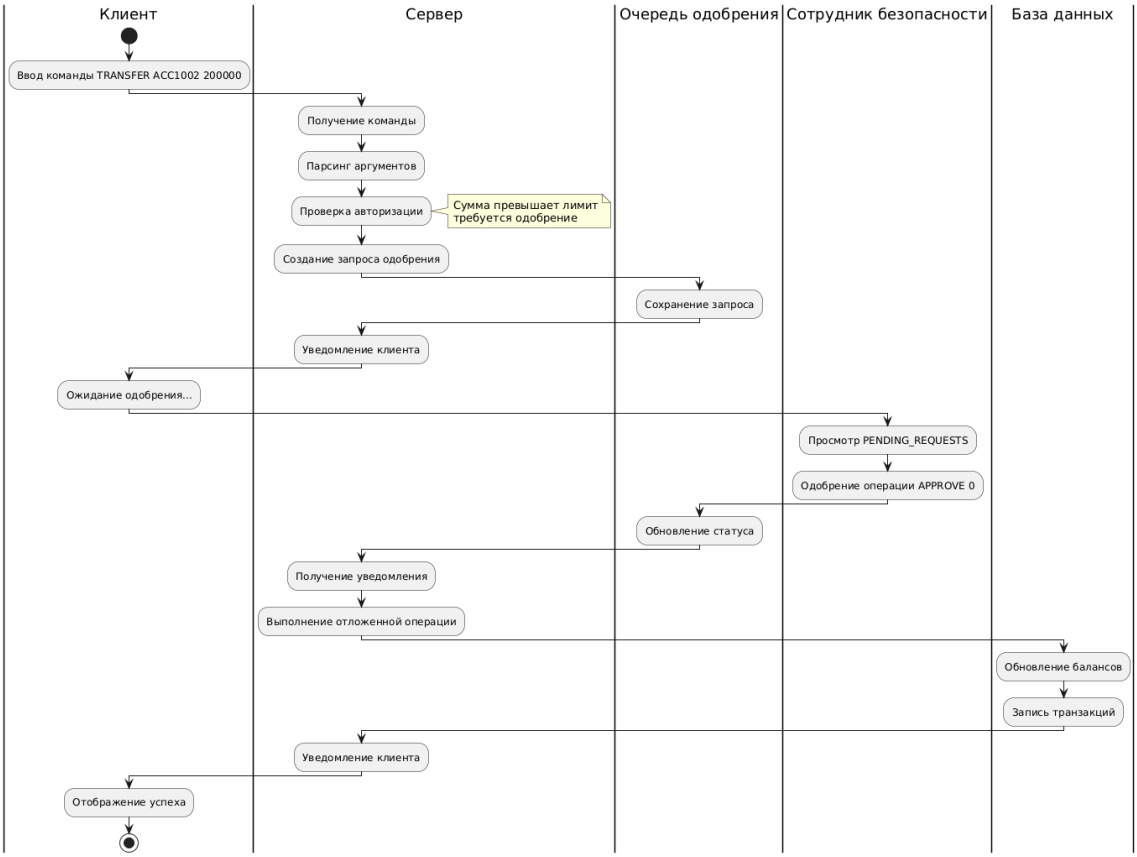

Рис. 7: Пайплайн операции перевода, требующей одобрения безопасности

5.3 Процесс регистрации нового клиента

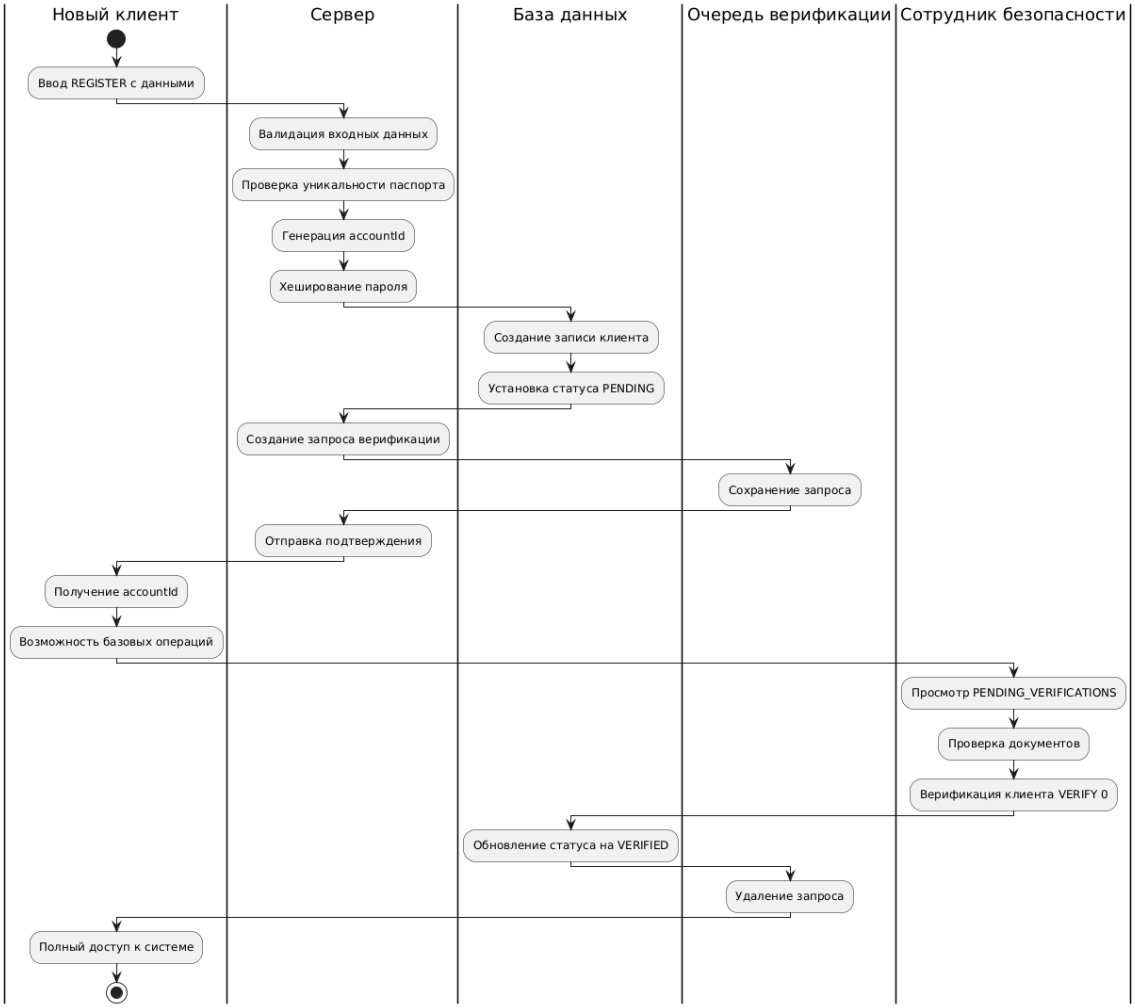

Рис. 8: Процесс регистрации и верификации нового клиента

6 Технические особенности

Шифрование данных

- Алгоритм: XOR + Base64 (для демонстрации, но можно и более продвинутые)
- Ключ: bank-system-key-2024 (для проверки дешифровальщика)
- Область применения: файлы базы данных и настроек

Сетевое взаимодействие

- Протокол: TCP/IP
- Порт по умолчанию: 8080 (но можно открывать на произвольном)
- Формат команд: текстовые строки
- Кодировка: UTF-8 (по умолчанию)

Управление памятью

- Умные указатели: для автоматического управления памятью
- RAII и умные указатели: для корректного распределения ресурсов (сокеты, файлы)
- Контейнеры STL: для хранения данных

Обработка ошибок

- Исключения: для корректной обработки ошибок
- Коды возврата: для прослеживания логики работы
- Логирование: в консоль сервера и системные файлы

7 Пайплайн работы над проектом

Изучение теоретической базы

- Принципы клиент-серверной архитектуры
- Сетевые протоколы TCP/IP, сокеты в C++
- Многопоточное программирование
- Основы криптографии (хеширование, шифрование XOR + Base64)
- Паттерны проектирования для банковских систем
- Системы аутентификации и авторизации
- Принципы безопасного хранения данных

Поэтапная разработка системы

Фаза 1: Базовая архитектура (Недели 1-2)

Создание "костяка" системы:

- Разработана базовая структура проекта с разделением на модули
- Реализованы классы Account, Transaction для представления банковских сущностей
- Создана заготовка для системы клиент-серверного взаимодействия

Основные решения:

- Выбрана трехуровневая архитектура (клиент-сервер-база данных)
- Определены основные типы счетов: SAVINGS, CHECKING, CREDIT, DEPOSIT
- Разработана структура транзакций с историей операций

Фаза 2: Сетевое взаимодействие (Недели 3-4)

Реализация клиент-серверной связи:

- Разработан класс BankServer с многопоточным подключением клиентов
- Создан класс BankClient для взаимодействия с сервером
- Реализован текстовый протокол для команд (в терминале)

Особенности реализации:

- Многопоточная обработка клиентов через std::thread
- Система сессий для управления состоянием клиентов

Фаза 3: Работа с данными и система аутентификации (Недели 5-7)

Реализация системы хранения:

- Класс Database для управления данными
- Шифрование записей в базе данных (создан модуль Crypto для шифрования и хеширования)
- Реализована система регистрации и логина
- Добавлена двухуровневая аутентификация (клиенты/суперпользователи)

Особенности реализации:

- Автоматическое сохранение при изменениях
- Резервное копирование и восстановление
- Формат хранения с поддержкой сохранения истории транзакций

Реализовано:

- REGISTER – регистрация новых пользователей
- LOGIN – вход для обычных клиентов
- SUPERLOGIN – вход для сотрудников безопасности
- Реализована утилита view_database для просмотра зашифрованных данных
- Создан скрипт инициализации тестовых данных init_database
- Добавлено логирование ключевых событий на сервере

Фаза 4: Базовые банковские операции (Неделя 8)

Реализация основных функций:

- DEPOSIT / DEPOSIT_TO – пополнение счетов
- WITHDRAW / WITHDRAW_FROM – снятие средств
- TRANSFER / TRANSFER_FROM – переводы между счетами
- CREATE_ACCOUNT – создание новых счетов

Особенности реализации:

- Автоматическое ведение истории транзакций
- Поддержка кредитных лимитов для счетов

Фаза 5: Система одобрения операций (Недели 9-10)

Реализация безопасности для крупных операций:

- Механизм очередей запросов на одобрение
- Команды для суперпользователей: APPROVE, REJECT, PENDING_REQUESTS
- Автоматическое определение крупных операций по пороговому значению

Особенности реализации:

- Операции выше порога требуют одобрения безопасности
- Система ожидания ответа с таймаутом
- Уведомления клиентов о статусе операций

Фаза 6: Верификация пользователей (Недели 9-10)

Система проверки клиентов:

- Статусы пользователей: PENDING_VERIFICATION, VERIFIED, BLOCKED (в процессе)
- Очередь запросов на верификацию
- Команды VERIFY, PENDING_VERIFICATIONS для суперпользователей

Для неверефицированных пользователей установлены:

- Лимиты на размер операций
- Запрет на создание кредитных и депозитных счетов

Для верифицированных пользователей остаётся доступен весь ранее описанный функционал.

Фаза 7: Процентные ставки и настройки (Неделя 11, не закончено)

Управление банковскими параметрами:

- Система хранения настроек банка (лимиты, ставки)
- Команды SET_RATES, SETTINGS для управления ставками
- Динамическое применение изменений

Фаза 8: Тестирование системы (Недели 12-13)

Модульное тестирование

- Создан комплексный тестовый класс BankSystemTest
- Тестирование основных сценариев: регистрация, логин, операции
- Проверка обработки ошибок и граничных случаев

Интеграционное тестирование

- Тестирование взаимодействия клиент-сервер
- Проверка работы системы одобрения операций
- Тестирование восстановления после сбоев

Нагрузочное тестирование

- Проверка многопользовательской работы
- Тестирование обработки параллельных запросов

Инструменты тестирования

- Google Test framework для unit-тестов
- Автоматизированные сценарии для интеграционного тестирования
- Ручное тестирование пользовательских сценариев

Документирование

- Создана документация на русском языке
- Описаны функциональные и системные требования
- Разработаны архитектурные диаграммы и схемы работы
- Создано пользовательское руководство с примерами

Оптимизация и рефакторинг

- Улучшена обработка ошибок
- Оптимизирована работа с памятью
- Улучшена производительность сетевого взаимодействия

Итог работы

Проект развивался по принципу “снизу вверх” – от базовых структур данных к сложной логике взаимодействия клиента, сервера и суперпользователя. Каждый этап включал проектирование, реализацию, а также обязательное тестирование и интеграцию (что облегчалось, так как старались придерживаться модульности приложения по SOLID). Такой подход позволил создать стабильную, безопасную и расширяемую банковскую систему.

8 Заключение

Secure Bank System представляет собой комплексное решение для безопасного банковского обслуживания с модульной архитектурой и многоуровневой системой безопасности. Система демонстрирует лучшие практики разработки финансовых приложений и может служить основой для более сложных банковских систем.

Список литературы

[1] ISO/IEC 14882:2017 - Programming Language C++

[2] Seacord, R. C. (2013). *Secure Coding in C and C++*

[3] Donahoo, M. J., & Calvert, K. L. (2001). *TCP/IP Sockets in C: Practical Guide for Programmers*

[4] RAII и умные указатели // Справочник по C++ Яндекс [Электронный ресурс]. URL: https://education.yandex.ru/handbook/cpp/article/raii-and-smart-pointers.

[5] StivenHacker. Base64 and XOR Encryption // DeepWiki [Электронный ресурс]. URL: https://deepwiki.com/stivenhacker/GhostStrike/3.1-base64-and-xor-encryption.