

Вступление.....	1
Общая логика.....	2
Activity diagram.....	2
CDR.....	3
Описание логики.....	3
Entity Relationship Diagram.....	4
Use Case & User Story.....	7
BRT.....	9
Описание логики.....	9
Entity Relationship Diagram.....	13
Use Case & User Story.....	18
Sequence Diagram.....	23
HRS.....	26
Концепция сущностей и параметров.....	26
Описание логики.....	27
Entity Relationship Diagram.....	32
Use Case & User Story.....	41
Sequence Diagram.....	43
CRM.....	44
Описание логики.....	44
Entity Relationship Diagram.....	44
Use Case & User Story.....	45
Use Case Diagram.....	51
Sequence Diagram.....	51
Activity Diagram.....	55
Swagger.....	58

Вступление

Здравствуйте! Данный документ содержит все мои наработки по проекту nexign bootcamp 2025 по разработке биллинговой системы.

Для удобства все артефакты системного анализа были разнесены по соответствующим их содержанию сервисам и разделам. Для удобного поиска можете воспользоваться оглавлением.

Общая логика

Activity diagram

Общая Activity диаграмма, демонстрирующая весь процесс поступления звонков и их тарификации представлена на рисунке 1.

[Код для PlantUml](#)

[Онлайн версия в большем разрешении](#)

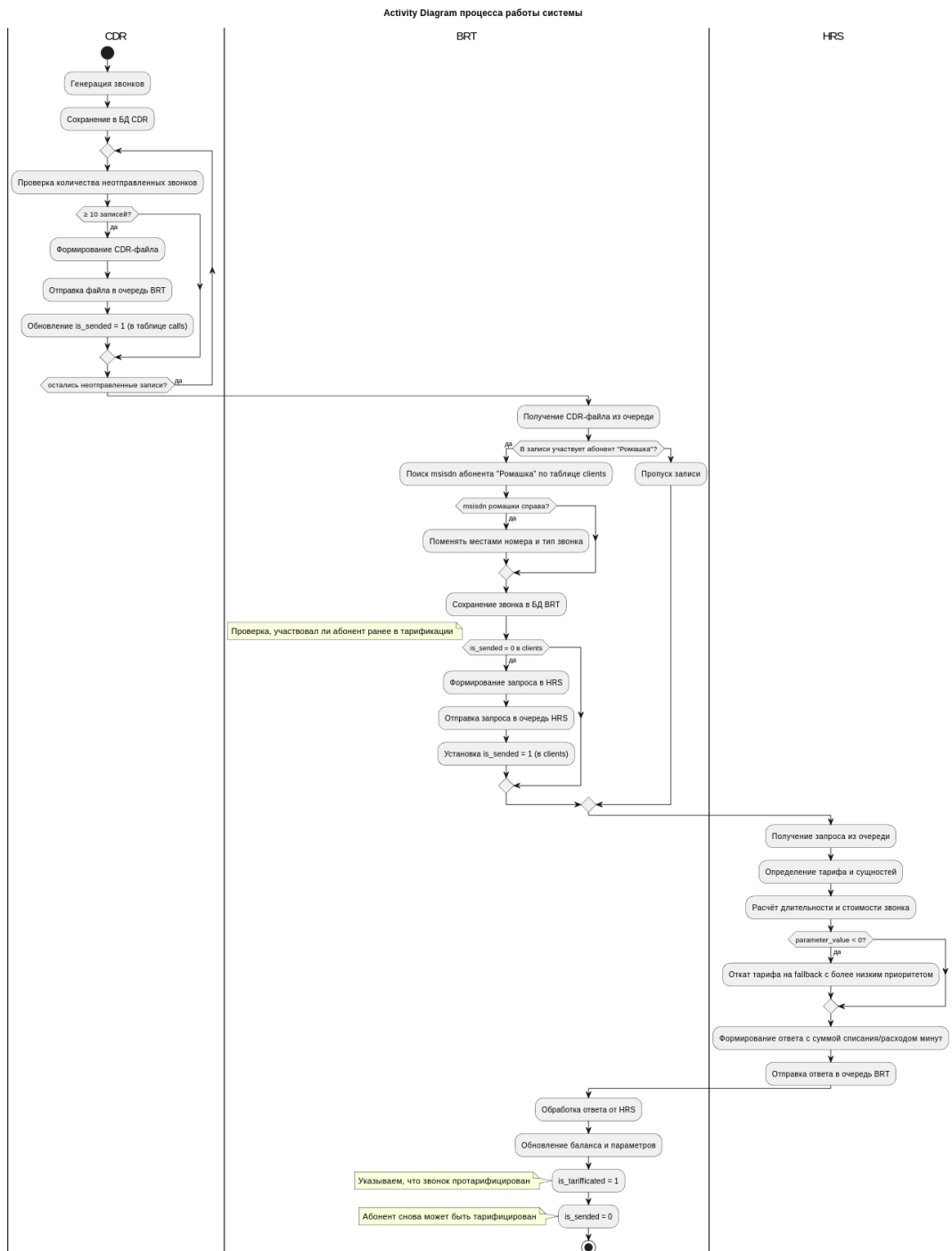


Рисунок 1 - Логика работы системы

CDR

Описание логики

Система формирует несколько потоков генерации записей.

Сам поток представляет из себя генератор, в котором из заготовленной БД пользователей (callers) берутся два случайных человека, далее случайно определяется входящий или исходящий вызов и случайное время начала и конца звонка. Записи формируются в пределах одного года, начиная с дня и месяца, равного сегодняшнему, но на год раньше.

На рисунке 2 представлена структура таблицы callers в БД CDR, которая содержит всех возможных людей для генерации звонков. Комментарии к каждому столбцу написаны справа в скобках. Общий вид БД со всеми таблицами и связями представлен в разделе [ERD](#) данного сервиса.

callers	Таблица всех звонивших людей			
🔑 caller_id	INT	N-N	UQ	(ИД)
msisdn	BIGINT	N-N	UQ	(Номер телефона звонящего)

Рисунок 2 - Таблица callers

Если во время генерации внутри потока возникает пересекающиеся звонки одного пользователя, то запись удаляется и генерируется новая. Если оказывается, что звонок начался до 24:00, а закончился после, то такой звонок занимает две записи.

Затем полученные звонки переносятся в таблицу calls из хранилища звонков каждого потока генерации.

На рисунке 3 представлена структура таблицы calls в БД CDR, в которую заносятся все отобранные звонки из всех потоков генерации.

calls	Таблица всех звонков			
🔑 call_id	INT	N-N	UQ	(ИД)
call_type	INT	N-N	UQ	(Тип звонка: входящий - 1 / исходящий - 2)
left_msisdn	BIGINT	N-N	UQ	(Номер телефона слева в записи)
right_msisdn	BIGINT	N-N	UQ	(Номер телефона справа в записи)
start_datetime	DATETIME	N-N	UQ	(Дата и время начала звонка)
end_datetime	DATETIME	N-N	UQ	(Дата и время начала звонка)
is_sended	BIT	N-N	UQ	(0 - Звонок не отправлен / 1 - Звонок отправлен)

Рисунок 3 - Таблица calls

Сервис проверяет звонки в БД на пересечение и удаляет их.

Далее система сортирует записи по дате и времени начала звонка от самой ранней до самой поздней.

Система формирует txt файл CDR по 10 штук из БД calls. Пример данных в txt файле CDR представлена на рисунке 4.

```
1,79996667755,79876543221,2025-02-10T10:12:25,2025-02-10T11:12:57
2,79001234567,79112223344,2025-02-10T11:03:15,2025-02-10T11:04:02
1,79881234567,79005556677,2025-02-10T12:45:00,2025-02-10T12:47:30
1,79997778899,79886665544,2025-02-10T13:20:10,2025-02-10T13:21:55
1,79110002233,79991112233,2025-02-10T14:00:00,2025-02-10T15:00:45
2,79880001122,79006667788,2025-02-10T14:30:25,2025-02-10T14:33:40
2,79009998877,79998887766,2025-02-10T15:10:10,2025-02-10T15:15:22
1,79881112233,79117778899,2025-02-10T16:05:00,2025-02-10T17:00:18
2,79990001122,79887776655,2025-02-10T17:31:00,2025-02-10T17:54:12
1,79119998877,79007778899,2025-02-10T17:40:45,2025-02-10T18:23:00
```

Рисунок 4 - Пример CDR файла

Сформированный файл асинхронно отправляется в BRT. Если запись отправлена из CDR в очередь в BRT, то она меняет значение “is_sended” на 1 в таблице calls БД CDR.

Система формирует новые CDR файлы по описанному принципу, пока все записи из БД calls не будут отправлены.

Entity Relationship Diagram

Общий вид ERD для данного сервиса представлен на рисунках 5 и 6. Часть информации по таблицам, которая нужна была для рассмотрения логики работы системы представлена в разделе [Описание логики](#).

[Онлайн версия в большем разрешении](#)

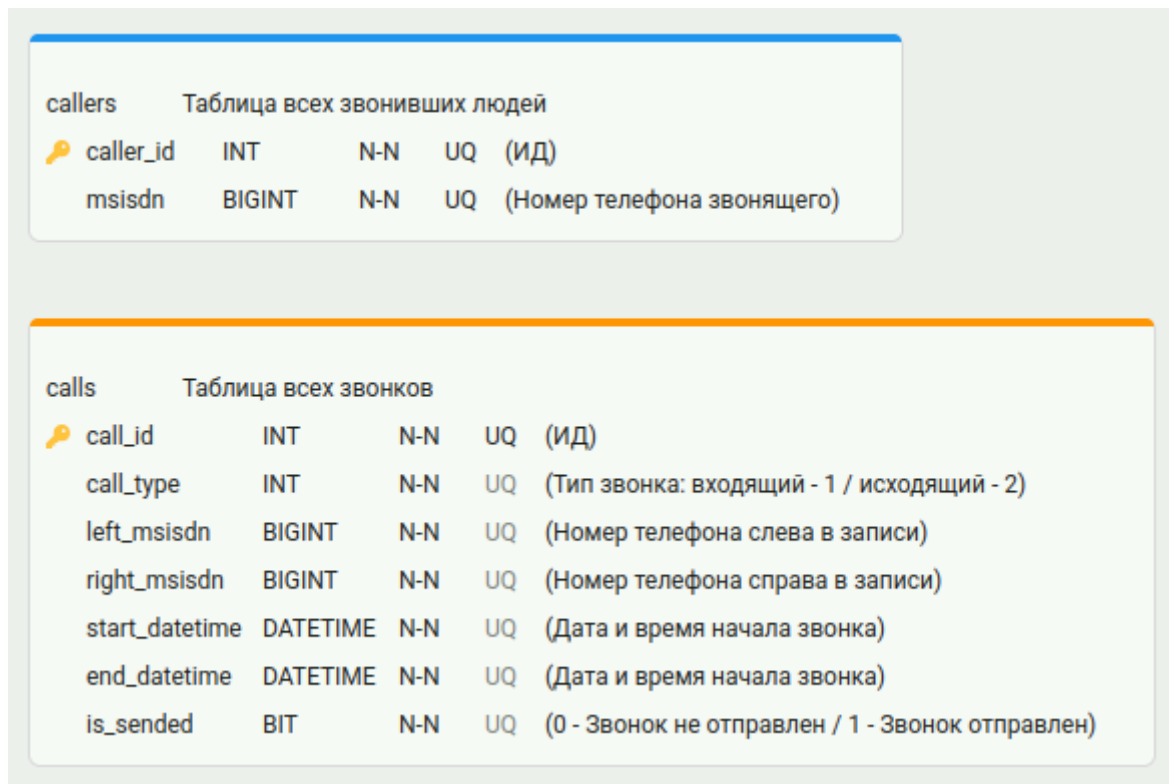


Рисунок 5 - ERD диаграмма сервиса CDR

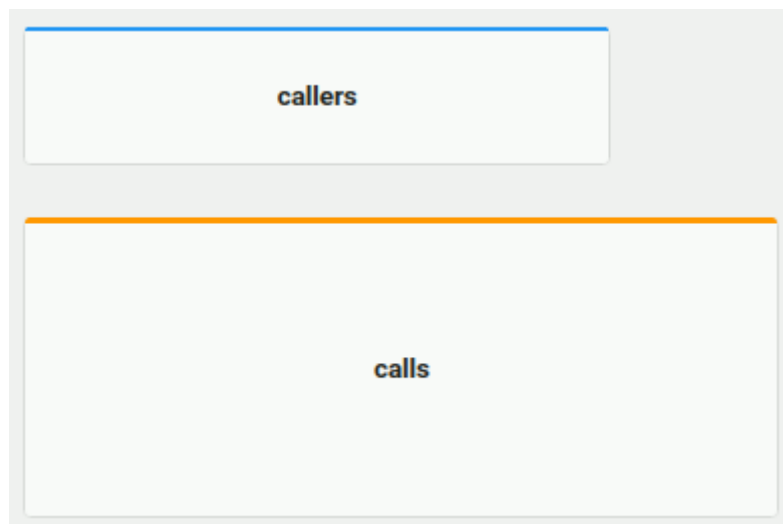


Рисунок 6 - Обобщённая ERD диаграмма сервиса CDR

Описание всех таблиц с изображениями и примерами:

- **Таблица callers** содержит всю информацию о людях, участвующих в генерации звонков.

Таблица представлена на рисунке 7.

Пример данных представлен в таблице 1

callers	Таблица всех звонивших людей			
🔑 caller_id	INT	N-N	UQ	(ИД)
msisdn	BIGINT	N-N	UQ	(Номер телефона звонящего)

Рисунок 7 - Таблица callers

Таблица 1 - Пример для callers

caller_id	2
msisdn	79379802333

- **Таблица calls** содержит информацию о сгенерированных звонках
Таблица представлена на рисунке 8.
Пример данных представлен в таблице 2

calls	Таблица всех звонков			
🔑 call_id	INT	N-N	UQ	(ИД)
call_type	INT	N-N	UQ	(Тип звонка: входящий - 1 / исходящий - 2)
left_msisdn	BIGINT	N-N	UQ	(Номер телефона слева в записи)
right_msisdn	BIGINT	N-N	UQ	(Номер телефона справа в записи)
start_datetime	DATETIME	N-N	UQ	(Дата и время начала звонка)
end_datetime	DATETIME	N-N	UQ	(Дата и время начала звонка)
is_sended	BIT	N-N	UQ	(0 - Звонок не отправлен / 1 - Звонок отправлен)

Рисунок 8 - Таблица calls

Таблица 2 - Пример для calls

call_id	15
call_type	1
left_msisdn	79379802644
right_msisdn	79279803453
start_datetime	2024-05-17T14:30:00
end_datetime	2024-05-17T14:45:00
is_sended	1

Use Case & User Story

В таблице 3 приведен перечень Use Case и User Story.

Таблица 3 - CDR UC & US

	Вариант использования	Пользовательская история
01	Генерация звонков	Как менеджер, я хочу сгенерировать записи звонков, чтобы в системе были входные данные для дальнейшей обработки
02	Формирование и отправка CDR	Как менеджер, я хочу сохранять и отправлять звонки в систему, чтобы в дальнейшем списать у абонентов плату за них

01 User Story

Как менеджер, я хочу сгенерировать записи звонков, чтобы в системе были входные данные для дальнейшей обработки

01 Use Case

Название: Генерация звонков

Триггер: Пользователь инициировал запуск системы

Предусловия:

1. Таблица callers БД CDR не пустая
2. Система находится в запущенном состоянии

Результат: В БД CDR(таблица calls) добавлена новая запись звонка.

Основной поток:

1. Система выбирает случайную пару абонентов
2. Система определяет направление вызова (входящий/исходящий)
3. Система случайно генерирует дату начала и окончания звонка.
4. Система проверяет, когда начался и закончился звонок
5. Система повторяет процесс создания звонков, пока не будет сгенерировано записей на 1 год
6. Система проверяет, пересекаются ли звонки
7. Система сохраняет записи в БД.

Альтернативные потоки:

4а Звонок начался до 24:00, а закончился после

4а.1 Система разделяет такой звонок на два звонка (один до 24:00, второй после)

4а.2 Система записывает получившиеся звонки в БД

4а.3 Возврат к пункту 5 основного потока

6а Звонки пересекаются

6a.1 Система удаляет звонок, содержащий конфликт

02 User Story

Как менеджер, я хочу сохранять и отправлять звонки в систему, чтобы в дальнейшем списать у абонентов плату за них

02 Use Case

Название: Формирование и отправка CDR

Триггер: В БД CDR(таблица calls) появляется 10-ая неотправленная запись

Предусловия:

1. Таблица calls БД CDR содержит 10 неотправленных записей
2. Система находится в запущенном состоянии

Основной поток:

1. Система выбирает 10 последних записей из таблицы calls в БД CDR
2. Система формирует файл в нужном формате
3. Система отправляет файл в BRT
4. Система обновляет статус этих записей в БД(таблица calls) CDR как отправленные
5. Система ожидает получения новых 10 неотправленных записей
6. Основной поток повторяется пока все записи в БД calls не будут помечены, как отправленные

BRT

Описание логики

Сервис BRT получает новый CDR файл, после чего записывает в таблицу calls звонки исключительно содержащие абонентов “ромашки” (данные абоненты содержатся в таблице clients).

На рисунке 9 представлена структура таблицы calls в БД BRT, в которую вносятся записи звонков из CDR. Комментарии к каждому столбцу написаны справа в скобках. Общий вид БД со всеми таблицами и связями представлен в разделе [ERD](#) данного сервиса.





calls	Таблица звонков наших абонентов			
 call_id	int	N-N	UQ	(ИД)
 call_type_id	int	N-N	UQ	(ИД типа звонка)
left_msisdn	bigint	N-N	UQ	(Номер телефона слева в записи)
 left_operator_id	int	N-N	UQ	(ИД оператора номера слева)
right_msisdn	bigint	N-N	UQ	(Номер телефона справа в записи)
 right_operator_id	int	N-N	UQ	(ИД оператора номера справа)
start_datetime	datetime	N-N	UQ	(Дата и время начала звонка)
end_datetime	datetime	N-N	UQ	(Дата и время окончания звонка)
is_tarifficated	bit	N-N	UQ	(0 - Звонок не тарифицирован / 1 - Звонок тарифицирован)

Рисунок 9 - Таблица calls

Записи из файла CDR вносятся таким образом, чтобы номер абонента “ромашки” оказался “слева”. Например, BRT во время читки очередного файла CDR получает запись:

- (01, 79876543221 (*msisdn чужого оператора*), 79123456789 (*msisdn ромашки*), 2025-02-10T14:56:12, 2025-02-10T14:58:20)

В таком случае наша полученная запись преобразуется в следующую:

- (02, 79123456789 (*msisdn ромашки*), 79876543221 (*msisdn чужого оператора*), 2025-02-10T14:56:12, 2025-02-10T14:58:20)

То есть, при записи звонков наших абонентов в БД мы записываем номер нашего абонента первым для дальнейшей удобной передачи в HRS.

Если получается так, что звонок был между двумя абонентами “ромашки”, то в БД calls будет создана зеркальная копия данного звонка. Таким образом формируются две записи:

1. (01, 79876543323 (*msisdn ромашка1*), 79123456444 (*msisdn ромашка2*), 2025-02-10T14:56:12, 2025-02-10T16:24:13)
2. (02, 79123456444 (*msisdn ромашка2*), 79876543323 (*msisdn ромашка1*), 2025-02-10T14:56:12, 2025-02-10T16:24:13).

На рисунке 10 представлена структура таблицы clients в БД BRT, в которой содержатся наши абоненты.

clients Таблица наших абонентов				
 client_id	int	N-N	UQ	(ИД)
msisdn	bigint	N-N	UQ	(Номер телефона абонента)
last_name	varchar(30)	N-N	UQ	(Фамилия абонента)
first_name	varchar(30)	N-N	UQ	(Имя абонента)
middle_name	varchar(30)	NULL	UQ	(Отчество абонента)
 operator_id	int	N-N	UQ	(ИД оператора абонента)
registration_date	date	N-N	UQ	(Дата регистрации)
head_entity_instance_id	int	NULL	UQ	(ИД сущности, которая является изначальным тарифом для данного абонента)
balance	decimal(10,1)	N-N	UQ	(Баланс абонента)
balance_currency_id	int	N-N	UQ	(ИД валюты баланса)
region_id	int	N-N	UQ	(ИД региона абонента)
is_sended	int	N-N	UQ	(0 - Абонент не отправлен для тарификации / 1 - Абонент отправлен для тарификации)
description	varchar(50)	NULL	UQ	(Информация об абоненте)

Рисунок 10 - Таблица clients

Как только в БД появляется новая запись, то происходит проверка в таблице clients, чему равно поле “is_sended” у нашего абонента. Если оно равно 1, то звонок с данным абонентом уже отправлен в очередь в HRS для тарификации и система пропускает данную запись и идёт к следующей. Если же у абонента “is_sended” равно 0, то данный звонок отправляется в очередь в сервис HRS для тарификации и поле “is_sended” автоматически становится равно 1.

Данное решение существует, чтобы не оказалось, что при совершении ряда звонков одним нашим абонентом, получалась ситуация когда в очереди друг за другом стоят его звонки, а остаток минут во всех запросах одинаков, как и приоритет, как и прочие значения параметров. Из-за этого возникнет ситуация, что “помесячный” тариф сбросился до классического, но так как в очереди всё ещё устаревшая информация, то тарификация будет по не актуальному тарифу.

На рисунке 11 представлен пример JSON файла для асинхронной отправки в RabbitMQ в сервис HRS, на котором можно понять структуру тела запроса.

```
{
  "call_id": 23,
  "call_type": 1,
  "left_msisdn": 79261234567,
  "left_operator_name": "Ромашка",
  "right_msisdn": 79037654321,
  "right_operator_name": "Другой оператор",
  "start_datetime": "2025-05-17T14:30:00",
  "end_datetime": "2025-05-17T14:45:00",
  "entity_instance_id": [3, 4],
  "priority": [1, 1],
  "parameter_instance_id": [1],
  "parameter_value": [50],
  "period_payment_date": ["2025-05-01", null]
}
```

Рисунок 11 - Пример запроса BRT -> HRS

Поле `entity_instance_id` содержит ИД сущностей тарифа абонента, будь то головная сущность - тариф, а также пакет звонков и прочего (Для понимания, что такое сущность в моей реализации тарифов предлагаю ознакомиться с ними подробнее в разделе [Концепция сущностей и параметров](#))

Поле `priority` содержит приоритет данных сущностей абонента. Эти поля нужны, чтобы мы могли откатиться на другой тариф (или отдельные пакеты) при истечении лимита услуг

Поле `parameter_instance_id` содержит ИД параметров тарифа данного пользователя. Это нужно, чтобы можно было хранить остаток минут или прочего ресурса для каждого параметра.

Поле `parameter_value` содержит количество ресурса данного параметра (в нашем случае это минуты, но также это могут быть ГБ, смс и тд)

Поле `period_payment_date` содержит даты последней оплаты сущностей. В данном случае оплата есть только у сущности тарифа помесичный, поэтому только у данного тарифа будет дата последней оплаты, а у других сущностей (в том числе тарифа “классика” и пакетов звонков) будет стоять значение `null`

В HRS происходит тарификация и данные поступают обратно в BRT. Пример асинхронного запроса на изменение данных абонента из HRS в BRT представлен на рисунке 12.

```

{
  "call_id": 23,
  "msisdn": 79261234567,
  "entity_instance_id": [3, 4],
  "priority": [1, 1],
  "parameter_instance_id": [1],
  "parameter_value": [35],
  "period_payment_date": ["2025-05-01", null],
  "balance_change": 0
}

```

Рисунок 12 - Пример запроса HRS -> BRT

Полученный ответ обрабатывается и в таблицах clients, client_entities и entity_parameters происходят изменения соответствующих полей. Столбец “is_sended” данного абонента становится равен 0. Также в таблице calls у данного call_id в столбце “is_tarrificated” ставится 1.

На рисунках 13 и 14 представлена структура таблиц client_entities и entity_parameters соответственно. Данные таблицы нужны для хранения актуальной информации по тарифу клиента, которая может меняться (тариф “помесячный” может откатиться до тарифа “классика”, а количество оставшихся минут для параметра входящих и исходящих звонков уменьшается после звонка).

client_entities Таблица сущностей клиентов



 client_entity_id	int	N-N	UQ	(ИД)
 client_id	int	N-N	UQ	(ИД абонента)
entity_instance_id	int	N-N	UQ	(ИД сущности)
entity_type_id	int	N-N	UQ	(ИД типа сущности: тариф / пакет звонков / пакет минут и т.д.)
priority	int	N-N	UQ	(Приоритет сущности: После исчерпания лимита ресурса сущность заменяется на другую с более низким приоритетом)
entity_activation_date	date	N-N	UQ	(Дата активации сущности)
period_payment_date	date	NULL	UQ	(Дата оплаты сущности)

Рисунок 13 - Таблица client_entities

entity_parameters Таблица параметров сущностей



 entity_parameter_id	int	N-N	UQ	(ИД)
 client_entity_id	int	N-N	UQ	(ИД сущности)
parameter_instance_id	int	N-N	UQ	(ИД параметра)
parameter_value	int	N-N	UQ	(Количество оставшихся ресурсо...

Рисунок 14 - Таблица entity_parameters

Процесс повторяется для следующей записи в файле CDR, пока они не закончатся. Когда все записи в CDR файле обработаны, то он удаляется и из очереди поступает новый.

Entity Relationship Diagram

Общий вид ERD для данного сервиса представлен на рисунках 15 и 16. Часть информации по таблицам, которая нужна была для рассмотрения логики работы системы представлена в разделе [Описание логики](#).

[Онлайн версия в большем разрешении](#)

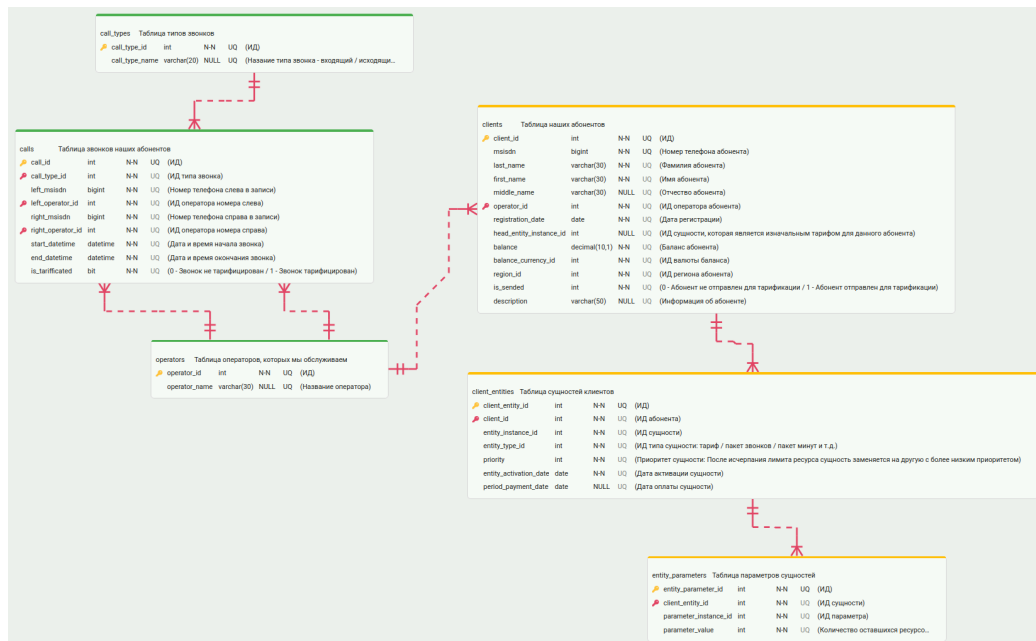


Рисунок 15 - ERD диаграмма сервиса BRT

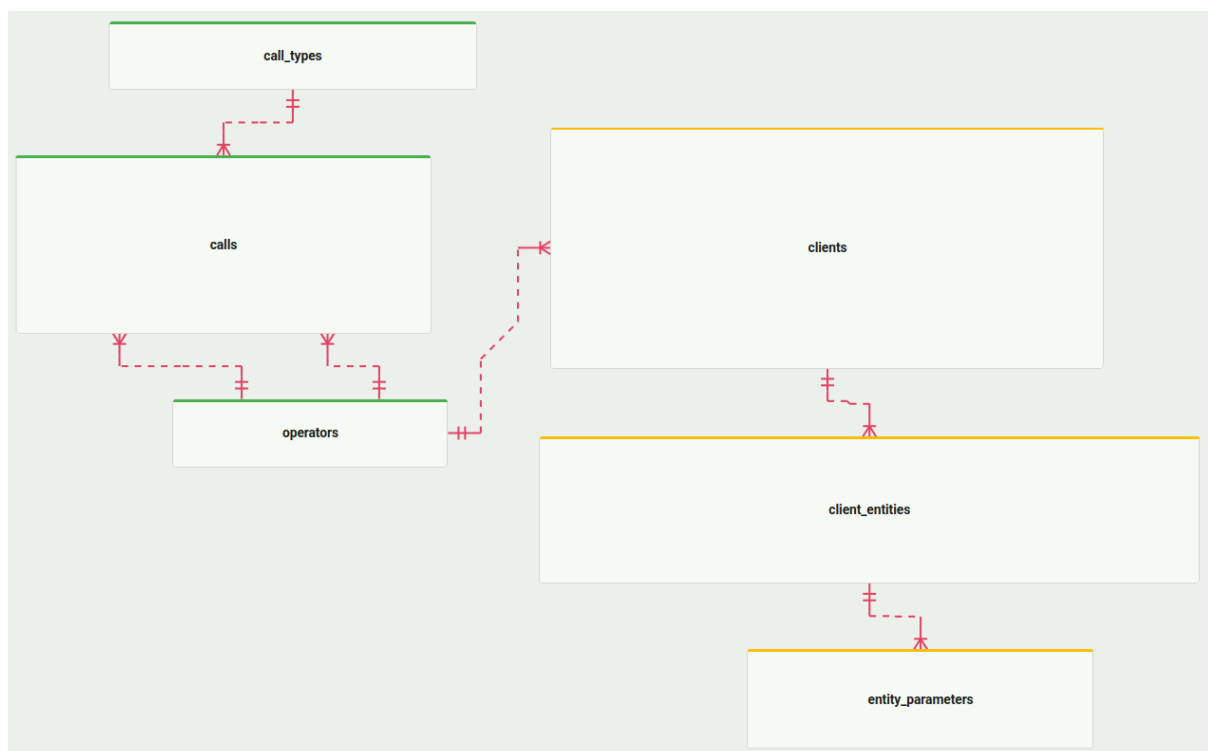


Рисунок 16 - Обобщённая ERD диаграмма сервиса BRT

Описание всех таблиц с изображениями и примерами:

- **Таблица calls** содержит всю информацию о звонках абонентов
Таблица представлена на рисунке 17.
Пример данных представлен в таблице 4.

calls Таблица звонков наших абонентов				
🔑 call_id	int	N-N	UQ	(ИД)
🔑 call_type_id	int	N-N	UQ	(ИД типа звонка)
left_msisdn	bigint	N-N	UQ	(Номер телефона слева в записи)
🔑 left_operator_id	int	N-N	UQ	(ИД оператора номера слева)
right_msisdn	bigint	N-N	UQ	(Номер телефона справа в записи)
🔑 right_operator_id	int	N-N	UQ	(ИД оператора номера справа)
start_datetime	datetime	N-N	UQ	(Дата и время начала звонка)
end_datetime	datetime	N-N	UQ	(Дата и время окончания звонка)
is_tarifficated	bit	N-N	UQ	(0 - Звонок не тарифицирован / 1 - Звонок тарифицирован)

Рисунок 17 - Таблица calls

Таблица 4 - Пример для calls

call_id	54
call_type_id	1
left_msisdn	79379802654
left_operator_id	2
right_msisdn	79279803456
right_operator_id	1
start_datetime	2024-05-17T14:30:00
end_datetime	2024-05-17T14:45:00
is_tarifficated	0

- **Таблица call_types** содержит тип вызова. Пользователи могут или инициировать или получать звонок.
Таблица представлена на рисунке 18.
Пример данных представлен в таблице 5.


call_types	Таблица типов звонков				
 call_type_id	int	N-N	UQ	(ИД)	
call_type_name	varchar(20)	NULL	UQ	(Название типа звонка - входящий / исходящий...)	

Рисунок 18 - Таблица call_types

Таблица 5 - Пример call_types

call_type_id	1
call_type_name	Исходящий

- **Таблица clients** содержит всю необходимую информацию про абонентов, которых мы обслуживаем

Таблица представлена на рисунке 19.

Пример данных представлен в таблице 6.



clients	Таблица наших абонентов				
 client_id	int	N-N	UQ	(ИД)	
msisdn	bigint	N-N	UQ	(Номер телефона абонента)	
last_name	varchar(30)	N-N	UQ	(Фамилия абонента)	
first_name	varchar(30)	N-N	UQ	(Имя абонента)	
middle_name	varchar(30)	NULL	UQ	(Отчество абонента)	
 operator_id	int	N-N	UQ	(ИД оператора абонента)	
registration_date	date	N-N	UQ	(Дата регистрации)	
head_entity_instance_id	int	NULL	UQ	(ИД сущности, которая является изначальным тарифом для данного абонента)	
balance	decimal(10,1)	N-N	UQ	(Баланс абонента)	
balance_currency_id	int	N-N	UQ	(ИД валюты баланса)	
region_id	int	N-N	UQ	(ИД региона абонента)	
is_sended	int	N-N	UQ	(0 - Абонент не отправлен для тарификации / 1 - Абонент отправлен для тарификации)	
description	varchar(50)	NULL	UQ	(Информация об абоненте)	

Рисунок 19 - Таблица clients

Таблица 6 - Пример clients

client_id	33
msisdn	79379802233
last_name	Костоглолов
first_name	Олег
middle_name	Павлович

operator_id	2
registration_date	2024-11-22
head_entity_instance_id	1
balance	100.0
balance_currency_id	1
region_id	1
is_sended	0
description	День рождения: 02.02.2001

- **Таблица operators** содержит операторов, которых мы обслуживаем. В данной работе оператор всего два - “Ромашка” и “Прочие”, но на перспективу была добавлена возможность добавлять новых. Структура представлена на рисунке 20. Пример данных представлен в таблице 7.

```

operators  Таблица операторов, которых мы обслуживаем
┌──────────┴──────────┐
│  operator_id  int      N-N  UQ  (ИД) │
│  operator_name varchar(30) NULL UQ  (Название оператора) │
└──────────┴──────────┘

```

Рисунок 20 - Таблица operators

Таблица 7 - Пример operators

operator_id	2
operator_name	Ромашка

- **Таблица client_entities** содержит сущности, которые относятся к тарифам клиентов. Подробнее про сущности и что это такое можно почитать в разделе [Концепция сущностей и параметров](#). Примеры сущностей: тариф “классика”, пакет звонков, пакет интернета и т.д. Таблица представлена на рисунке 21. Пример данных представлен в таблице 8.



client_entities Таблица сущностей клиентов				
 client_entity_id	int	N-N	UQ	(ИД)
 client_id	int	N-N	UQ	(ИД абонента)
entity_instance_id	int	N-N	UQ	(ИД сущности)
entity_type_id	int	N-N	UQ	(ИД типа сущности: тариф / пакет звонков / пакет минут и т.д.)
priority	int	N-N	UQ	(Приоритет сущности: После исчерпания лимита ресурса сущность заменяется на другую с более низким приоритетом)
entity_activation_date	date	N-N	UQ	(Дата активации сущности)
period_payment_date	date	NULL	UQ	(Дата оплаты сущности)

Рисунок 21 - Таблица client_entities

Таблица 8 - Пример client_entities

client_entity_id	2
client_id	48
entity_instance_id	1
entity_type_id	2
priority	1
entity_activation_date	2024-06-04
period_payment_date	2024-11-22

- Таблица entity_parameters** содержит параметры сущностей абонента с количеством ресурсов (ресурсом могут быть минуты, Гб интернета, смс и прочее)
 Примеры параметров: исходящие и входящие звонки, исходящие звонки на ромашку, исходящие звонки на другого оператора
 Таблица представлена на рисунке 22.
 Пример данных представлен в таблице 9.


entity_parameters Таблица параметров сущностей				
 entity_parameter_id	int	N-N	UQ	(ИД)
 client_entity_id	int	N-N	UQ	(ИД сущности)
parameter_instance_id	int	N-N	UQ	(ИД параметра)
parameter_value	int	N-N	UQ	(Количество оставшихся ресурсо...

Рисунок 22 - Таблица entity_parameters

Таблица 9 - Пример entity_parameters

entity_parameter_id	2
---------------------	---

client_entity_id	1
parameter_instance_id	1
parameter_value	50

Use Case & User Story

В таблице 1 приведен перечень Use Case и User Story.

Таблица 10 - BRT UC & US

	Вариант использования	Пользовательская история
01	Обработка CDR	Как менеджер, я хочу сохранять звонки абонентов в системе, чтобы иметь рассчитать, сколько абонент потратил средств на звонок
02	Отправка данных в HRS	Как менеджер, я хочу обрабатывать звонки абонентов в системе, чтобы понимать, сколько средств списать с баланса клиента за звонок
03	Оплата звонка	Как менеджер, я хочу сохранять звонки из CDR с привязкой к абонентам, чтобы передавать рассчитать, сколько абонент потратил средств на звонок
04	Смена тарифа	Как менеджер, я хочу менять тариф пользователям, чтобы удовлетворить их потребность
05	Списание за тариф	Как менеджер, я хочу списывать деньги с абонентов за пользование тарифом, чтобы получать прибыль от услуг
06	Пополнение	Как менеджер, я хочу дать возможность менеджерам пополнять баланс абонентов, чтобы они могли пользоваться услугами оператора
07	Запрос информации об абоненте	Как менеджер, я хочу дать возможность клиенту пополнить баланс, чтобы он мог пользоваться услугами оператора

08	Запрос на создание абонента	Как менеджер, я хочу дать возможность менеджеру создать нового абонента, чтобы он мог сохранять клиентов
----	-----------------------------	--

01 User Story

Как менеджер, я хочу сохранять звонки абонентов в системе, чтобы иметь рассчитать, сколько абонент потратил средств на звонок

01 Use Case

Название: Обработка CDR

Триггер: Получен новый файл от коммутатора

Предусловия:

1. Таблица clients БД BRT не пустая
2. Система находится в запущенном состоянии

Результат: В таблицу calls БД BRT добавлена новая запись звонка между двумя людьми.

Основной поток:

1. Система считывает записи из полученного файла
2. Система выбирает из полученного файла записи, относящиеся к абонентам “ромашки” из таблицы clients таблицы BRT
3. Система записывает выбранные записи в таблицу calls БД BRT
4. Система заканчивает обработку файла и удаляет его

Альтернативные потоки:

- 2а. В полученном файле ни один абонент не относится к оператору “ромашка”
 - 2а.1. Таблица calls БД BRT остаётся без изменений
 - 2а.2 Возврат к пункту 4 основного потока

02 User Story

Как менеджер, я хочу обрабатывать звонки абонентов в системе, чтобы понимать, сколько средств списать с баланса клиента за звонок

02 Use Case

Название: Отправка данных в HRS

Триггер: В таблице calls БД BRT появился новый звонок от абонента

Предусловия:

1. Таблица calls БД BRT не пустая
2. Таблица calls содержит не тарифицированный звонок
3. В таблице clients БД BRT есть абонент, баланс которого должен быть изменен
4. Система находится в запущенном состоянии
5. В сервис HRS поступил запрос на тарификацию звонка абонента

Результат: В HRS отправлен запрос на тарификацию звонка абонента

Основной поток:

1. Система считывает полученный нетарифицированный звонок из таблицы calls БД BRT
2. Система считает продолжительность звонка по времени начала и конца
3. Система отправляет запрос в HRS на тарификацию пользователя по продолжительности звонка

03 User Story

Как менеджер, я хочу сохранять звонки из CDR с привязкой к абонентам, чтобы передавать рассчитать, сколько абонент потратил средств на звонок

03 Use Case

Название: Оплата звонка

Триггер: Получен ответ от сервиса HRS

Предусловия:

6. Таблица clients БД BRT не пустая
7. В таблице clients БД BRT есть абонент, баланс которого должен быть изменен
8. Система находится в запущенном состоянии
9. В сервис HRS поступил запрос на тарификацию звонка абонента

Результат: Баланс абонента изменён

Основной поток:

1. Система обрабатывает полученный ответ от HRS
2. Система находит нужного абонента в таблице clients БД BRT
3. Система определяет тариф и регион абонента
4. Система считывает баланс абонента из таблицы
5. Система отнимает от баланса абонента плату (деньги или минуты в помесечном тарифе) за звонок
6. Система сохраняет данные абонента в таблице

Альтернативные потоки:

- 5а. У абонента кончились минуты на помесечном тарифе
 - 5а.1. Тарификация происходит по классическому тарифу
 - 5а.2 Система отнимает средства от баланса абонента
 - 5а.3 Возврат к пункту 6 основного потока

04. User Story:

Как менеджер, я хочу менять тариф пользователям, чтобы удовлетворить их потребность

04. Use Case:

Название: Запрос на смену тарифа

Триггер: Поступил запрос из CRM

Предусловия:

- Абонент существует в системе
- Система активна
- Запрос содержит id клиента, id тарифа и id подключенных сервисов

Результат: Абоненту присвоен новый тариф

Основной поток:

1. Система определяет по запросу уникальный номер абонента
2. Система находит абонента в базе данных clients
3. Система меняет id тарифа абонента на нужный
4. Система удаляет из таблицы баланса сервисов сервисы старого тарифа
5. Система вставляет в таблицу сервисов соответствующие новому тарифу сервисы
6. Система сохраняет изменения в БД
7. Система отправляет ответ в HRS

05. User Story:

Как менеджер, я хочу списывать деньги с абонентов за пользование тарифом, чтобы получать прибыль от услуг

05. Use Case:

Название: Списание за тариф

Триггер: Наступила дата списания платы за тариф

Предусловия:

- Система активна
- Абонент существует в системе
- Прошёл период оплаты, указанный в тарифе

Результат: У абонента списана плата за тариф

Основной поток:

1. Система определяет уникальный номер абонента, который должен оплатить тариф
2. Система находит абонента в базе данных clients
3. Система списывает с баланса абонента средства за тариф
4. Система сохраняет изменения в БД

06. User Story:

Как менеджер, я хочу дать возможность менеджерам пополнять баланс абонентов, чтобы они могли пользоваться услугами оператора

06. Use Case:

Название: Запрос на пополнение баланса

Триггер: Из HRS пришёл запрос на пополнение баланса

Предусловия:

- Абонент существует в системе
- Система активна
- Запрос содержит id клиента и сумму пополнения

Результат: Баланс абонента пополнен

Основной поток:

1. Система определяет по запросу уникальный номер абонента
2. Система находит абонента в базе данных clients
3. Система меняет баланс абонента на нужный
4. Система сохраняет изменения в БД

5. Система отправляет ответ в HRS

07. User Story:

Как менеджер, я хочу дать возможность клиенту пополнить баланс, чтобы он мог пользоваться услугами оператора

07. Use Case:

Название: Пополнение баланса

Триггер: Из HRS пришёл запрос на пополнение баланса

Предусловия:

- Абонент существует в системе
- Система активна
- Запрос содержит id клиента и сумму пополнения

Результат: Баланс абонента пополнен

Основной поток:

1. Система определяет по запросу уникальный номер абонента
2. Система находит абонента в базе данных clients
3. Система пополняет баланс абонента на указанную сумму
4. Система сохраняет изменения в БД
5. Система отправляет ответ в HRS

08. User Story:

Как менеджер, я хочу дать возможность менеджеру создать нового абонента, чтобы он мог сохранять клиентов

08. Use Case:

Название: Запрос на создание абонента

Триггер: Из HRS пришёл запрос на создание абонента

Предусловия:

- Система активна
- Запрос содержит информацию клиента, id тарифа и сервисов тарифа

Результат: Создан новый абонент

Основной поток:

1. Система проверяет, существует ли данный абонент в БД
2. Система добавляет в БД абонента с информацией, указанной в запросе
3. Система сохраняет изменения в БД
4. Система отправляет ответ в HRS

Альтернативные потоки:

- 1а. Абонент уже существует
 - 1а.1. Сервер возвращает ответ в HRS о том, что абонент уже существует

Sequence Diagram

Результат построения sequence diagram представлен на рисунках 23 - 27.

[Код для PlantUml](#)

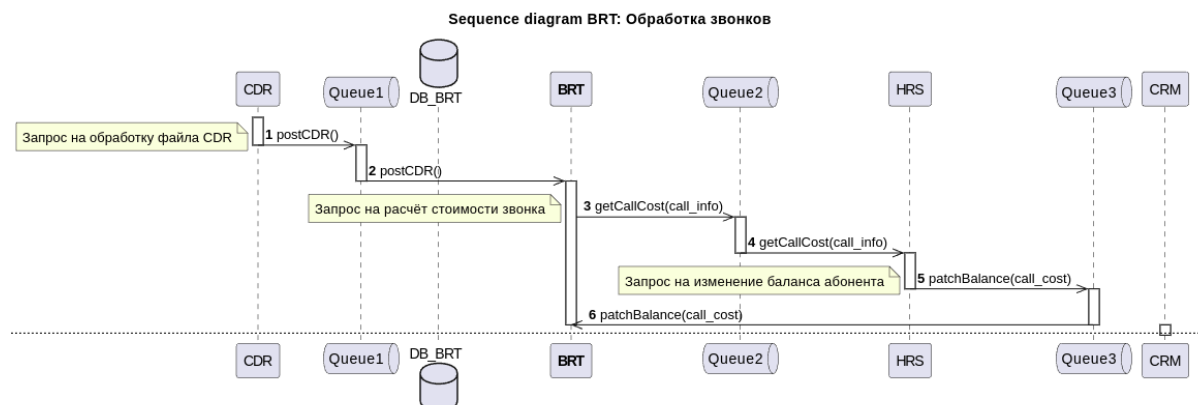


Рисунок 23 - Обработка звонков

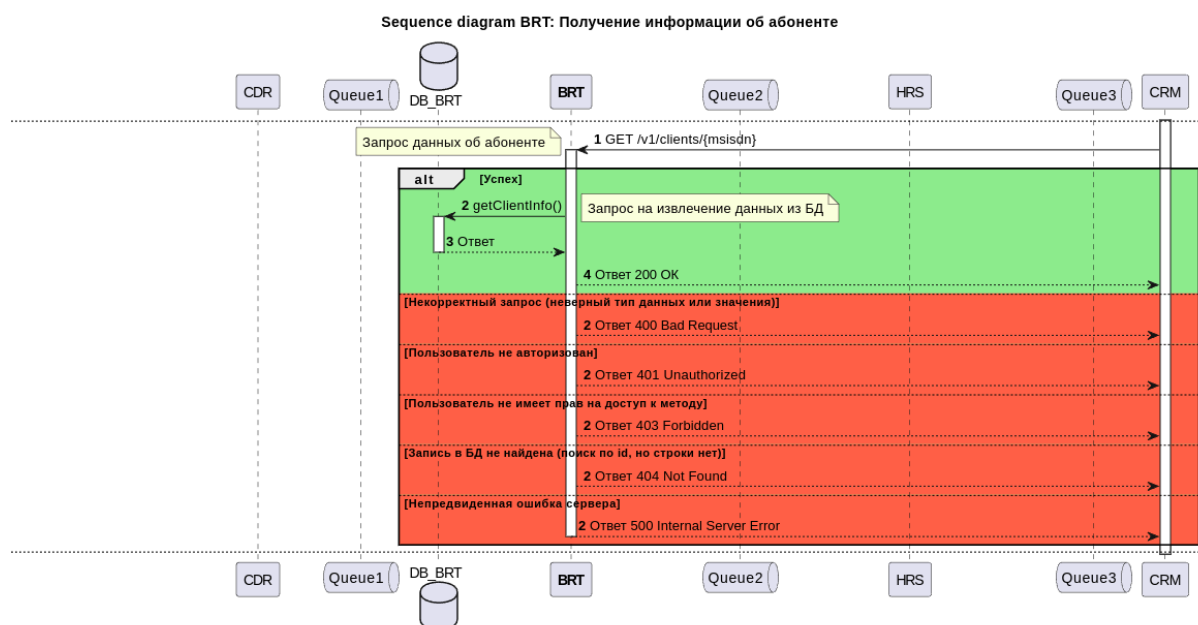


Рисунок 24 - Получение информации об абоненте

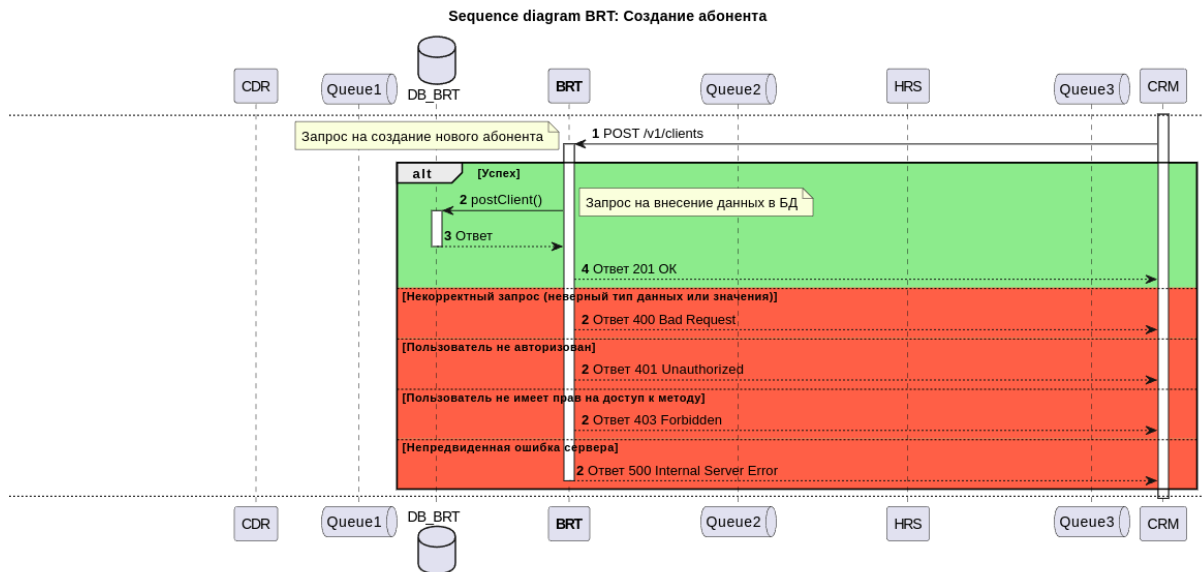


Рисунок 25 - Создание абонента

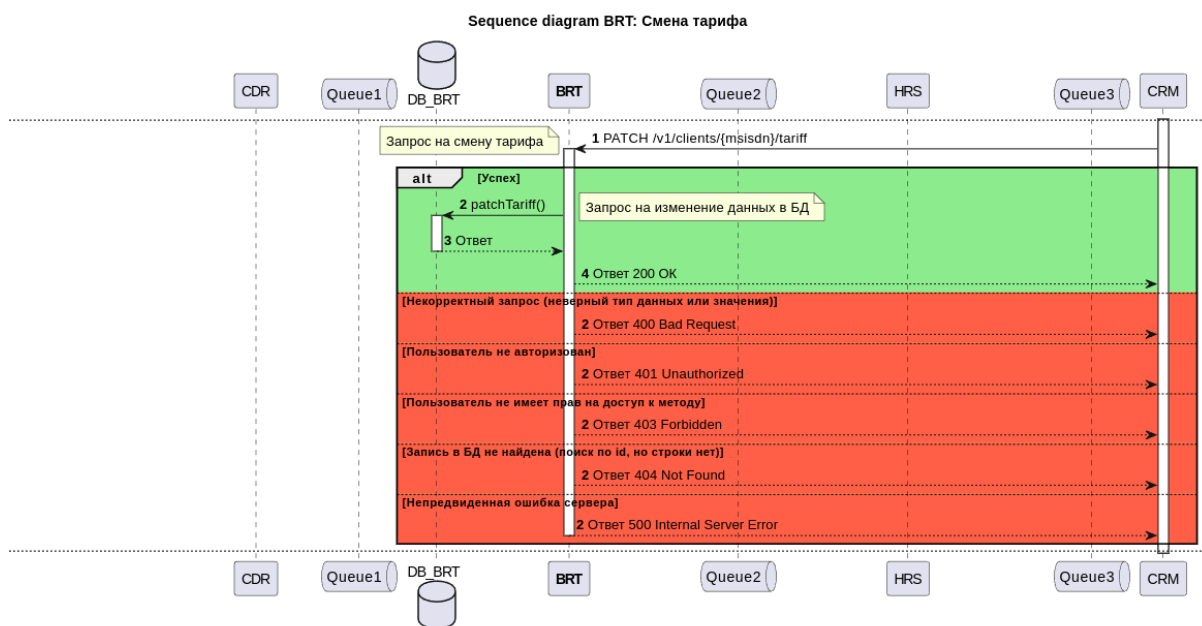


Рисунок 26 - Смена тарифа

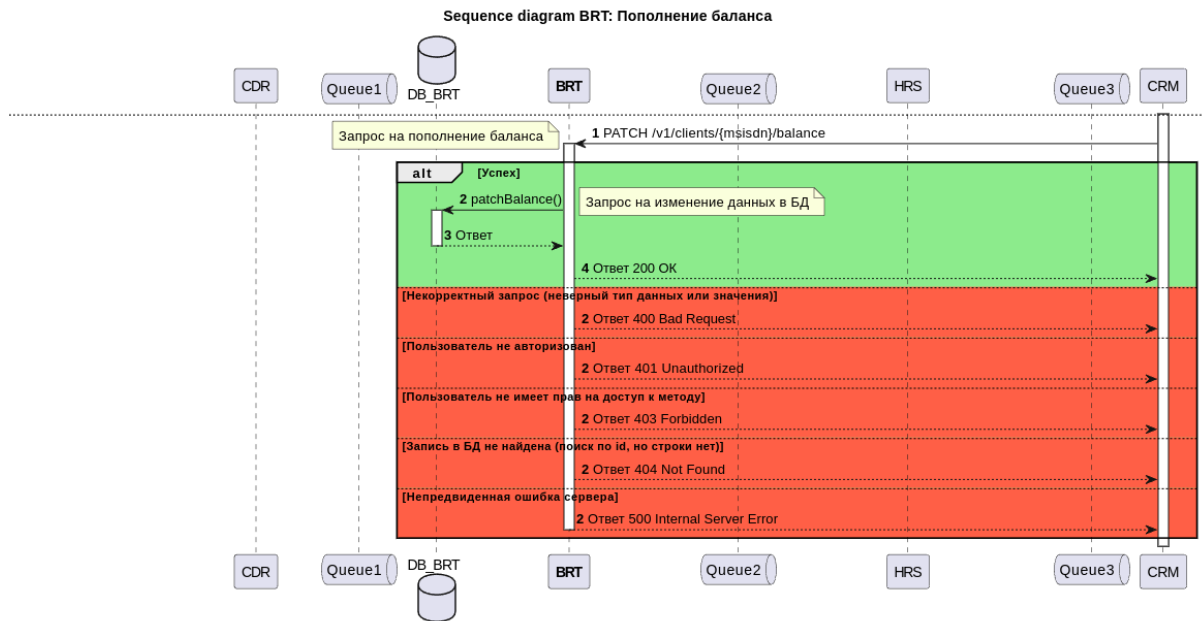


Рисунок 27 - Пополнение баланса

HRS

Концепция сущностей и параметров

Перед началом хотелось бы прояснить пару терминов и концепций, которые послужили основой для проектирования базы данных HRS. В данной работе часто встречается понятие *сущность / entity* и *параметр / parameter*.

Сущность - общее название для группирующих элементов в структуре тарифа. Тариф представляет собой древовидную графовую структуру, где наверху находится главный группирующий элемент, тариф, а под ним различные другие группирующие элементы - узлы / пакеты. Также для каждой сущности можно привязать параметры.

Параметр - общее название для инструкций для тонкой настройки сущностей. Примеры параметров: “Исходящие звонки на Ромашку”, “Входящие звонки от другого оператора”.

Для примера представим некоторую структуру, которая изображена на рисунке 28.

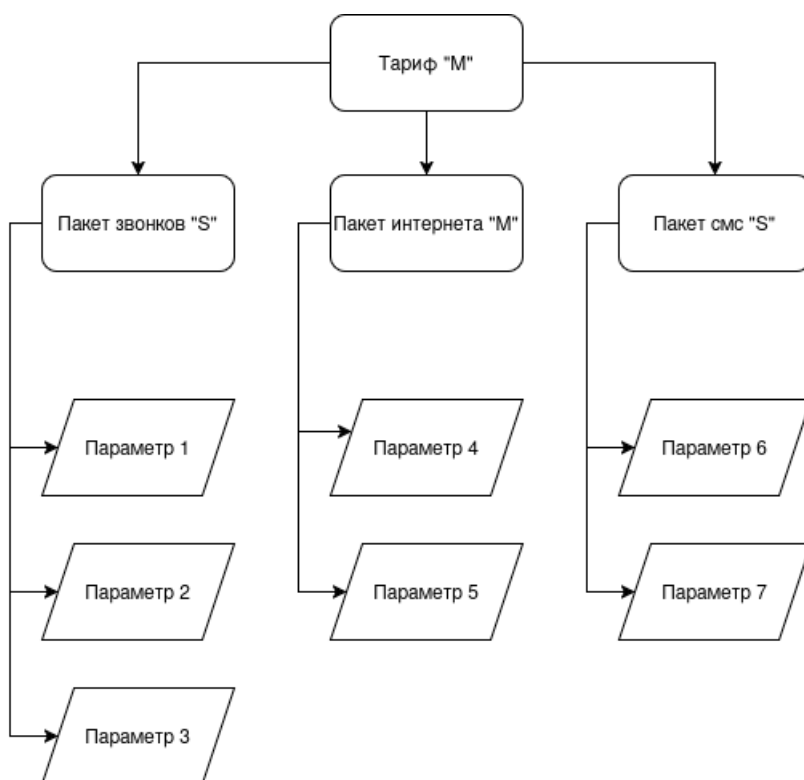


Рисунок 28 - Пример структуры тарифа

Но просто сущности и параметры представляют мало интереса, так как не содержат информации о цене, регионе, лимите ресурса и прочем. Для этого существуют экземпляры сущностей и экземпляры параметров. Такое решение можно сравнить с ООП, где сущность это класс, а экземпляр сущности это объект. Аналогично с параметрами.

Получается так, что мы можем создавать общие тарифы и просто заводить экземпляры для каждого региона с разными ценами, доступностью и т.д. В данной работе не требуется создавать сложноструктурированные тарифы, поэтому такое архитектурное решение служит для гибкой настройки и возможного масштабирования.

Описание логики

В систему HRS поступают из очереди запросы от BRT на расчёт стоимости звонка. Пример данных запроса представлена на рисунке 29.

```
{
  "call_id": 23,
  "call_type": 1,
  "left_msisdn": 79261234567,
  "left_operator_name": "Ромашка",
  "right_msisdn": 79037654321,
  "right_operator_name": "Другой оператор",
  "start_datetime": "2025-05-17T14:30:00",
  "end_datetime": "2025-05-17T14:45:00",
  "entity_instance_id": [3, 4],
  "priority": [1, 1],
  "parameter_instance_id": [1],
  "parameter_value": [50],
  "period_payment_date": ["2025-05-01", null]
}
```

Рисунок 29 - Пример входящего запроса BRT -> HRS

После получения данного json система ищет головную сущность (в нашей работе под ней подразумевается тариф) данного пользователя. Головная сущность указана в массиве entity_instance_id вместе с прочими сущностями. Чтобы определить тариф мы должны перебрать данный массив и в таблице entity_instances найти entity_id.

На рисунке 30 представлена структура таблицы entity_instances в БД HRS, в которой находятся экземпляры сущностей. Комментарии к каждому столбцу написаны справа в скобках. Общий вид БД со всеми таблицами и связями представлен в разделе [ERD](#) данного сервиса.

entity_instances Таблица объектов сущностей				
 entity_instance_id	INT	N-N	UQ	(ИД)
 entity_id	INT	N-N	UQ	(ИД сущности)
 access_id	INT	N-N	UQ	(ИД доступа)
 region_id	INT	N-N	UQ	(ИД региона)
 pricing_id	INT	N-N	UQ	(ИД стоимости)
start_date	DATE	N-N	UQ	(Дата создания объекта)
end_date	DATE	N-N	UQ	(Дата отключения объект...
 status_id	INT	N-N	UQ	(Статус объекта)
description	VARCHAR(50)	NULL	UQ	(Описание)

Рисунок 30 - Таблица entity_instances

После предыдущих шагов уже в таблице entities мы можем узнать к какому типу сущности (тариф / пакет звонков / пакет интернета и т.д.) относится наша entity по полю entity_type_id.

На рисунке 31 представлена структура таблицы entities в БД HRS, в которой находятся сущности (В этой таблице уже находятся сущности как классы, без привязки к региону, цене и прочим параметрам).




entities Таблица сущностей				
 entity_id	INT	N-N	UQ	(ИД)
entity_name	VARCHAR(30)	N-N	UQ	(Название сущности)
 entity_type_id	INT	N-N	UQ	(Тип сущности: Тариф / Пакет звонков / Пакет интернета и т.д.)
description	VARCHAR(50)	NULL	UQ	(Описание)
 fallback_condition_id	INT	NULL	UQ	(ИД условия смены сущности)

Рисунок 31 - Таблица entities

Когда мы убеждаемся, что наш экземпляр сущности из json сообщения это тариф, то дальше мы уже можем в таблице entity_instances посмотреть соответствующее ему pricing_id из таблицы pricing.

На рисунке 32 представлена структура таблицы pricing в БД HRS, в которой находятся поля, связанные с ценой.




pricing	Таблица стоимости сущностей/параметров			
 pricing_id	INT	N-N	UQ	(ИД)
payment_level	INT	N-N	UQ	(Сущность на каком уровне вложенности тарифа отвечает за оплату)
 payment_type_id	INT	N-N	UQ	(ИД типа оплаты)
 currency_id	INT	N-N	UQ	(ИД валюты)
price_for_use	DECIMAL(10,1)	N-N	UQ	(Цена использования)

Рисунок 32 - Таблица pricing

В данной таблице нас интересует payment_level. Так как структура тарифа представляет из себя древовидный граф, то мы можем задать оплату по разным узлам этого графа. Например, оплата может считаться по головному узлу, может по листьям ветвей, а может в узлах между. Для этого мы и указываем уровень вложенности (Например, для тарифа он будет 1, для пакета звонков он будет 2) По данному значению мы можем понять на каком уровне происходит оплата.

Так как в задании нет требования на сложную структуру тарифа, то в данном случае все сделанные тарифы состоят из самой сущности тарифа, пакета звонков и параметров. Тарифы “Классика” реализован с оплатой на уровне 0 (за каждый использованный параметр) и “Помесячный” реализованы с оплатой на уровне 1, то есть на самом верхнем с оплатой раз в период.

Так как мы уже находимся на данном уровне вложенности, то теперь система смотрит на payment_type_id. Этот ИД указывает на тип оплаты сущности (За месяц / Единоразовый / Без оплаты / За единицу времени).

На рисунке 33 представлена структура таблицы payment_types в БД HRS, в которой находятся типы оплаты.


payment_types	Таблица типов оплаты			
 payment_type_id	INT	N-N	UQ	(ИД)
payment_type	VARCHAR(30)	N-N	UQ	(Описание типа оплаты)

Рисунок 33 - Таблица payment_types

Зная тип оплаты, мы можем разобрать два случая тарификации:

- 1) Для тарифа “Классика”
 - 2) Для тарифа “Помесячный”
- 1) **Тариф “Классика”**. Payment_type для тарифа будет называться “Без оплаты”, потому что оплата происходит на уровне вложенности 0, то есть на параметрах. Возвращаемся к полученному json, в котором ещё есть поля parameter_instance_id и parameter_value. parameter_instance_id содержит идентификаторы параметров сущностей. Параметры это значения для тонкой

настройки сущностей. Мы можем сделать параметр “Исходящий звонок на ромашку” и указать у него цену за использование и лимит использования.

На рисунке 34 представлена структура таблицы `parameter_instances` в БД HRS, в которой находятся экземпляры параметров (Как и с сущностями, у параметров есть и экземпляры и классы).




parameter_instances Таблица объектов параметров				
 parameter_instance_id	INT	N-N	UQ	(ИД)
 parameter_id	INT	N-N	UQ	(ИД параметра)
 pricing_id	INT	N-N	UQ	(ИД стоимости)
 entity_instance_id	INT	N-N	UQ	(ИД соответствующей сущности)
description	VARCHAR(50)	NULL	UQ	(Описание)

Рисунок 34 - Таблица `parameter_instances`

По `entity_instance_id` система может понять к какой сущности привязан параметр. В данном случае параметр будет привязан в пакету звонков. Сами параметры с описанием и названием содержатся в таблице `parameters`.

На рисунке 35 представлена структура таблицы `parameters` в БД HRS, в которой находятся базовые параметры.


parameters Таблица параметров				
 parameter_id	INT	N-N	UQ	(ИД)
parameter_name	VARCHAR(30)	N-N	UQ	(Название параметр...
description	VARCHAR(50)	NULL	UQ	(Описание)

Рисунок 35 - Таблица `parameters`

В данном случае у тарифа “Классика” будут следующие параметры:

- “Исходящий на Ромашку” с типом цены “За единицу ресурса” (В данном случае ресурс это минуты) и ценой в 1.5. у.е.
- “Исходящий на другого оператора” с типом цены “За единицу ресурса” и ценой в 2.5. у.е.
- “Входящий” с типом цены “Без оплаты” и ценой 0 у.е.

Лимит на ресурс у каждого параметра будет стоять некоторое огромное число, потому что мы не можем указать бесконечность как лимит, поэтому в таких ситуациях обычной практикой считается указывать сильно большое число.

- 2) **Тариф “Помесячный”**. `Payment_type` для тарифа будет называться “За месяц”. А поле `price_for_use` будет содержать цену за месяц. После подсчета

длительности звонка система отнимет от `parameter_value` количество минут в зависимости от потраченного параметра (В данном случае это параметр “Входящие и исходящие звонки” с количеством ресурса, равным 50 минутам).

Нужна или нет оплата за месяц использования определяется по полю `period_payment_date` в json сообщении, которое поступило на вход в начале.

Если сообщение датируется следующим месяцем после указанной даты, то абоненту вернётся стоимость тарифа как цена звонка.

Теперь, когда система знает как проводить тарификацию, то она может посчитать стоимость (все расчёты округляются до 0.1 у.е.) и количество потраченных минут (минуты определяются с округлением вверх).

Система определяет длительность звонка как разность `start_datetime` и `end_datetime` из запроса и отнимает данное количество минут от лимита (`parameter_value`) и в зависимости от типа оплаты считает сколько средств потратил абонент. Для “Классики” это умножение длительности звонка на использованный параметр, а для тарифа “Помесячный” это оплата раз в месяц.

Теперь система формирует ответный запрос в BRT для внесения изменений данных абонента. Пример такого запроса представлен на рисунке 36.

```
{
  "call_id": 23,
  "msisdn": 79261234567,
  "entity_instance_id": [3, 4],
  "priority": [1, 1],
  "parameter_instance_id": [1],
  "parameter_value": [35],
  "period_payment_date": ["2025-05-01", null],
  "balance_change": 0
}
```

Рисунок 36 - Пример ответного запроса HRS -> BRT

Если возникнет ситуация, что `parameter_value` стал меньше 0, то тариф должен откатиться на другой, с более низким приоритетом. Это определяется по таблице `entity_connections`. На рисунке 37 представлена структура таблицы `entity_connections` в БД HRS, в которой находятся связи сущностей.

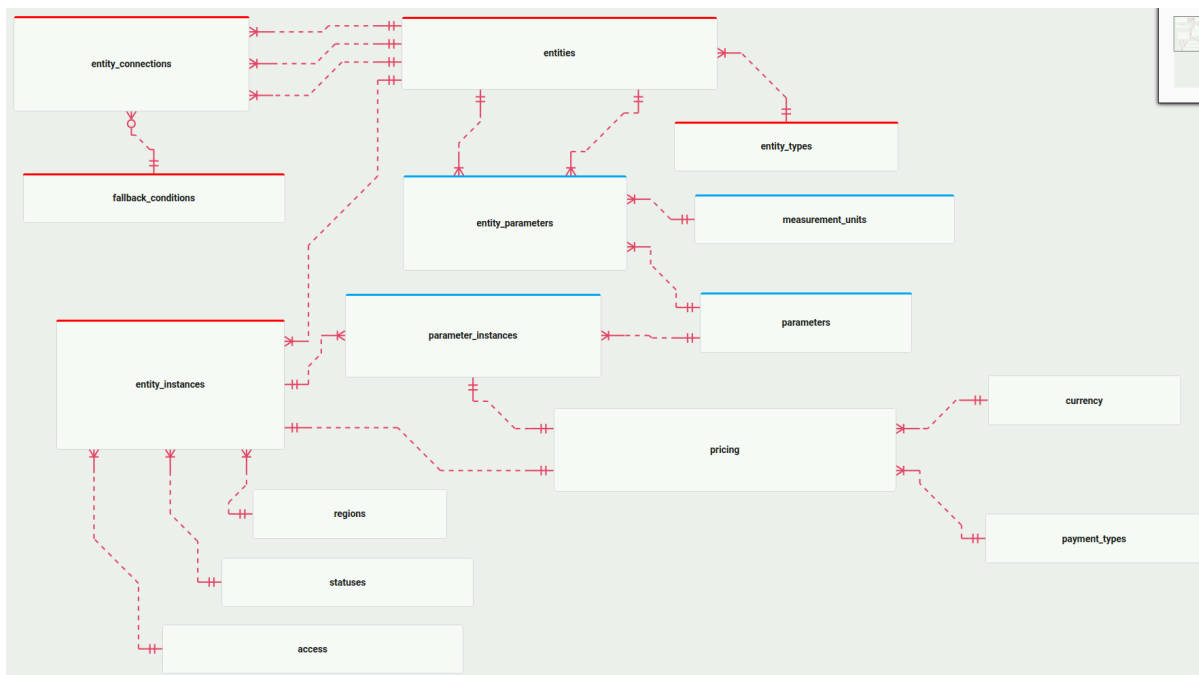


Рисунок 39 - Обобщённая ERD диаграмма сервиса HRS

Описание всех таблиц с изображениями и примерами:

- **Таблица entities** содержит информацию о сущностях. Сущность это общее название для группирующих элементов в построении тарифа. Сущность это и тариф и пакет звонков и т.д. В данном случае тариф и пакет звонков это сущности с разными типами. Такой подход даёт возможность делать очень глубокую вложенность и добавлять новые сущности. Таблица представлена на рисунке 40. Пример данных представлен в таблице 11.

entities	Таблица сущностей			
entity_id	INT	N-N	UQ	(ИД)
entity_name	VARCHAR(30)	N-N	UQ	(Название сущности)
entity_type_id	INT	N-N	UQ	(Тип сущности: Тариф / Пакет звонков / Пакет интернета и т.д.)
description	VARCHAR(50)	NULL	UQ	(Описание)

Рисунок 40 - Таблица entities

Таблица 11 - Пример для entities

entity_id	12
entity_name	Помесячный
entity_type_id	3

description	
-------------	--

- **Таблица entity_connections** содержит всю информацию о связях сущностей. Сущности связываются для создания структуры тарифа. Например, мы можем привязать к головной сущности (тариф) дочернюю сущность (пакет звонков). Также мы можем привязать несколько сущностей к тарифу с одним типом, указав разный приоритет. Тогда по истечении ресурса и удовлетворения условия перехода привязанная сущность заменится на сущность с более низким приоритетом.

Таблица представлена на рисунке 41.

Пример данных представлен в таблице 12.

entity_connections Таблица связи сущностей					
	entity_connection_id	INT	N-N	UQ	(ИД)
	head_entity_id	INT	N-N	UQ	(Корневая сущность)
	parent_entity_id	INT	N-N	UQ	(Родительская сущность)
	child_entity_id	INT	N-N	UQ	(Дочерняя сущность)
	priority	INT	N-N	UQ	(Приоритет)
	fallback_condition_id	INT	N-N	UQ	(ИД условия смены сущности)

Рисунок 41 - Таблица entity_connections

Таблица 12 - Пример для entity_connections

entity_connection_id	11
head_entity_id	3
parent_entity_id	3
child_entity_id	2
priority	1
fallback_condition_id	1

- **Таблица fallback_conditions** содержит информацию об условиях замены сущности другой.

Таблица представлена на рисунке 42.

Пример данных представлен в таблице 13.

fallback_conditions Таблица условий смены сущностей

🔑 fallback_condition_id	INT	N-N	UQ	(ИД)
fallback_condition	VARCHAR(50)	N-N	UQ	(Описание условия смены сущности)

Рисунок 42 - Таблица fallback_conditions

Таблица 13 - Пример для fallback_conditions

fallback_condition_id	2
fallback_condition	Закончился ресурс

- **Таблица entity_types** содержит всю информацию о типах сущностей
Таблица представлена на рисунке 43.
Пример данных представлен в таблице 14.

entity_types Таблица типов сущностей

🔑 entity_type_id	INT	N-N	UQ	(ИД)
entity_type_name	VARCHAR(30)	N-N	UQ	(Название типа сущности)

Рисунок 43 - Таблица entity_types

Таблица 14 - Пример для entity_types

entity_type_id	29
entity_type_name	Тариф

- **Таблица entity_parameters** содержит информацию о связи сущности и параметра.
Таблица представлена на рисунке 44.
Пример данных представлен в таблице 15.

entity_parameters Таблица параметров сущностей				
 entity_parameter_id	int	N-N	UQ	(ИД)
 head_entity_id	INT	N-N	UQ	(ИД головной сущности)
 entity_id	INT	N-N	UQ	(ИД сущности)
 measurement_unit_id	INT	N-N	UQ	(ИД единицы измерения)
 parameter_id	INT	N-N	UQ	(ИД параметра)
value	INT	N-N	UQ	(Значение параметра)

Рисунок 44 - Таблица entity_parameters

Таблица 15 - Пример для entity_parameters

entity_parameter_id	31
head_entity_id	1
entity_id	2
measurement_unit_id	2
parameter_id	1
value	50

- **Таблица measurement_units** содержит информацию об единицах измерения. Таблица представлена на рисунке 45. Пример данных представлен в таблице 16.


measurement_units Таблица единиц измерения				
 measurement_unit_id	INT	N-N	UQ	(ИД)
measurement_unit_name	VARCHAR(30)	N-N	UQ	(Название единицы измерения)

Рисунок 45 - Таблица measurement_units

Таблица 16 - Пример для measurement_units

measurement_unit_id	15
measurement_unit_name	Минуты

- **Таблица entity_instances** содержит всю информацию об экземплярах сущностей. Для понятности можно представить entity как класс в ООП, а

entity_instance как экземпляр. Таким образом, мы можем создавать массу тарифов для разных регионов с разными ценами и доступами. Таблица представлена на рисунке 46. Пример данных представлен в таблице 17.

entity_instances Таблица объектов сущностей					
	entity_instance_id	INT	N-N	UQ	(ИД)
	entity_id	INT	N-N	UQ	(ИД сущности)
	access_id	INT	N-N	UQ	(ИД доступа)
	region_id	INT	N-N	UQ	(ИД региона)
	pricing_id	INT	N-N	UQ	(ИД стоимости)
	start_date	DATE	N-N	UQ	(Дата создания объекта)
	end_date	DATE	NULL	UQ	(Дата отключения объекта)
	status_id	INT	N-N	UQ	(Статус объекта)
	description	VARCHAR(50)	NULL	UQ	(Описание)

Рисунок 46 - Таблица entity_instances

Таблица 17 - Пример для entity_instances

entity_instance_id	25
entity_id	1
access_id	2
region_id	1
pricing_id	7
start_date	2024-05-17
end_date	null
status_id	2
description	

- **Таблица parameter_instances** содержит информацию об экземплярах параметров. Таблица представлена на рисунке 47. Пример данных представлен в таблице 18.




parameter_instances Таблица объектов параметров					
 parameter_instance_id	INT	N-N	UQ	(ИД)	
 parameter_id	INT	N-N	UQ	(ИД параметра)	
 pricing_id	INT	N-N	UQ	(ИД стоимости)	
 entity_instance_id	INT	N-N	UQ	(ИД соответствующей сущности)	
description	VARCHAR(50)	NULL	UQ	(Описание)	

Рисунок 47 - Таблица parameter_instances

Таблица 18 - Пример для parameter_instances

parameter_instance_id	10
parameter_id	1
pricing_id	17
entity_instance_id	2
description	

- **Таблица parameters** содержит информацию о параметрах сущностей. В параметры указываются все детали сущности (Исходящие звонки, входящие звонки и т.д.)

Таблица представлена на рисунке 48.

Пример данных представлен в таблице 19.


parameters Таблица параметров					
 parameter_id	INT	N-N	UQ	(ИД)	
parameter_name	VARCHAR(30)	N-N	UQ	(Название параметр...	
description	VARCHAR(50)	NULL	UQ	(Описание)	

Рисунок 48 - Таблица parameters

Таблица 19 - Пример для parameters

parameter_id	2
parameter_name	Исходящие на Ромашку
description	

- **Таблица pricing** содержит информацию о стоимости услуг. Данная таблица может использоваться как сущностями, так и параметрами.
Таблица представлена на рисунке 49.
Пример данных представлен в таблице 20.




pricing	Таблица стоимости сущностей/параметров			
 pricing_id	INT	N-N	UQ	(ИД)
 payment_type_id	INT	N-N	UQ	(ИД типа оплаты)
 currency_id	INT	N-N	UQ	(ИД валюты)
price_for_use	DECIMAL(10,1)	N-N	UQ	(Цена использования)

Рисунок 49 - Таблица pricing

Таблица 20 - Пример для pricing

pricing_id	4
payment_type_id	1
currency_id	1
price_for_use	100

- **Таблица currency** содержит информацию о валюте.
Таблица представлена на рисунке 50.
Пример данных представлен в таблице 21.

currency	Таблица валюты			
 currency_id	INT	N-N	UQ	(ИД)
currency_name	varchar(20)	N-N	UQ	(Название валют...

Рисунок 50 - Таблица currency

Таблица 21 - Пример для currency

currency_id	6
currency_name	y.e.

- **Таблица payment_types** содержит информацию о типе оплаты.
Таблица представлена на рисунке 51.
Пример данных представлен в таблице 22.

payment_types Таблица типов оплаты

🔑 payment_type_id	INT	N-N	UQ	(ИД)
payment_type	VARCHAR(30)	N-N	UQ	(Описание типа оплаты)

Рисунок 51 - Таблица payment_types

Таблица 22 - Пример для payment_types

payment_type_id	1
payment_type	За месяц

- **Таблица regions** содержит информацию о регионах. В данном задании не стояло задачи обрабатывать тарифы с разных регионов, но такая возможность была предусмотрена.
Таблица представлена на рисунке 52.
Пример данных представлен в таблице 23.

regions Таблица регионов

🔑 region_id	INT	N-N	UQ	(ИД)
region_name	VARCHAR(30)	N-N	UQ	(Название региона)

Рисунок 52 - Таблица regions

Таблица 23 - Пример для regions

region_id	1
region_name	Россия

- **Таблица statuses** содержит всю информацию о статусах сущностей.
Таблица представлена на рисунке 53.
Пример данных представлен в таблице 24.

statuses Таблица статусов

🔑 status_id	INT	N-N	UQ	(ИД)
status_name	VARCHAR(30)	NULL	UQ	(Название статуса: Активный / В архиве)

Рисунок 53 - Таблица statuses

Таблица 24 - Пример для statuses

status_id	1
status_name	Активный

- **Таблица access** содержит информацию о доступе абонента к тарифу
Таблица представлена на рисунке 54.
Пример данных представлен в таблице 25.


access	Таблица доступа			
 access_id	INT	N-N	UQ	(ИД)
access_name	VARCHAR(30)	N-N	UQ	(Название доступа к тарифу: Открытый / Корпоративный)

Рисунок 54 - Таблица access

Таблица 25 - Пример для access

access_id	3
access_name	Открытый

Use Case & User Story

В таблице 26 приведен перечень Use Case и User Story.

Таблица 26 - HRS UC & US

	Вариант использования	Пользовательская история
1	Расчет тарификации “Классика”	Как менеджер, я хочу рассчитать стоимость звонка, чтобы знать сколько средств списать со счёта абонента
2	Расчет тарификации “Помесячный”	Как менеджер, я хочу рассчитать стоимость звонка, чтобы знать сколько средств списать со счёта абонента

1. User Story:

Как менеджер, я хочу рассчитать стоимость звонка, чтобы знать сколько средств списать со счёта абонента

1. Use Case:

Название: Расчет тарификации “Классика”

Триггер: Пришёл новый запрос из BRT

Предусловия:

- Тариф абонента есть в таблице tariffs БД HRS
- Номер абонента относится к оператору “Ромашка”
- Система находится в запущенном состоянии

Результат: В BRT отправлена сумма списания средств абонента

Основной поток:

1. Система определяет тип тарифа
2. Система определяет тип звонка (исходящий или входящий)
3. Система определяет длительность звонка, округляя вверх до минуты
4. Система ведёт подсчёт стоимости звонка по длительности и цене за минуту разговора
5. Система отправляет сумму списания в BRT

Альтернативные потоки:

1а. Тип звонка - входящий

1а.1 Система указывает сумму списания как 0 у.е

1а.2 Возврат к 5 шагу основного потока

2. User Story:

Как менеджер, я хочу рассчитать стоимость звонка, чтобы знать сколько средств списать со счёта абонента

2. Use Case:

Название: Расчет тарификации “Помесячный”

Триггер: Пришёл новый запрос из BRT

Предусловия:

- Тариф абонента есть в таблице tariffs БД HRS
- Номер абонента относится к оператору “Ромашка”
- Система находится в запущенном состоянии

Результат: В BRT отправлена сумма списания средств абонента

Основной поток:

1. Система определяет тип тарифа
2. Система определяет тип звонка (исходящий или входящий)
3. Система проверяет баланс минут абонента
4. Система определяет длительность звонка, округляя вверх до минуты
5. Система отправляет длительность звонка в BRT

Альтернативные потоки:

1а. Минуты тарифа закончились

1а.1 Система рассчитывает продолжительность звонка

1а.2 Система считает стоимость разговора, согласно тарифу “Классика”

1а.2 Система отправляет сумму списания в BRT

Sequence Diagram

Результат построения sequence diagram представлен на рисунке 55.

[Код для PlantUml](#)

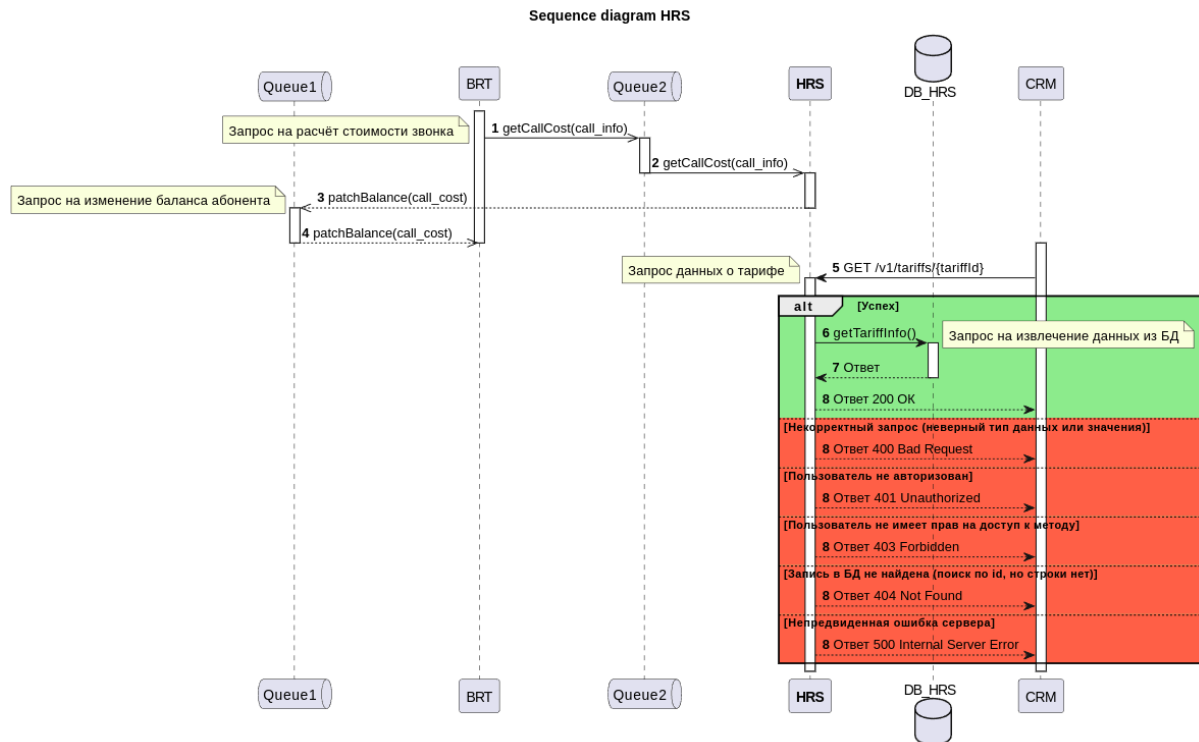


Рисунок 55 - Общая Sequence диаграмма для сервиса HRS

CRM

Описание логики

Сервис CRM предоставляет интерфейс для работы с абонентами оператора сотовой связи. Он поддерживает две роли: абонент и менеджер. Авторизация абонента происходит по номеру телефона, после чего система проверяет наличие пользователя в таблице clients БД BRT. Если абонент не найден — возвращается ошибка.

Менеджер авторизуется по логину и паролю, которые хранятся в БД CRM.

Менеджер может создавать новых абонентов: при создании указывается номер, имя и стартовый тариф. Новый пользователь автоматически получает баланс 100 у.е., а все данные сохраняются в таблицу clients БД BRT. Также менеджер имеет возможность менять тариф существующему абоненту — при этом система проверяет корректность указанного тарифа и наличие абонента в системе. В случае успеха тариф обновляется.

Менеджер может пополнять баланс абонента на указанную сумму. Абонент также способен пополнять свой баланс.

Дополнительно CRM предоставляет метод получения полной информации об абоненте. Запрос доступен только менеджеру и возвращает ФИО, номер, текущий баланс, регион, информацию о тарифе, а также дату регистрации.

Entity Relationship Diagram

Общий вид ERD для данного сервиса представлен на рисунках 56 и 57. Часть информации по таблицам, которая нужна была для рассмотрения логики работы системы представлена в разделе [Описание логики](#).

[Онлайн версия в большем разрешении](#)

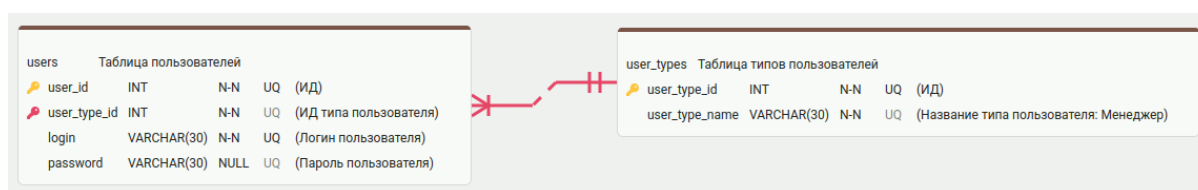


Рисунок 56 - ERD диаграмма сервиса CRM



Рисунок 57 - Обобщённая ERD диаграмма сервиса CRM

Описание всех таблиц с изображениями и примерами:

- **Таблица users** содержит всю информацию о пользователях CRM

Таблица представлена на рисунке 58.
Пример данных представлен в таблице 27.

users	Таблица пользователей				
 user_id	INT	N-N	UQ	(ИД)	
 user_type_id	INT	N-N	UQ	(ИД типа пользователя)	
login	VARCHAR(30)	N-N	UQ	(Логин пользователя)	
password	VARCHAR(30)	NULL	UQ	(Пароль пользователя)	

Рисунок 58 - Таблица users

Таблица 27 - Пример для users

user_id	1
user_type_id	1
login	admin
password	admin

- **Таблица user_types** содержит информацию о типах пользователей CRM
Таблица представлена на рисунке 59.
Пример данных представлен в таблице 28.

user_types Таблица типов пользователей					
	user_type_id	INT	N-N	UQ	(ИД)
	user_type_name	VARCHAR(30)	N-N	UQ	(Название типа пользователя: Менеджер)

Рисунок 59 - Таблица user_types

Таблица 28 - Пример для user_types

user_type_id	1
user_type_name	Менеджер

Use Case & User Story

В таблице 29 приведен перечень Use Case и User Story.

Таблица 29 - CRM UC & US

	Вариант использования	Пользовательская история
1	Авторизация менеджера	Как менеджер, я хочу авторизоваться в сервисе, чтобы управлять данными абонентов
2	Получение информации об абоненте	Как менеджер, я хочу получить информацию об абоненте и его тарифе, чтобы понимать текущие условия обслуживания
3	Создание абонента	Как менеджер, я хочу зарегистрировать нового абонента, чтобы он мог пользоваться услугами оператора
4	Смена тарифа	Как менеджер, я хочу менять тариф пользователям, чтобы удовлетворить их потребность
5	Пополнение баланса менеджером	Как менеджер, я хочу пополнять баланс абонентов, чтобы они могли пользоваться услугами оператора
6	Авторизация абонента	Как абонент, я хочу авторизоваться в сервисе, чтобы пополнять свой баланс
7	Пополнение баланса абонентом	Как абонент, я хочу пополнить баланс, чтобы иметь возможность пользоваться услугами оператора

1. User Story:

Как менеджер, я хочу авторизоваться в сервисе, чтобы управлять данными абонентов

1. Use Case:

Название: Авторизация менеджера

Триггер: Нажата кнопка входа в аккаунт

Предусловия:

- Система активна

Результат: Менеджер авторизован

Основной поток:

1. Система отображает форму для входа в аккаунт
2. Менеджер вводит логин и пароль
3. Менеджер нажимает кнопку “Войти”
4. Система валидирует введенные данные (заполненность обязательных полей и формат введенных данных)
5. Система проверяет, существует ли данный менеджер в системе

6. Система проверяет, соответствие введенных логина и пароля данным аккаунта менеджера
7. Система возвращает ответ об успешной авторизации
8. Система переводит пользователя на админку менеджера

Альтернативные потоки:

- 4a. Использованы некорректные символы
 - 4a.1. Система очищает поле ввода пароля
 - 4a.2 Система подсвечивает некорректные поля
 - 4a.3 Система выводит памятку о запрещённых символах
 - 4a.4 Возврат к шагу 1 основного потока
- 5a. Менеджер отсутствует в системе
 - 5a.1. Система очищает поле ввода пароля
 - 5a.2 Система выводит ошибку, что введены неверные логин или пароль
 - 5a.3 Возврат к шагу 2 основного потока
- 6a. Неверный логин или пароль
 - 6a.1 Система очищает поле ввода пароля
 - 6a.2 Система выводит ошибку, что введены неверные логин или пароль
 - 6a.3 Возврат к шагу 2 основного потока

2. User Story:

Как менеджер, я хочу получить информацию об абоненте и его тарифе, чтобы понимать текущие условия обслуживания

2. Use Case:

Название: Получение информации об абоненте

Триггер: Менеджер нажимает на поле ввода номера абонента

Предусловия:

- Менеджер авторизован
- Абонент существует в системе
- Система активна

Результат: Система отобразила данные абонента

Основной поток:

1. Менеджер вводит номер абонента
2. Менеджер нажимает на кнопку поиск
3. Система проверяет корректность введенных данных
4. Система производит поиск абонента с данным номером в системе
5. Система собирает информацию о тарифе по id тарифа в БД абонентов
6. Система переводит менеджера на страницу с данными абонента и тарифа

Альтернативные потоки:

- 3a. Использованы некорректные символы
 - 3a.1 Система возвращает ошибку о введенных данных
 - 3a.2 Система выводит памятку о запрещённых символах
 - 3a.3 Возврат к шагу 1 основного потока

4а. Абонент отсутствует в системе

4а.1 Система выводит сообщение, что данный абонент не найден

4а.2 Возврат к шагу 1 основного потока

3. User Story:

Как менеджер, я хочу зарегистрировать нового абонента, чтобы он мог пользоваться услугами оператора

3. Use Case:

Название: Создание абонента

Триггер: Менеджер нажимает на кнопку “Создать нового абонента”

Предусловия:

- Менеджер авторизован
- Система активна
- В БД тарифов есть хотя бы один тариф

Результат: Создан новый абонент в системе

Основной поток:

1. Система открывает форму создания абонента
2. Менеджер вводит данные абонента
3. Менеджер нажимает кнопку “Далее”
4. Система отображает выпадающий список с выбором тарифа на основе информации абонента
5. Менеджер выбирает абоненту тариф
6. Менеджер нажимает кнопку “Сохранить”
7. Система проверяет, зарегистрирован ли уже абонент с этим номером
8. Система проверяет доступность тарифа
9. Система заносит абонента в таблицу clients БД BRT
10. Система присваивает данному абоненту баланс в размере 100 у.е.
11. Система выводит сообщение об успешной регистрации
12. Система переводит менеджера на страницу информации о созданном абоненте

Альтернативные потоки:

3а. Отсутствуют доступные тарифы

3а.1 Система возвращает ошибку: «По данным абонента не удалось найти доступных тарифов».

3а.2 Возврат к 1 шагу основного потока

7а. Номер уже зарегистрирован

7а.1 Система возвращает ошибку: «Абонент с таким номером уже существует».

7а.2 Возврат к 1 шагу основного потока

7б. Номера телефона не существует

7а.1 Система возвращает ошибку: «Данного номера телефона не существует».

7а.2 Возврат к 1 шагу основного потока

4. User Story:

Как менеджер, я хочу менять тариф пользователям, чтобы удовлетворить их потребность

4. Use Case:

Название: Смена тарифа

Триггер: Менеджер нажимает кнопку смены тарифа на странице абонента

Предусловия:

- Менеджер авторизован
- Абонент существует в системе
- Система активна
- Менеджер открыл страницу информации об абоненте

Результат: Абоненту присвоен новый тариф

Основной поток:

1. Система отображает форму смены тарифа
2. Система отображает выпадающий список тарифов на основе информации абонента
3. Менеджер выбирает нужный тариф
4. Менеджер нажимает кнопку “Сохранить”
5. Система изменяет id тарифа абонента в таблице clients БД BRT
6. Система возвращает сообщение об успешном изменении тарифа
7. Система открывает страницу информации об абоненте

5. User Story:

Как менеджер, я хочу пополнять баланс абонентов, чтобы они могли пользоваться услугами оператора

5. Use Case:

Название: Пополнение баланса менеджером

Триггер: Менеджер нажимает кнопку пополнения баланса на странице абонента

Предусловия:

- Менеджер авторизован
- Абонент существует в системе
- Система активна
- Менеджер открыл страницу информации об абоненте

Результат: Баланс абонента пополнен

Основной поток:

1. Система отображает форму пополнения баланса
2. Менеджер вводит в поле сумму пополнения
3. Менеджер нажимает кнопку “Сохранить”
4. Система проверяет корректность введенных данных
5. Система переводит менеджера на страницу информации об абоненте

Альтернативные потоки:

- 4а. Используются некорректные символы

4a.1 Система возвращает ошибку: «Использованы запрещенные символы».

4a.2 Система отображает памятку с разрешенными символами

4a.3 Возврат к 1 шагу основного потока

6. User Story:

Как абонент, я хочу авторизоваться в сервисе, чтобы пополнять свой баланс

6. Use Case:

Название: Авторизация абонента

Триггер: Нажата кнопка входа в аккаунт

Предусловия:

- Система активна

Результат: Абонент авторизован

Основной поток:

1. Система отображает форму для входа в аккаунт
2. Абонент вводит номер телефона
3. Абонент нажимает кнопку “Войти”
4. Система валидирует введенные данные (заполненность поля номера телефона и формат введенных данных)
5. Система проверяет, существует ли данный абонент в системе
6. Система возвращает ответ об успешной авторизации
7. Система переводит абонента на страницу профиля

Альтернативные потоки:

4a. Использованы некорректные символы

4a.1 Система подсвечивает некорректное поле

4a.2 Система выводит памятку о запрещённых символах

4a.3 Возврат к шагу 1 основного потока

5a. Абонент отсутствует в системе

5a.1 Система выводит ошибку, что номер телефона отсутствует в системе

5a.2 Возврат к шагу 1 основного потока

7. User Story:

Как абонент, я хочу пополнить баланс, чтобы иметь возможность пользоваться услугами оператора

7. Use Case:

Название: Пополнение баланса абонентом

Триггер: Абонент нажимает кнопку пополнения баланса на странице профиля

Предусловия:

- Абонент авторизован
- Абонент существует в системе
- Система активна

Результат: Баланс абонента пополнен

Основной поток:

1. Система отображает форму пополнения баланса
2. Абонент вводит в поле сумму пополнения
3. Абонент нажимает кнопку “Сохранить”
4. Система проверяет корректность введенных данных
5. Система переводит абонента на страницу профиля

Альтернативные потоки:

- 4а. Использованы некорректные символы
 - 4а.1 Система возвращает ошибку: «Использованы запрещенные символы».
 - 4а.2 Система отображает памятку с разрешенными символами
 - 4а.3 Возврат к 1 шагу основного потока

Use Case Diagram

Результат построения use case diagram представлен на рисунке 60.

[Код для PlantUml](#)

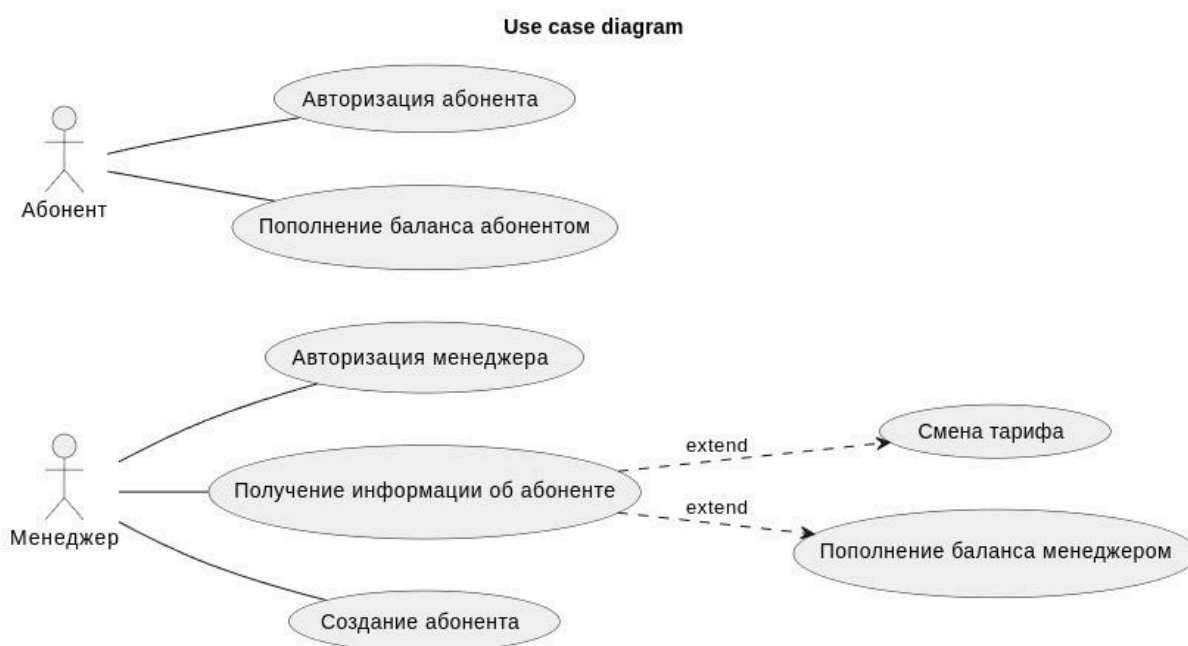


Рисунок 60 - Use Case диаграмма

Sequence Diagram

Результат построения sequence diagram представлен на рисунках 61 - 67.

[Код для PlantUml](#)

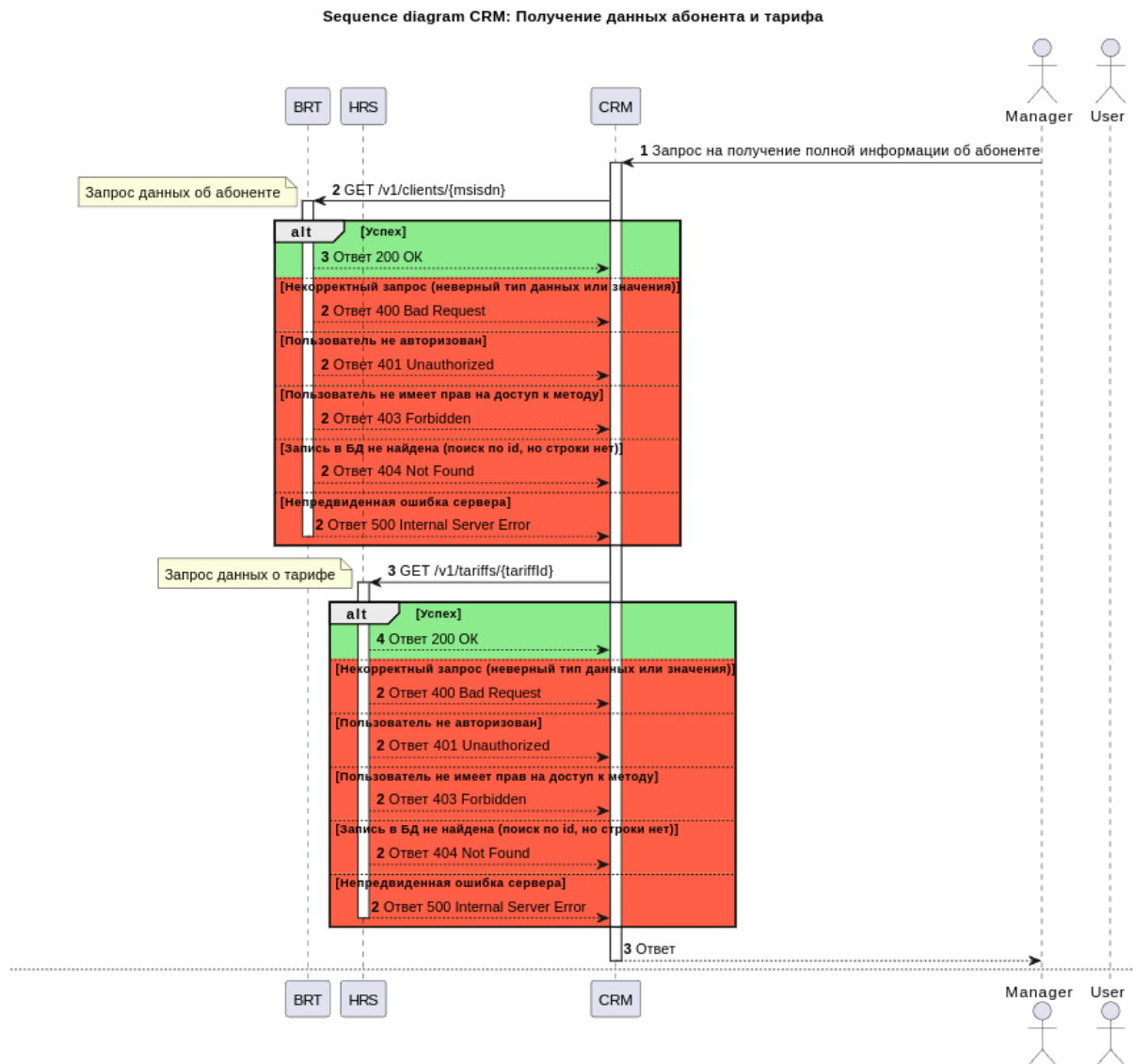


Рисунок 61 - Получение данных абонента и тарифа

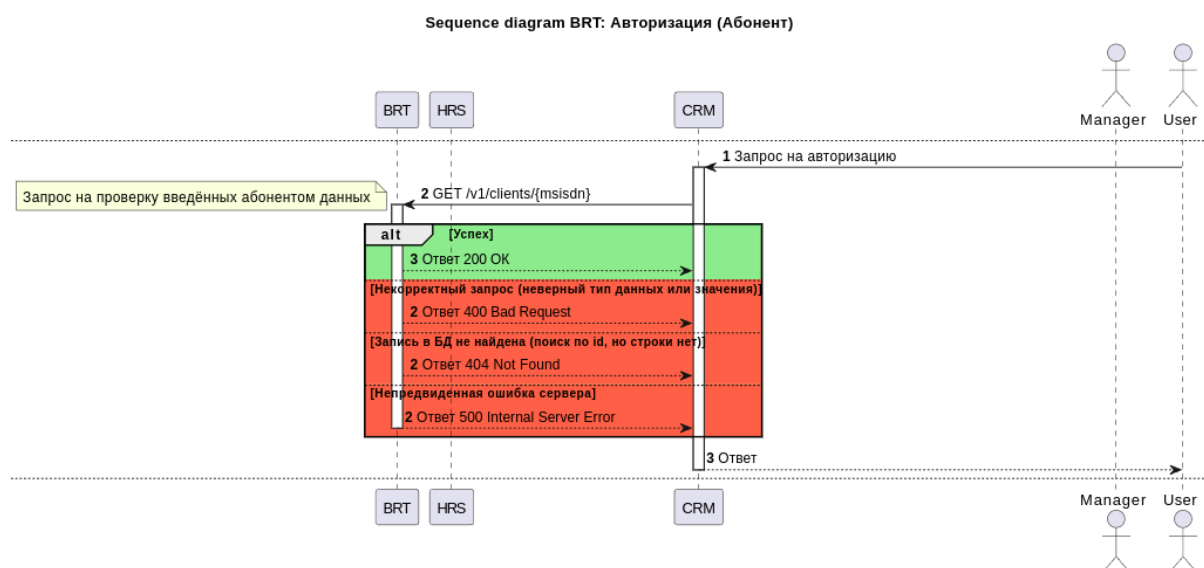


Рисунок 62 - Авторизация абонента

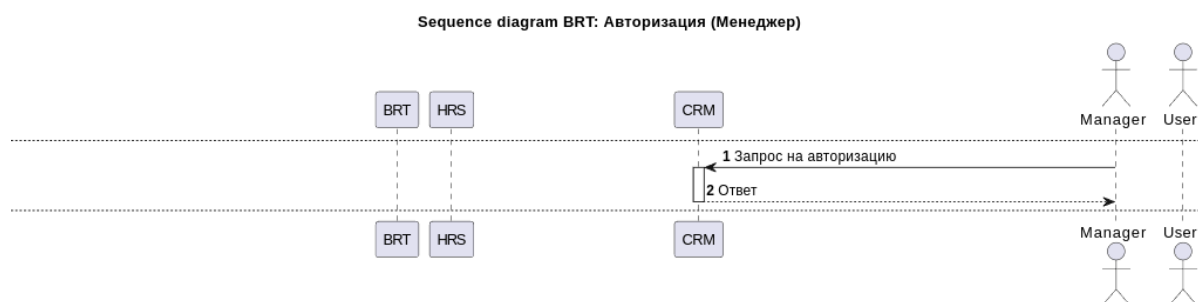


Рисунок 63 - Авторизация менеджера

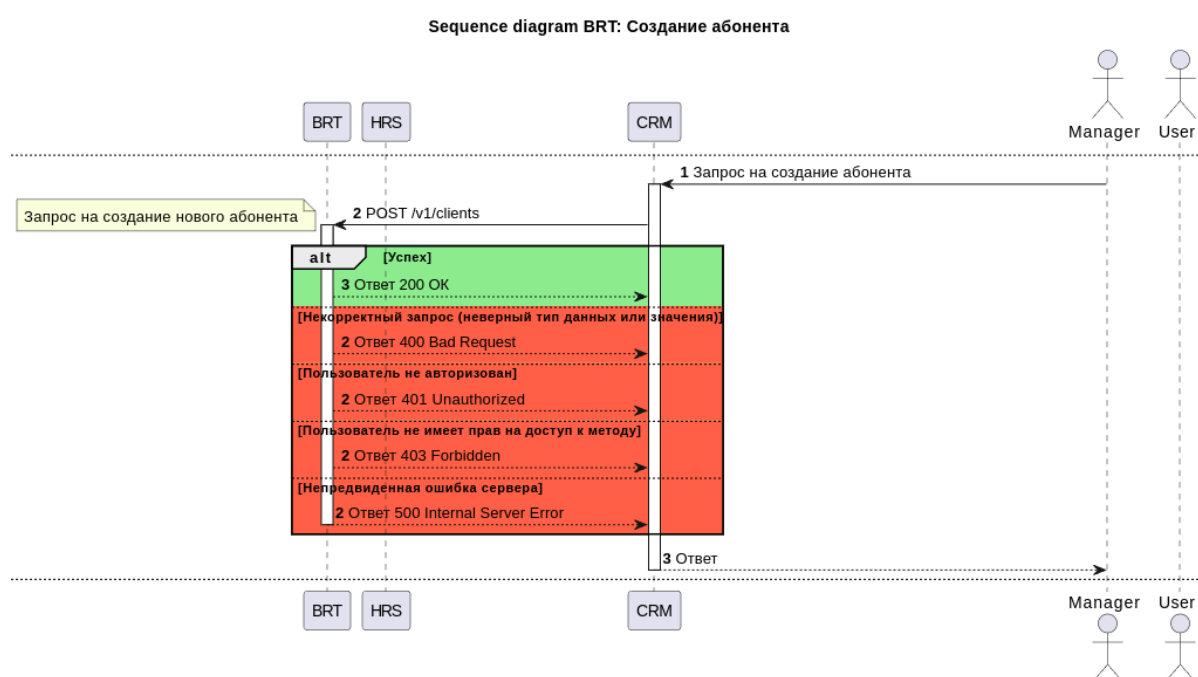


Рисунок 64 - Создание абонента

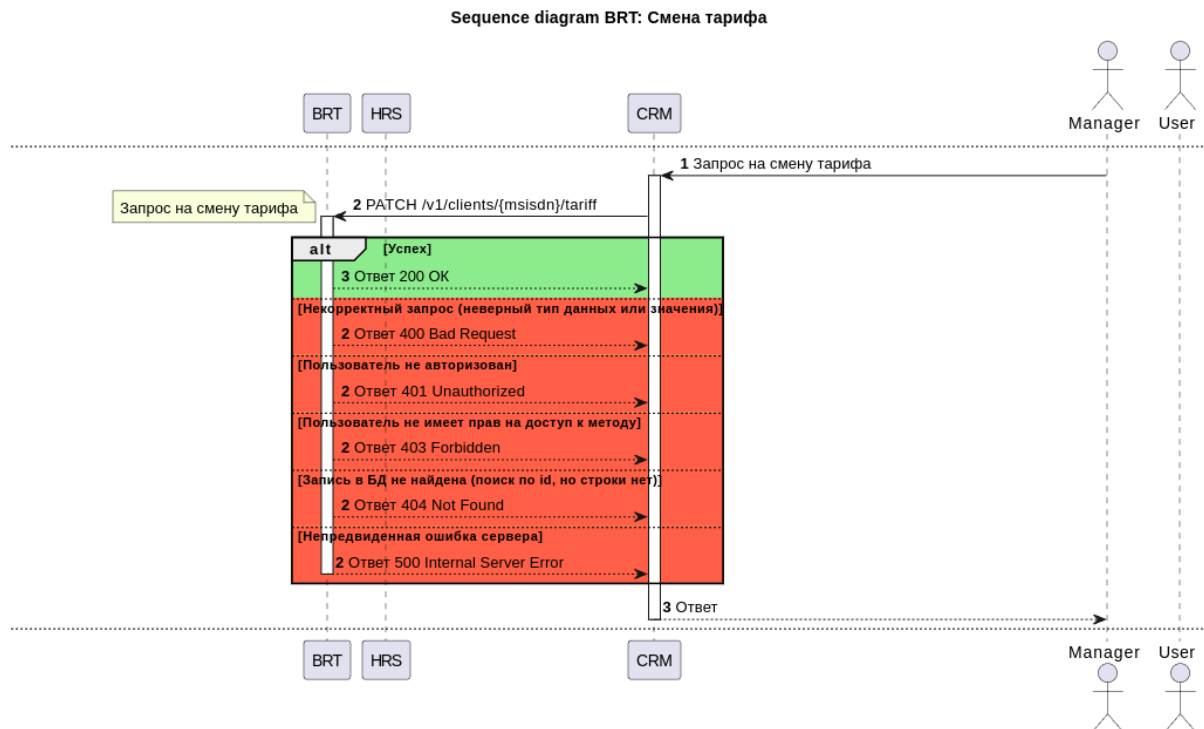


Рисунок 65 - Смена тарифа

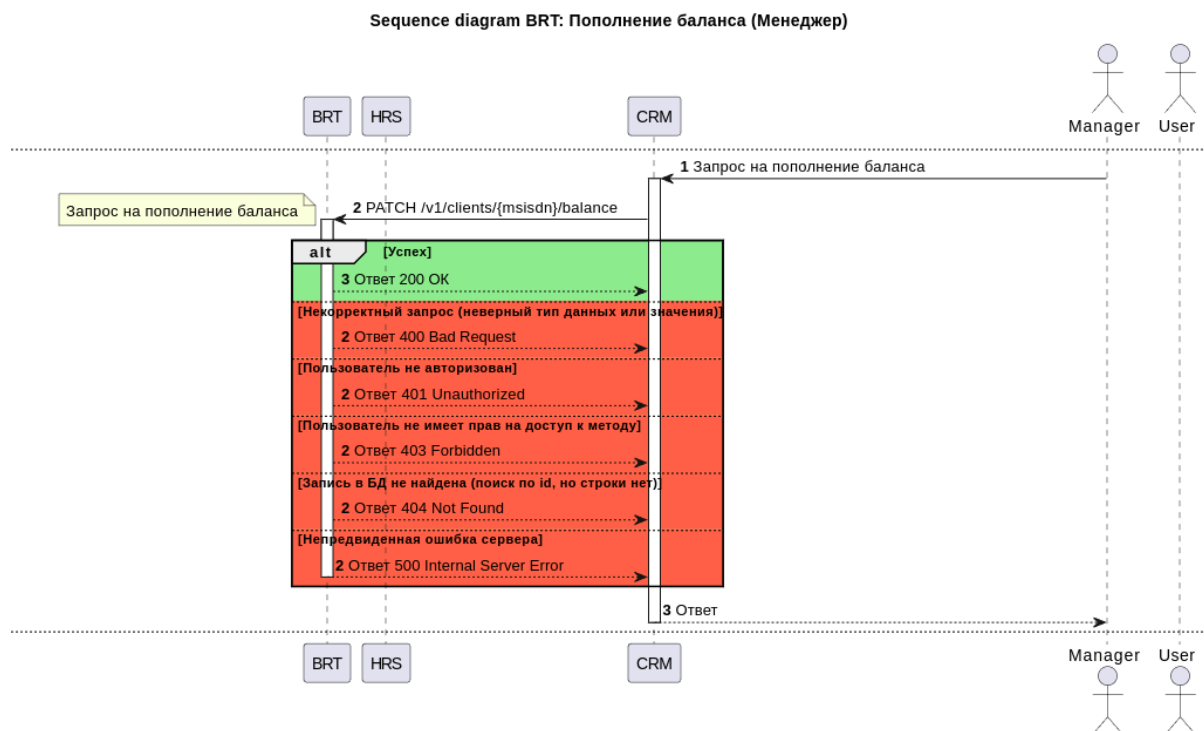


Рисунок 66 - Пополнение баланса менеджером

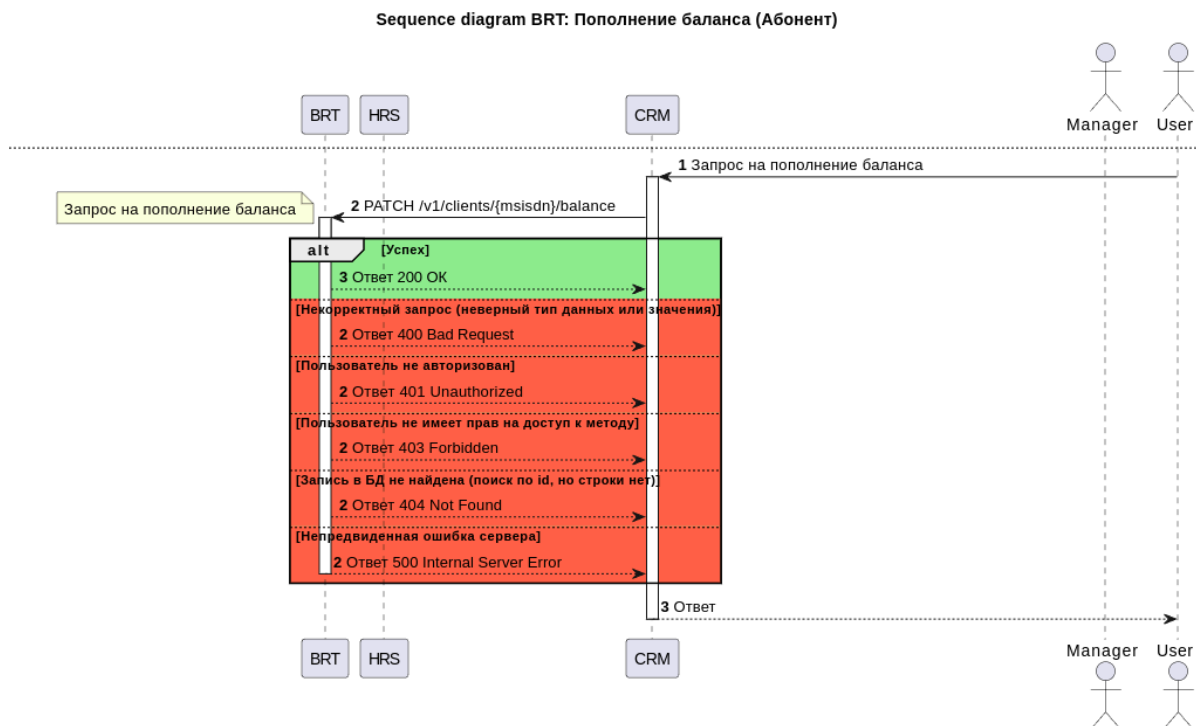


Рисунок 67 - Пополнение баланса абонентом

Activity Diagram

Результат построения activity diagram представлен на рисунках 68 - 72.

[Папка с кодом для PlantUml и изображениями](#)

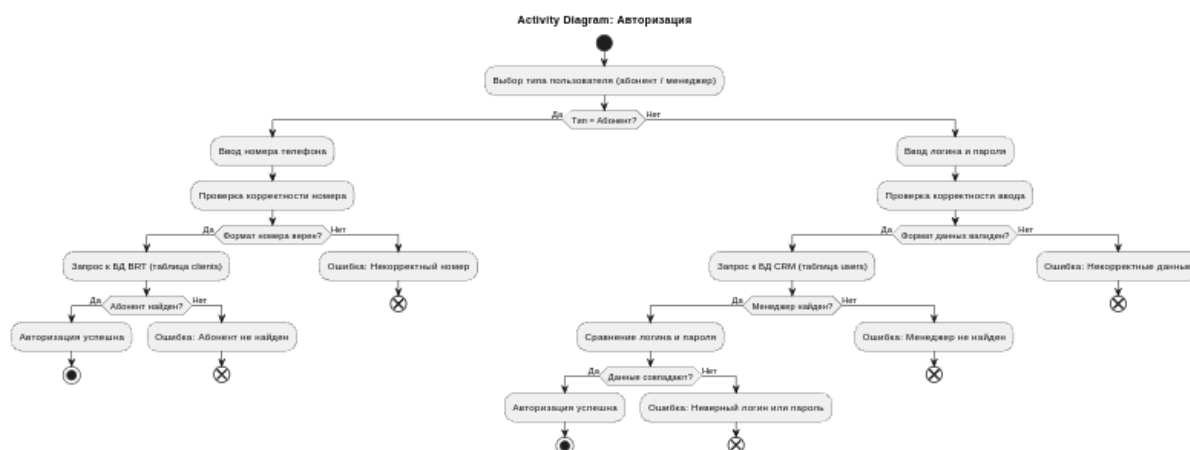


Рисунок 68 - Авторизация

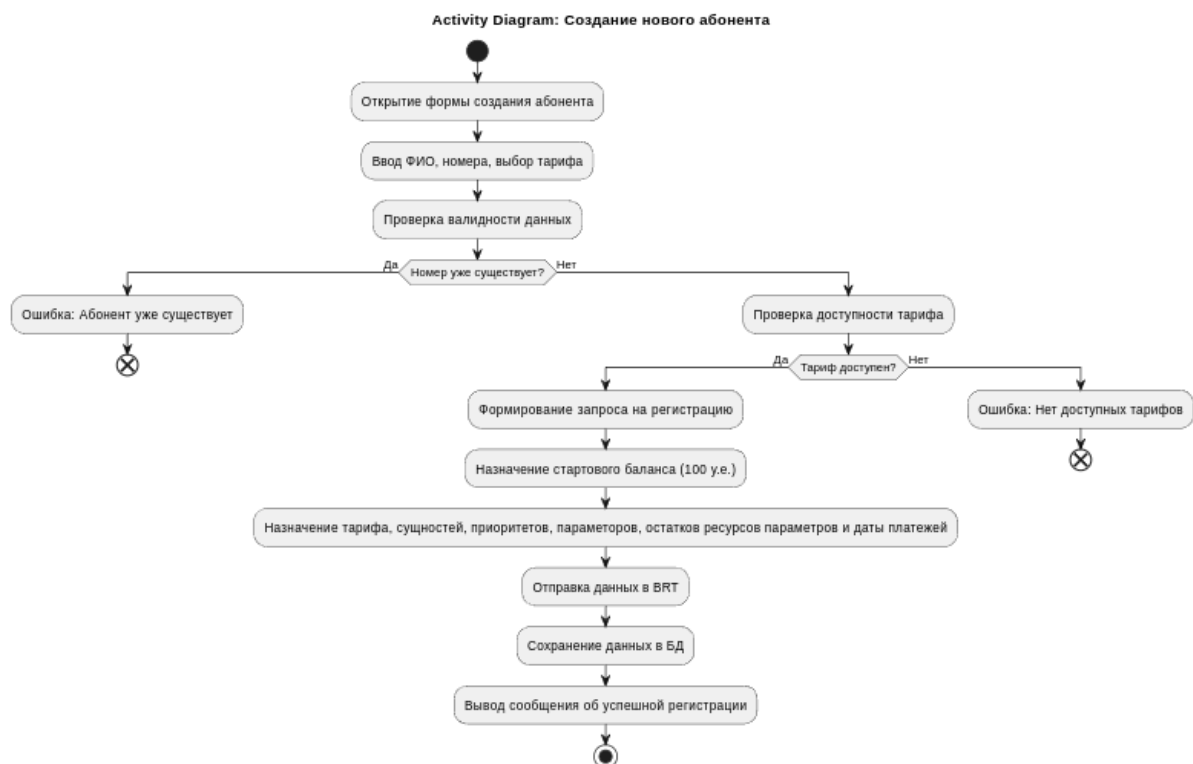


Рисунок 69 - Создание нового абонента



Рисунок 70 - Просмотр подробной информации об абоненте

Activity Diagram: Пополнение баланса

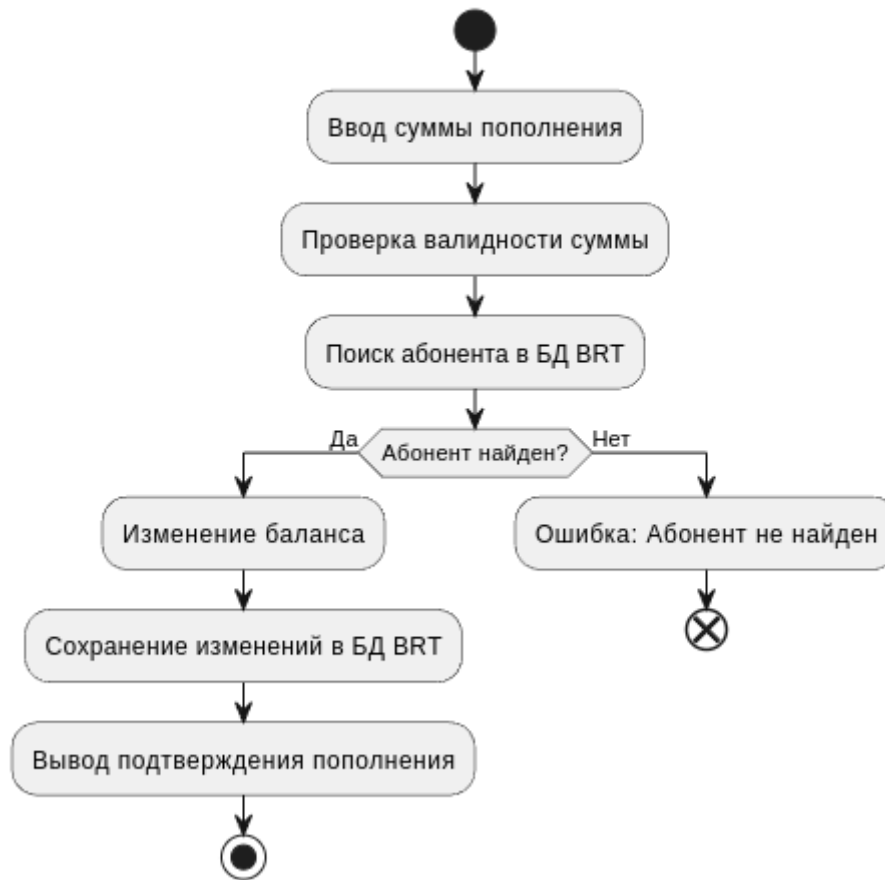


Рисунок 71 - Пополнение баланса

Activity Diagram: Смена тарифа абонента

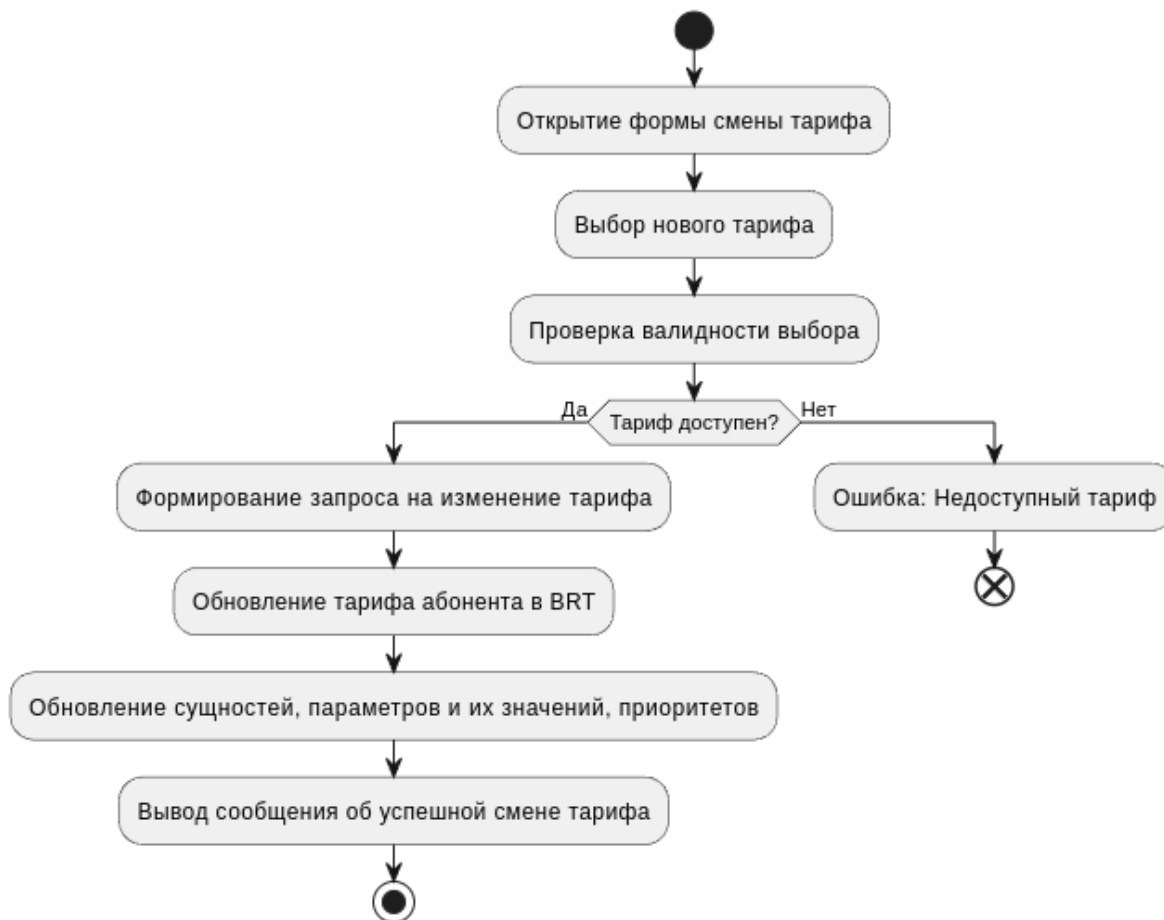


Рисунок 72 - Смена тарифа абонента

Swagger

Был разработан API по REST для взаимодействия менеджера и абонента с системой. Результат с краткой структурой эндпоинтов и методов представлен на рисунке 73.

Для более подробного рассмотрения параметров, содержимого запросов и возвращаемых кодах, предлагаю ознакомиться с yaml документом - [Файл swagger](#)

Client Management API 1.0.0 OAS 3.0

API для управления информацией клиентов

Servers

<http://localhost:8080/api> ▾

Абоненты Управление абонентами (создание, обновление, удаление)

GET	/v1/clients	Получить список всех абонентов	▾
POST	/v1/clients	Создать нового абонента	▾
GET	/v1/clients/{msisdn}	Получить абонента по MSISDN	▾
PUT	/v1/clients/{msisdn}	Обновить данные абонента	▾
DELETE	/v1/clients/{msisdn}	Удалить абонента	▾
GET	/v1/clients/{msisdn}/balance	Получить баланс абонента	▾
PATCH	/v1/clients/{msisdn}/balance	Изменить баланс абонента	▾
GET	/v1/clients/{msisdn}/tariff	Получить тариф абонента	▾
PATCH	/v1/clients/{msisdn}/tariff	Изменить тариф абонента	▾

Тарифы Просмотр тарифов

GET	/v1/tariffs/{tariffId}	Получить информацию о тарифе по ID	▾
GET	/v1/tariffs/{tariffName}	Получить информацию о тарифе по названию	▾

Рисунок 73 - API для сервиса CRM