**Практическая работа 3**

**Цель работы:** Научиться моделировать процессы и структуры проекта с использованием UML-диаграмм, а также закрепить понимание их применения на практике. Для выполнения задания необходимо разработать несколько UML-диаграмм для проекта на выбранную тему.

Тема: салон красоты

**Диаграмма использования (Use Case Diagram**)

Постройте диаграмму, которая отразит взаимодействие между пользователями системы (акторами) и ее функциональными возможностями (кейсы использования). Диаграмма должна включать:

- Основные роли пользователей.

- Основные функции системы.

- Взаимодействие акторов с кейсами использования.

Для каждого кейса использования кратко опишите его сценарий.

@startuml

left to right direction

actor "Гость" as Guest

actor "Пользователь" as User

actor "Администратор" as Admin

actor "Менеджер" as Manager

' Базовые функции для Гостя

Guest --> (Просмотр услуг)

Guest --> (Просмотр акций)

Guest --> (Просмотр контента)

Guest --> (Просмотр мастеров)

Guest --> (Задать вопрос)

Guest --> (Регистрация)

Guest --> (Авторизация)

(Регистрация) ..> (Ввод ФИО): <<include>>

(Регистрация) ..> (Ввод логина): <<include>>

(Регистрация) ..> (Ввод пароля): <<include>>

(Регистрация) ..> (Ввод телефона): <<include>>

' Функции Пользователя

User --> (Редактировать аккаунт)

User --> (Удалить аккаунт)

User --> (Просмотр аккаунта)

User --> (Оставить отзыв)

User --> (Купить сертификат)

User --> (Забронировать услугу)

(Купить сертификат) ..> (Выбор сертификата): <<include>>

(Забронировать услугу) ..> (Выбор услуги) : <<include>>

(Забронировать услугу) ..> (Выбор филиала) : <<include>>

(Забронировать услугу) ..> (Выбор мастера) : <<include>>

(Забронировать услугу) ..> (Выбор даты и времени) : <<include>>

' Функции Администратора

Admin --> (Создать пользователя)

Admin --> (Удалить пользователя)

Admin --> (Редактировать пользователя)

Admin --> (Просмотр пользователя)

Admin --> (Создать менеджера)

Admin --> (Удалить менеджера)

Admin --> (Редактировать менеджера)

Admin --> (Просмотр менеджера)

' Функции Менеджера

Manager --> (Управление контентом)

Manager --> (Управление акциями)

Manager --> (Управление мастерами)

Manager --> (Управление отзывами)

Manager --> (Управление вопросами)

Manager --> (Управление услугами)

Manager --> (Управление бронированиями)

Manager --> (Управление сертификатами)

' Обобщения для Менеджера

(Управление контентом) <|-- (Создать контент)

(Управление контентом) <|-- (Удалить контент)

(Управление контентом) <|-- (Редактировать контент)

(Управление контентом) <|-- (Просмотр контента)

(Управление акциями) <|-- (Создать акцию)

(Управление акциями) <|-- (Удалить акцию)

(Управление акциями) <|-- (Редактировать акцию)

(Управление акциями) <|-- (Просмотр акции)

(Управление мастерами) <|-- (Создать мастера)

(Управление мастерами) <|-- (Удалить мастера)

(Управление мастерами) <|-- (Редактировать мастера)

(Управление мастерами) <|-- (Просмотр мастера)

(Управление отзывами) <|-- (Создать отзыв)

(Управление отзывами) <|-- (Удалить отзыв)

(Управление отзывами) <|-- (Редактировать отзыв)

(Управление отзывами) <|-- (Просмотр отзывов)

(Управление вопросами) <|-- (Просмотр вопроса)

(Управление вопросами) <|-- (Ответить на вопрос)

(Управление услугами) <|-- (Создать услугу)

(Управление услугами) <|-- (Удалить услугу)

(Управление услугами) <|-- (Редактировать услугу)

(Управление услугами) <|-- (Просмотр услуг)

' НОВОЕ: Управление бронированиями для менеджера

(Управление бронированиями) <|-- (Просмотр всех броней)

(Управление бронированиями) <|-- (Подтвердить бронь)

(Управление бронированиями) <|-- (Отменить бронь)

(Управление бронированиями) <|-- (Перенести бронь)

' НОВОЕ: Управление сертификатами для менеджера

(Управление сертификатами) <|-- (Создать сертификат)

(Управление сертификатами) <|-- (Активировать сертификат)

(Управление сертификатами) <|-- (Деактивировать сертификат)

(Управление сертификатами) <|-- (Просмотр сертификатов)

' Связи между созданием и потреблением контента

(Создать контент) ..> (Просмотр контента) : <<include>>

(Создать акцию) ..> (Просмотр акций) : <<include>>

(Создать мастера) ..> (Просмотр мастеров) : <<include>>

(Оставить отзыв) ..> (Просмотр отзывов) : <<include>>

(Задать вопрос) ..> (Ответить на вопрос) : <<include>>

' Связи для услуг и бронирований

(Создать услугу) ..> (Просмотр услуг) : <<include>>

(Редактировать услугу) ..> (Просмотр услуг) : <<include>>

(Создать сертификат) ..> (Купить сертификат) : <<include>>

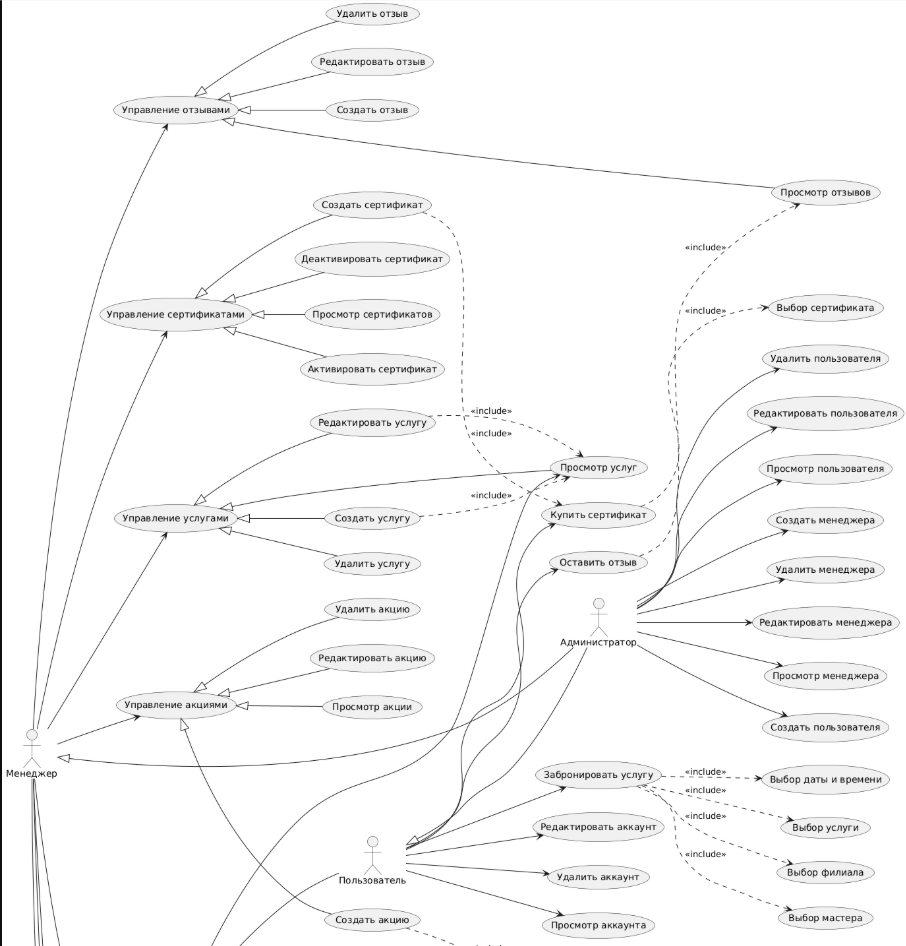
' Наследование ролей

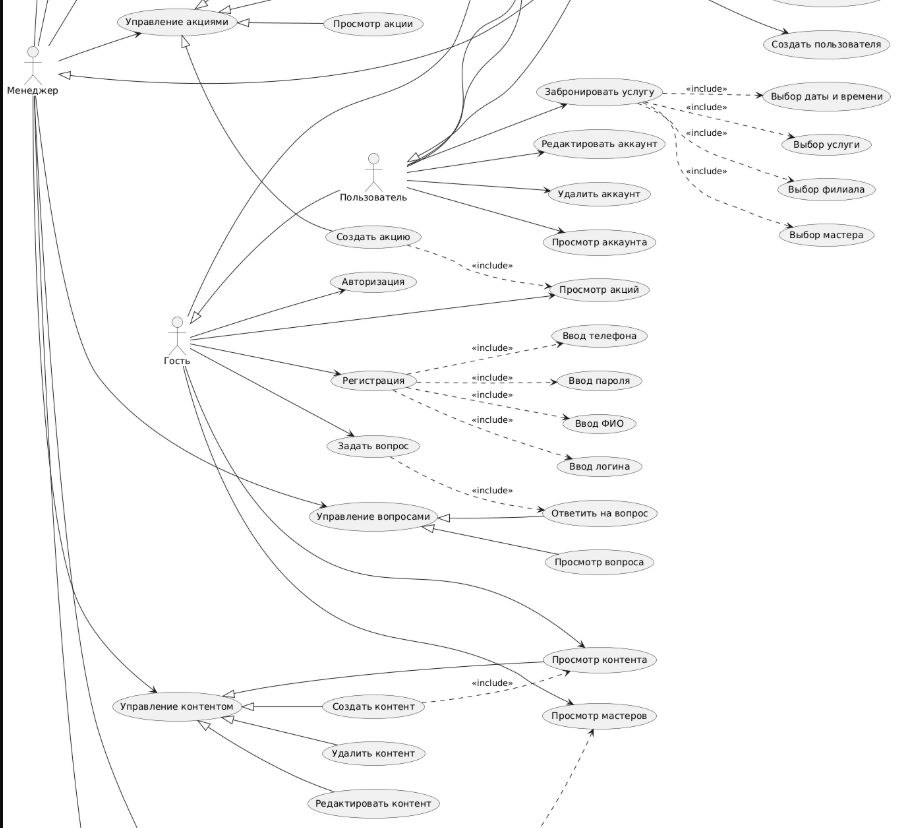
User -up-|> Guest

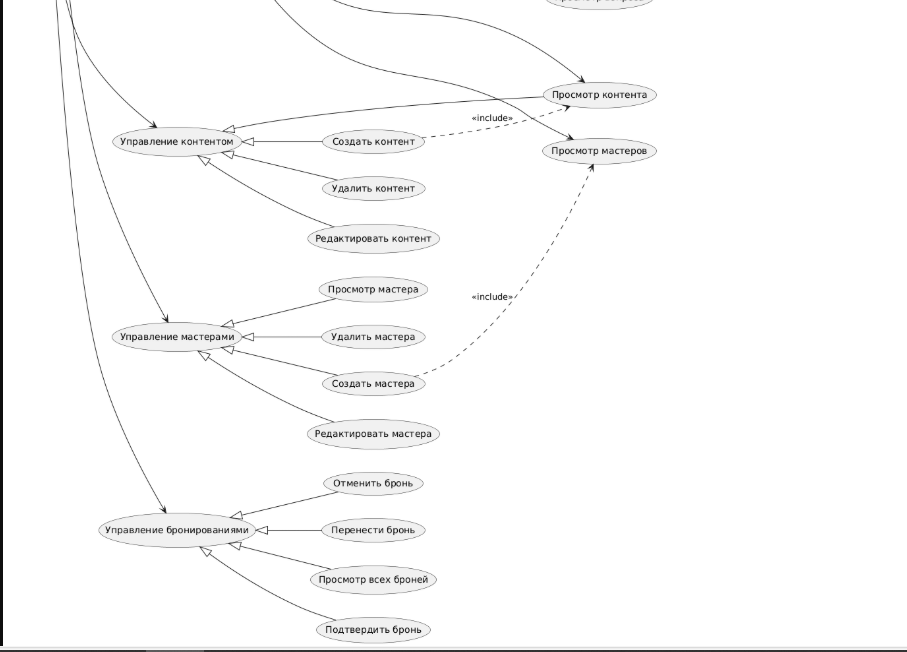
Admin -up-|> User

Admin -up-|> Manager

@enduml







### Диаграмма активностей (Activity diagram)

### Постройте диаграмму активностей (видов деятельности), отображающую динамические аспекты поведения системы.

@startuml

title Процесс бронирования услуги в салоне красоты

start

:Вход на сайт;

:Выбор услуги;

if (Выбранная услуга доступна?) then (да)

else (нет)

:Показать другие доступные услуги;

if (Клиент выбирает другую услугу?) then (да)

else (нет)

:Бронь отменена;

stop

endif

endif

:Выбор филиала;

:Выбор мастера;

:Выбор даты и времени;

if (Пользователь авторизован?) then (да)

:Автоматическая подстановка данных;

else (нет)

if (Пользователь зарегистрирован?) then (да)

:Авторизация;

else (нет)

:Регистрация;

endif

endif

:Подтверждение данных брони;

:Бронь создана;

fork

:Отправка уведомления клиенту;

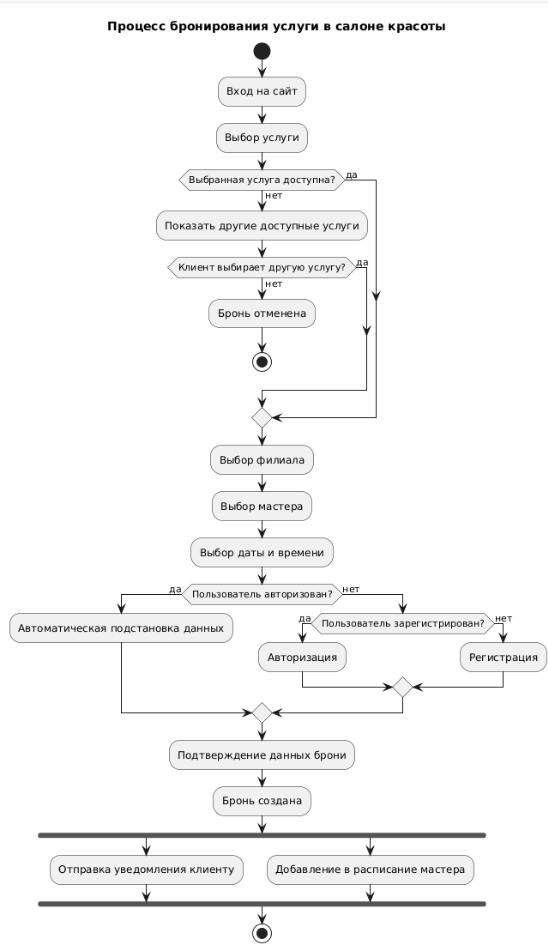
fork again

:Добавление в расписание мастера;

end fork

stop

@enduml



**Диаграмма последовательности (Sequence Diagram)**

Постройте диаграмму последовательности для одного или нескольких ключевых процессов в системе.

Отразите последовательность взаимодействия объектов внутри системы, показывая обмен сообщениями между ними.

*Диаграмма должна включать:*

- Акторов, участвующих в процессе.

- Основные объекты системы, участвующие в процессе.

- Последовательность сообщений и действий, которые происходят между объектами.

@startuml

title Процесс бронирования услуги в салоне красоты

actor Клиент

participant "Веб-интерфейс" as UI

participant "Контроллер\nбронирования" as Controller

participant "Сервис\nбронирования" as BookingService

participant "Сервис\nпользователей" as UserService

participant "Сервис\nрасписания" as ScheduleService

participant "База данных" as DB

participant "Сервис\nуведомлений" as NotificationService

participant "Личный кабинет" as PersonalAccount

participant "Мастер" as Master

== Инициализация бронирования ==

Клиент -> UI: Открывает форму бронирования

UI -> Controller: Запрос доступных услуг

Controller -> BookingService: Получить список услуг

BookingService -> DB: Запрос услуг

DB --> BookingService: Список услуг

BookingService --> Controller: Список услуг

Controller --> UI: Отобразить услуги

Клиент -> UI: Выбирает услугу

UI -> Controller: Запрос доступных филиалов для услуги

Controller -> BookingService: Получить филиалы(услуга)

BookingService -> DB: Запрос филиалов

DB --> BookingService: Список филиалов

BookingService --> Controller: Список филиалов

Controller --> UI: Отобразить филиалы

Клиент -> UI: Выбирает филиал

UI -> Controller: Запрос доступных мастеров\n(услуга, филиал)

Controller -> BookingService: Получить мастеров(услуга, филиал)

BookingService -> DB: Запрос мастеров

DB --> BookingService: Список мастеров

BookingService --> Controller: Список мастеров

Controller --> UI: Отобразить мастеров

Клиент -> UI: Выбирает мастера

UI -> Controller: Запрос свободного времени\n(мастер, услуга)

Controller -> ScheduleService: Получить свободные слоты(мастер)

ScheduleService -> DB: Запрос расписания мастера

DB --> ScheduleService: Расписание мастера

ScheduleService --> Controller: Свободные слоты

Controller --> UI: Отобразить свободное время

Клиент -> UI: Выбирает время из доступных

== Процесс бронирования ==

UI -> Controller: Подтверждение бронирования\n(услуга, филиал, мастер, время)

Controller -> UserService: Проверить авторизацию пользователя

alt Пользователь авторизован

UserService --> Controller: Пользователь авторизован

Controller -> BookingService: Создать бронирование\n(пользователь, услуга, мастер, время, филиал)

BookingService -> DB: Сохранить данные бронирования

DB --> BookingService: ID бронирования

BookingService -> ScheduleService: Зарезервировать слот\n(мастер, время, ID бронирования)

ScheduleService -> DB: Обновить расписание\n(мастер, время, ID бронирования)

DB --> ScheduleService: Успешно

BookingService --> Controller: Бронирование создано

Controller -> PersonalAccount: Добавить уведомление\n(пользователь, детали бронирования)

PersonalAccount -> DB: Сохранить уведомление

DB --> PersonalAccount: Успешно

Controller -> NotificationService: Уведомить мастера\n(мастер, детали бронирования)

NotificationService -> Master: Отправить уведомление\n(SMS/Email/Telegram)

Controller --> UI: Показать сообщение об успехе(ID бронирования)

UI -> Клиент: Показать подтверждение бронирования

else Пользователь не авторизован

UserService --> Controller: Пользователь не авторизован

Controller --> UI: Требуется авторизация

UI -> Клиент: Показать форму авторизации/регистрации

Клиент -> UI: Вводит учетные данные

UI -> Controller: Авторизация(учетные данные)

Controller -> UserService: Аутентифицировать(учетные данные)

UserService -> DB: Проверить пользователя(учетные данные)

DB --> UserService: Данные пользователя

UserService --> Controller: Успешная аутентификация

Note over Controller: Повторная попытка бронирования\nс авторизованным пользователем

Controller -> BookingService: Создать бронирование\n(пользователь, услуга, мастер, время, филиал)

BookingService -> DB: Сохранить данные бронирования

DB --> BookingService: ID бронирования

BookingService -> ScheduleService: Зарезервировать слот\n(мастер, время, ID бронирования)

ScheduleService -> DB: Обновить расписание\n(мастер, время, ID бронирования)

DB --> ScheduleService: Успешно

BookingService --> Controller: Бронирование создано

Controller -> PersonalAccount: Добавить уведомление\n(пользователь, детали бронирования)

PersonalAccount -> DB: Сохранить уведомление

DB --> PersonalAccount: Успешно

Controller -> NotificationService: Уведомить мастера\n(мастер, детали бронирования)

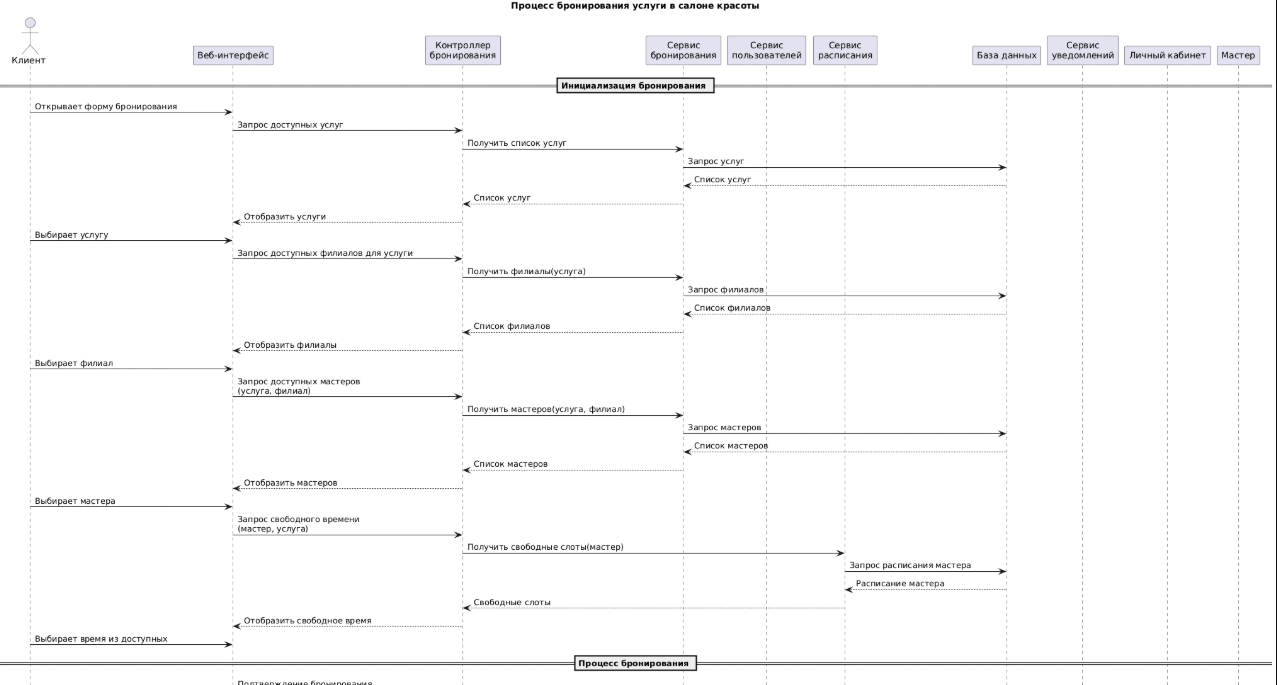
NotificationService -> Master: Отправить уведомление\n

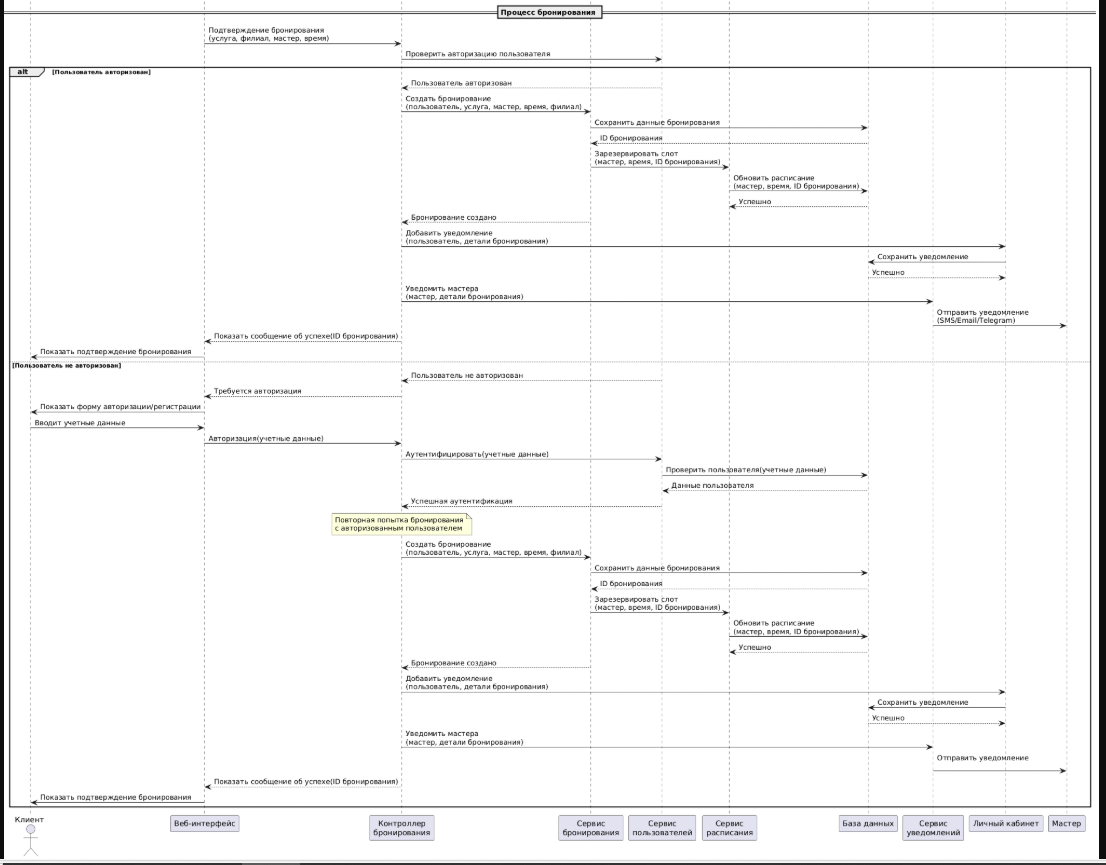
Controller --> UI: Показать сообщение об успехе(ID бронирования)

UI -> Клиент: Показать подтверждение бронирования

end

@enduml





**Создание диаграммма классов (Class Diagram)**

Постройте диаграмму классов для моделирования структуры проекта.

Определите основные классы системы, их атрибуты, методы и связи между ними.

*Диаграмма должна включать:*

- Классы с их атрибутами и методами.

- Взаимосвязи между классами.

- Определение типов отношений (один ко многим, многие ко многим и т.д.).

@startuml

class Клиент {

- id: Long

- имя: String

- телефон: String

- email: String

- пароль: String

- датаРегистрации: Date

+ зарегистрироваться(): Boolean

+ авторизоваться(): Boolean

+ создатьБронирование(слот: СлотРасписания, услуга: Услуга): Бронирование

+ получитьМоиБронирования(): List<Бронирование>

+ отменитьБронирование(бронирование: Бронирование): Boolean

+ изменитьПрофиль(): Boolean

}

class Бронирование {

- id: Long

- датаСоздания: DateTime

- статус: String

- сумма: Double

- комментарий: String

+ подтвердить(): Boolean

+ отменить(): Boolean

+ перенести(новыйСлот: СлотРасписания): Boolean

+ оплатить(): Boolean

}

class Услуга {

- id: Long

- название: String

- описание: String

- длительность: Integer

- базоваяЦена: Double

- категория: String

- активна: Boolean

+ доступнаВФилиале(филиал: Филиал): Boolean

+ получитьДоступныхМастеров(): List<Мастер>

+ рассчитатьЦену(мастер: Мастер): Double

}

class Мастер {

- id: Long

- имя: String

- специализация: String

- опыт: Integer

- рейтинг: Double

- телефон: String

- коэффициентЦены: Double

+ получитьСвободныеСлоты(дата: Date): List<СлотРасписания>

+ доступенВ(датаВремя: DateTime): Boolean

+ заблокироватьСлот(слот: СлотРасписания): Boolean

}

class Филиал {

- id: Long

- название: String

- адрес: String

- телефон: String

- графикРаботы: String

+ получитьМастеров(): List<Мастер>

+ получитьУслуги(): List<Услуга>

+ получитьСвободныеСлоты(дата: Date): List<СлотРасписания>

}

class Расписание {

- id: Long

- дата: Date

- мастер: Мастер

- филиал: Филиал

+ создатьСлотыНаДень(): List<СлотРасписания>

+ получитьСвободныеСлоты(): List<СлотРасписания>

+ обновитьСтатусСлота(слот: СлотРасписания): Boolean

}

class СлотРасписания {

- id: Long

- времяНачала: DateTime

- времяОкончания: DateTime

- статус: String

+ доступенДляБронирования(): Boolean

+ забронировать(бронирование: Бронирование): Boolean

+ освободить(): Boolean

+ заблокировать(): Boolean

}

class Уведомление {

- id: Long

- тип: String

- содержание: String

- датаОтправки: DateTime

- статус: String

+ отправитьКлиенту(клиент: Клиент, бронирование: Бронирование): Boolean

+ отправитьМастеру(мастер: Мастер, бронирование: Бронирование): Boolean

}

' Связи между классами

Клиент "1" -- "\*" Бронирование : создает

Бронирование "1" -- "1" Услуга : включает

Бронирование "1" -- "1" Мастер : назначается

Бронирование "1" -- "1" Филиал : выполняется\_в

Бронирование "1" -- "1" СлотРасписания : занимает

Мастер "1" -- "\*" Расписание : имеет

Расписание "1" -- "\*" СлотРасписания : содержит

Филиал "1" -- "\*" Мастер : работает\_в

Филиал "1" -- "\*" Услуга : предлагает

Мастер "\*" -- "\*" Услуга : предоставляет

Уведомление "1" -- "1" Клиент : отправляется\_клиенту

Уведомление "1" -- "1" Мастер : отправляется\_мастеру

Бронирование "1" -- "\*" Уведомление : генерирует

' Агрегации и композиции

Филиал \*-- Мастер : содержит

Филиал \*-- Услуга : предоставляет

Мастер \*-- Расписание : управляет

Расписание \*-- СлотРасписания : состоит\_из

@enduml

