Безопасность системы

Обзор архитектуры безопасности

Система обеспечивает многоуровневую защиту через комбинацию Spring Security с ролевой моделью, JWT-аутентификацию, криптографическую защиту банковских карт и безопасное хранение паролей.

Spring Security и управление доступом

Конфигурация безопасности

- **Сессии**: Используется `SessionCreationPolicy.STATELESS` для REST API
- **Публичные эндпойнты**: Доступ без аутентификации к `/public/**`, `/api/auth/**`, `/swagger-ui/**`
- **Защищенные ресурсы**: Все остальные эндпойнты требуют аутентификации

Ролевая модель

- **Роли пользователей**:
  - `USER` - обычные пользователи (роль `ROLE_USER`)
  - `ADMIN` - администраторы (роль `ROLE_ADMIN`)
- **Префикс ролей**: Используется стандартный префикс `ROLE_` Spring Security
- **Контроль доступа**: Методы с аннотацией `@PreAuthorize("hasRole('ROLE_ADMIN')")` доступны только администраторам

Права доступа по ролям

- **Администраторы**:
  - Создание новых карт
  - Активация и блокировка карт
  - Удаление карт
  - Просмотр всех данных карт (маскированные номера)
  - Обработка заявок на блокировку
- **Пользователи**:
  - Просмотр собственных карт (маскированные номера)
  - Операции с балансом (пополнение/снятие)

- Денежные переводы
- Подача заявок на блокировку карт

JWT аутентификация

Параметры токенов

- **Алгоритм подписи**: HMAC с секретным ключом
- **Срок действия**: 60 минут по умолчанию (настраивается через `security.jwt.duration`)
- **Формат**: Стандартный JWT (header.payload.signature)

Claims в токене

- `sub` - идентификатор пользователя (UUID)
- `role` - роль пользователя (USER/ADMIN)
- `iat` - время создания токена
- `exp` - время истечения токена

Валидация токенов

- **Обработка ошибок**:
  - `ExpiredJwtException` - истекший токен
  - `MalformedJwtException` - некорректный формат
  - `SignatureException` - неверная подпись
  - `UnsupportedJwtException` - неподдерживаемый тип
- **Извлечение данных**: Автоматическое создание `Authentication` объекта в `SecurityContext`

Защита номеров банковских карт

Двойное шифрование (Envelope Encryption)

1. **Первый уровень**: Номер карты шифруется индивидуальным AES-128 ключом
2. **Второй уровень**: AES ключ шифруется мастер-ключом
3. **Хранение**: В базе данных сохраняется зашифрованный номер и зашифрованный ключ отдельно

Алгоритмы шифрования

- **Симметричное шифрование**: AES-128 для номеров карт

- **Генерация ключей**: `KeyGenerator` с инициализацией 128 бит
- **Кодирование**: Base64 для хранения зашифрованных данных

Поиск карт

- **HMAC-SHA256**: Используется для создания хешей номеров карт
- **Уникальность**: Проверка через таблицу `CardHash` без расшифровки
- **Ключ HMAC**: Отдельный ключ из конфигурации `security.card.number.hmac`

Маскирование номеров

- **Формат отображения**: `**** **** **** XXXX` (показываются только последние 4 цифры)
- **Доступ к полному номеру**: Только для администраторов в тестовых методах
- **Соответствие PCI DSS**: Requirement 3.3 по маскированию PAN

Управление ключами

Иерархия ключей

- **Мастер-ключ**: Хранится в переменной окружения `security.card.number.key`
- **DEK (Data Encryption Key)**: Генерируется для каждой карты индивидуально
- **KEK (Key Encryption Key)**: Мастер-ключ для шифрования DEK

Операции с ключами

- **Генерация**: Автоматическая генерация AES ключей для новых карт
- **Шифрование ключей**: Мастер-ключом с алгоритмом AES
- **Расшифровка**: При необходимости доступа к номеру карты
- **Удаление**: Каскадное удаление при удалении карты

Хранение паролей

BCrypt хеширование

- **Алгоритм**: BCrypt с солью для каждого пароля
- **Cost factor**: Используется значение по умолчанию (обычно 10)
- **Проверка**: Метод `matches()` для валидации введенного пароля
- **Рекомендации**: Для повышения безопасности рекомендуется увеличить cost factor до 12-14

Безопасность паролей

- **Уникальная соль**: Автоматически генерируется для каждого пароля
- **Односторонняя функция**: Невозможность восстановления исходного пароля
- **Защита от Rainbow Tables**: Благодаря использованию соли

Дополнительные меры безопасности

Валидация операций

- **Проверка статуса карты**: Только активные карты могут использоваться
- **Проверка срока действия**: Карты с истекшим сроком блокируются
- **Валидация баланса**: Проверка достаточности средств перед операциями

Обработка ошибок

- **Маскирование ошибок**: Общие сообщения "Internal Error" для скрытия деталей реализации
- **Логирование**: Детальная информация об ошибках в логах (не показывается пользователю)
- **Exception handling**: Централизованная обработка через `AuthenticationEntryPoint`

Генерация уникальных номеров

- **Алгоритм**: 16-значные номера через `ThreadLocalRandom`
- **Проверка уникальности**: До 100 попыток генерации с проверкой по HMAC
- **Хранение хешей**: Таблица `CardHash` для быстрой проверки существования