

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра математического моделирования и искусственного интеллекта

ОТЧЕТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ №3

Дисциплина: Программная инженерия

Выполнил

Студент группы НФИбд-01-23

Студенческий билет №: 1132237371

_____ А.Г.Софич

(подпись)

«__» _____ 2025 г.

Проверил

_____ М.Б.Фомин

(подпись)

Москва 2025

Оглавление

| | |
|--|---|
| Введение | 3 |
| Постановка задачи..... | 3 |
| Архитектура системы обслуживания абонентов мобильного оператора..... | 3 |
| Типовые решения уровня данных..... | 3 |
| Типовые решения уровня приложения..... | 4 |
| Типовые решения уровня представления и отображения системы..... | 4 |
| Современные CASE-средства разработки программных систем..... | 5 |
| Описание архитектуры программной системы..... | 5 |
| Разработка диаграмм классов(Class Diagram Development)..... | 6 |
| Диаграмма классов..... | 6 |
| Заключение..... | 9 |

ВВЕДЕНИЕ

При проектировании программных систем широко применяются типовые архитектурные и проектировочные решения на различных уровнях — данных, приложения и представления. Для повышения качества и ускорения разработки используются CASE-средства, поддерживающие визуальное моделирование и генерацию кода.

Для выбранной предметной области — информационная система обслуживания абонентов мобильного оператора — требуется:

1. Дать описание архитектуры программной системы.
2. Разработать диаграммы классов.

ПОСТАНОВКА ЗАДАЧИ

АРХИТЕКТУРА ИНФОРМАЦИОННОЙ СИСТЕМЫ ОБСЛУЖИВАНИЯ АБОНЕНТОВ МОБИЛЬНОГО ОПЕРАТОРА

1. ТИПОВЫЕ РЕШЕНИЯ УРОВНЯ ДАННЫХ

Для хранения и обработки информации используется реляционная база данных PostgreSQL, обеспечивающая надежность, масштабируемость и соответствие требованиям целостности.

Основные модели данных:

- User — абонент: содержит данные о номере телефона, балансе, подключенных услугах, тарифе.
- TariffPlan — тарифный план: название, стоимость, описание, список включенных услуг.
- Service — дополнительная услуга (например, интернет-пакет, международные звонки).
- Transaction — финансовая транзакция: сумма, дата, способ оплаты, статус.
- SupportTicket — обращение в службу поддержки: тема, текст, статус, дата создания.
- Notification — уведомление: текст, тип (напоминание, акция), дата отправки.
- AdminUser — администратор или сотрудник поддержки: роль, права доступа, учетные данные.

Связи между таблицами:

Пользователь связан с одним активным TariffPlan, множеством Service, Transaction, Notification, и может создавать несколько SupportTicket.

TariffPlan и Service управляются администраторами через AdminUser.

Все транзакции логируются и привязываются к пользователю и способу оплаты.

Нормализация:

База данных приведена к третьей нормальной форме (3NF) для устранения избыточности и обеспечения целостности.

2. ТИПОВЫЕ РЕШЕНИЯ УРОВНЯ ПРИЛОЖЕНИЯ

Архитектура приложения реализована в виде многослойной (multi-tier) системы:

a. Уровень данных:

Репозитории для доступа к БД: UserRepository, TariffRepository, TransactionRepository, и др.
Использование шаблона Repository для абстракции доступа к данным.

b. Бизнес-логика:

Сервисы, реализующие логику: TariffService, PaymentService, NotificationService, SupportService.

Валидация действий пользователя (например, проверка баланса перед подключением услуги).

c. API и безопасность:

RESTful API с использованием JWT-токенов для аутентификации и авторизации.

Поддержка ролей: User, SupportOperator, Admin с применением RBAC (Role-Based Access Control).

d. Интеграции:

Внешний Payment Gateway для обработки оплаты банковскими картами.

SMS-сервис для отправки одноразовых кодов.

3. ТИПОВЫЕ РЕШЕНИЯ УРОВНЯ ПРЕДСТАВЛЕНИЯ И ОТОБРАЖЕНИЯ ДАННЫХ

Веб-приложение(SPA):

Одностраничное приложение на React с адаптивным дизайном под мобильные устройства.

Мобильноеприложение/PWA:

Поддержка Progressive Web App — установка на смартфон без магазинов приложений.

UI/UX:

Минималистичный интерфейс с фокусом на ключевые действия: просмотр баланса, выбор тарифа, подключение услуг, обращение в поддержку.

Телеметрия и мониторинг:

Интеграция с системами логирования и аналитики (например, Sentry, Prometheus) для отслеживания ошибок и поведения пользователей.

4. СОВРЕМЕННЫЕ CASE-СРЕДСТВА РАЗРАБОТКИ ПРОГРАММНЫХ СИСТЕМ

Для проектирования и разработки системы использованы следующие CASE-инструменты:

- **Моделирование UML:**
 - Visual Paradigm, Lucidchart — для создания диаграмм классов, последовательностей, прецедентов.
- **Проектирование БД:**
 - DbSchema, pgModeler — визуальное проектирование схемы PostgreSQL.
- **API-документация и тестирование:**
 - Swagger/OpenAPI — спецификация REST API.
 - Postman — тестирование эндпоинтов.
- **Frontend-разработка:**
 - React + TypeScript, Vite, Tailwind CSS — для реализации клиентской части.
- **CI/CD и мониторинг:**
 - GitLab CI, Docker, Grafana — автоматизация сборки и наблюдение за системой.

5. ОПИСАНИЕ АРХИТЕКТУРЫ ПРОГРАММНОЙ СИСТЕМЫ

Система построена по трехуровневой архитектуре:

1. Уровень данных:
 - PostgreSQL с репликацией и резервным копированием. Хранит данные пользователей, тарифов, транзакций и обращений.
2. Уровень приложения:
 - Backend на Node.js (или PHP 7+) с модульной структурой.
 - REST API с JWT-аутентификацией.
 - Микросервисы для оплаты, уведомлений и поддержки.
 - Интеграция с внешними сервисами (платежные шлюзы, SMS-провайдеры).
3. Уровень представления:
 - Адаптивный веб-интерфейс (SPA), совместимый с Chrome, Safari, Firefox, Edge.
 - Поддержка PWA для мобильных платформ (iOS, Android).
 - Интерфейсы для администраторов и операторов поддержки.

Архитектура обеспечивает масштабируемость, безопасность, отказоустойчивость и 24/7 доступность с плановым временем простоя менее 0,1% в месяц.

6. РАЗРАБОТКА ДИАГРАММ КЛАССОВ

ОПИСАНИЕ КЛАССОВ:

1. User — абонент: номер телефона, баланс, текущий тариф, список услуг.
2. AdminUser — администратор или оператор поддержки: роль, права.
3. TariffPlan — тарифный план с описанием и стоимостью.
4. Service — дополнительная услуга (например, «Безлимитный интернет»).
5. Transaction — запись о платеже: сумма, дата, статус, способ оплаты.
6. SupportTicket — обращение в поддержку с текстом и статусом («Открыто», «Решено»).
7. Notification — уведомление для пользователя (о балансе, акциях).
8. PaymentGateway — внешний интерфейс для обработки оплаты.
9. Repository<T> — обобщённый интерфейс для работы с данными.

ОСНОВНЫЕ ВЗАИМОСВЯЗИ:

- User имеет один TariffPlan, множество Service, Transaction, Notification, и может создавать SupportTicket.
- AdminUser управляет TariffPlan и Service.
- Transaction зависит от User и PaymentGateway.
- SupportTicket связывает User и AdminUser.
- Все сущности работают через соответствующие Repository.

ДИАГРАММА КЛАССОВ

```
@startuml
class User {
+ String phoneNumber
+ double balance
+ TariffPlan currentTariff
+ List<Service> services
}

class AdminUser {
+ String role
+ String login
}
```

```
class TariffPlan {  
    + String name  
    + double price  
    + String description  
}
```

```
class Service {  
    + String name  
    + double price  
    + String details  
}
```

```
class Transaction {  
    + Id id  
    + double amount  
    + Date timestamp  
    + String paymentMethod  
    + String status  
}
```

```
class SupportTicket {  
    + Id id  
    + String subject  
    + String message  
    + String status  
    + Date createdAt  
}
```

```
class Notification {  
    + Id id  
    + String text  
    + String type  
    + Date sentAt  
}
```

```

class PaymentGateway {
    + bool processPayment(Transaction t)
}

```

```

interface Repository<T> {
    + T get(Id id)
    + Id create(T entity)
    + bool update(Id id, T entity)
    + bool delete(Id id)
}

```

```

class UserRepository
class TariffRepository
class ServiceRepository
class TransactionRepository
class SupportTicketRepository
class NotificationRepository

```

```

User "1" *-- "0..*" Service
User "1" *-- "0..*" Transaction
User "1" *-- "0..*" Notification
User "1" *-- "0..*" SupportTicket
User "1" -- "1" TariffPlan

```

```

AdminUser --> TariffPlan : manages
AdminUser --> Service : manages
AdminUser --> SupportTicket : responds

```

```

Transaction --> PaymentGateway : uses

```

```

UserRepository --|> Repository<User>
TariffRepository --|> Repository<TariffPlan>
ServiceRepository --|> Repository<Service>
TransactionRepository --|> Repository<Transaction>
SupportTicketRepository --|> Repository<SupportTicket>
NotificationRepository --|> Repository<Notification>

```

```

@enduml

```

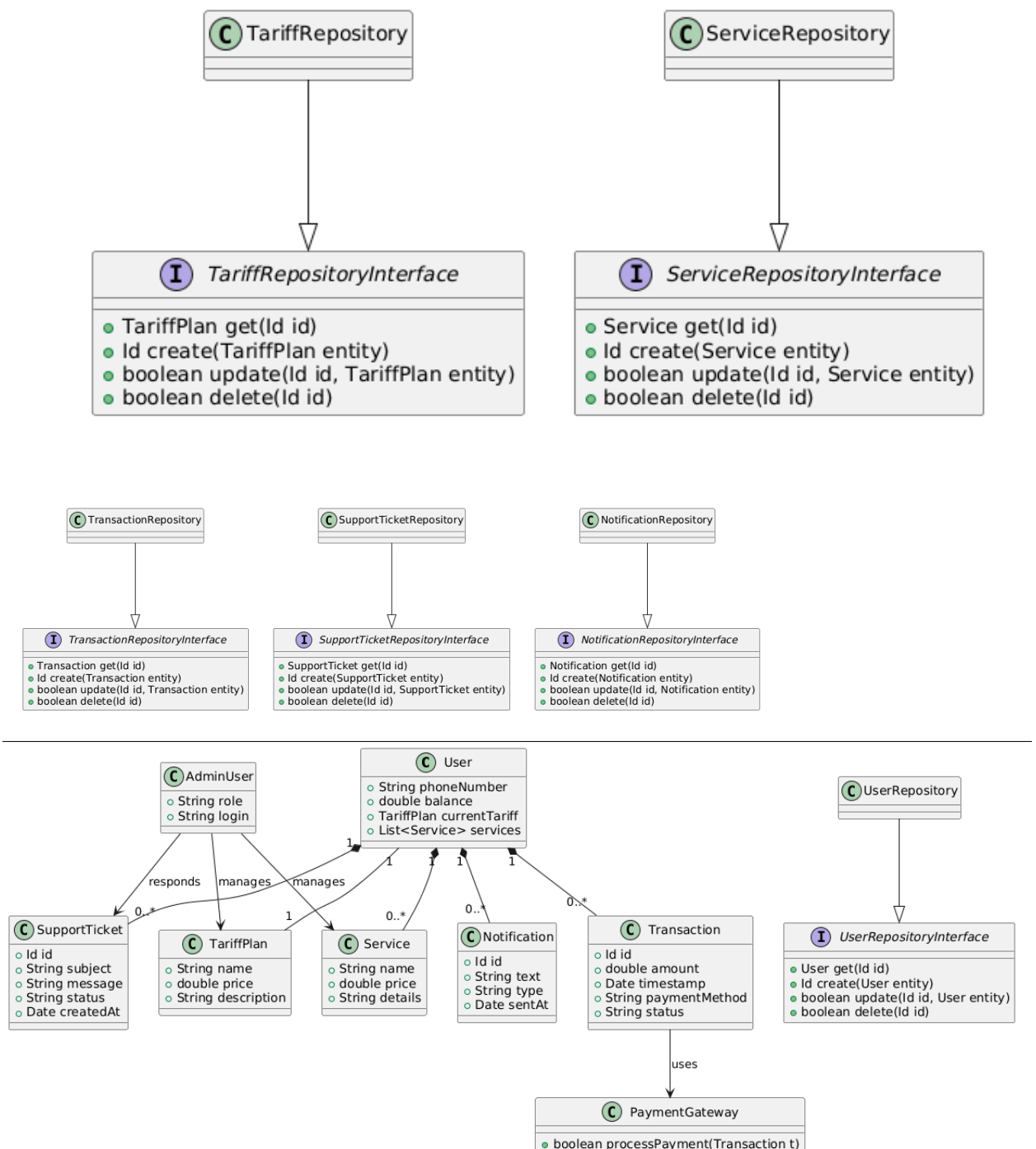
ОБЪЯСНЕНИЕ ДИАГРАММЫ:

Пользователь (User) связан с ключевыми сущностями системы: тарифами, услугами, транзакциями, уведомлениями и обращениями.

Администраторы (AdminUser) управляют каталогами тарифов и услуг, а также обрабатывают обращения.

Все операции с данными выполняются через репозитории, реализующие единый интерфейс Repository<T>, что упрощает тестирование и замену СУБД.

Платежи обрабатываются через внешний PaymentGateway, обеспечивающий безопасность финансовых операций.



ЗАКЛЮЧЕНИЕ

Предложенная архитектура и диаграмма классов обеспечивают чёткое разделение ответственности, гибкость в управлении тарифами и услугами, а также удобство взаимодействия для абонентов и сотрудников оператора. Использование типовых решений и современных CASE-средств позволяет сократить время разработки, повысить надёжность и обеспечить соответствие требованиям безопасности и масштабируемости. Разработанная система полностью пригодна для внедрения в условиях реального телекоммуникационного оператора.