

Министерство образования и науки Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО
ОБРАЗОВАНИЯ
“САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ,
МЕХАНИКИ И ОПТИКИ”
ФАКУЛЬТЕТ СРЕДНЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ № 2
«Построение диаграммы вариантов использования, диаграммы
классов»

Специальность 11.03.02 «Инфокоммуникационные технологии и системы связи»

Преподаватель: Коцюба И.Ю. «12» ноября 2025г. Оценка:	Выполнил: студент группы К3221 Доценников Н.А.
---	---

Санкт-Петербург
2025/2026

Цель работы.

Освоить построение диаграммы вариантов использования (use case) и диаграммы классов uml для заданной предметной области, закрепить навыки анализа требований и проектирования архитектуры предметной области.

Задачи.

Разработать диаграмму вариантов использования и диаграмму классов для информационной системы регистрации пациента в больнице. Подготовить пояснительный отчёт с текстовыми описаниями и обоснованием связей.

Индивидуальная тема.

Была выбрана предметная область «Информационная система регистрации пациента в больнице».

Данная тема включает процессы:

- регистрация и поиск пациентов;
- запись на приём и госпитализация;
- ведение медицинской карты (диагнозы, назначения, процедуры, рецепты);
- выписка пациента;
- финансовый учёт (выставление счетов, расчёт стоимости услуг, приём оплат);
- управленческий контроль (отчёты заведующего отделением, администрирование пользователей).

Ход работы.

1. Ознакомился с индивидуальной темой: «ИС регистрации пациента в больнице».
2. Выявил основных акторов системы: пациент, регистратор, врач, заведующий отделением, бухгалтерия, администратор системы.
3. Определил перечень основных вариантов использования: регистрация и поиск пациента, запись на приём, госпитализация, ведение медкарты, назначение лечения, выписка, выставление счета, приём оплат, просмотр отчётов, управление пользователями.
4. Составил диаграмму вариантов использования (use case), обозначив акторов, связи «включает» и «расширяет».
5. Определил основные сущности предметной области: пользователь и его роли, пациент, отделение, госпитализация, приём, медкарта, диагноз, план лечения, назначения, счёт, оплата, отчёт.
6. Построил диаграмму классов uml, указав атрибуты, методы и связи с кратностями.
7. Выполнил трассировку вариантов использования на классы, показав соответствие сценариев сущностям системы.
8. Подготовил текстовый отчёт, включающий цель, задание, глоссарий, диаграммы и выводы.

Диаграмма вариантов использования (use Case)

(отправлена отдельным файлом)

Диаграмма вариантов использования отражает функциональные возможности системы глазами её пользователей. Она показывает, кто взаимодействует с системой и какие функции для них доступны.

Акторы системы:

- Пациент — может записываться на приём, оплачивать услуги.

- Регистратор — выполняет регистрацию пациентов, поиск и редактирование данных, запись на приём, госпитализацию.
- Врач — назначает лечение, делает записи в медкарте, выписывает рецепты и оформляет выписку пациента.
- Заведующий отделением — просматривает и формирует отчёты по работе отделения.
- Бухгалтерия — выставляет счета и принимает оплаты.
- Администратор системы — управляет пользователями и их правами доступа.

Основные варианты использования:

- Авторизация пользователя
- Регистрация пациента
- Поиск пациента
- Редактирование данных пациента
- Запись на приём
- Госпитализация пациента
- Назначить лечение
- Выписка пациента
- Выставление счёта
- Принятие оплаты
- Просмотр и формирование отчётов
- Управление пользователями

Связи включает/расширяет:

- «Регистрация пациента» включает «Поиск пациента»

- «Назначить лечение» включает «Создать запись в медкарте» и «Выписать рецепт»
- «Выставить счёт» включает «Рассчитать стоимость услуг»
- «Просмотр отчётов» включает «Сформировать отчёт»
- «Госпитализация пациента» расширяет «Регистрация пациента»
- «Выписка пациента» расширяет «Назначить лечение»

flowchart tb

actor_P["Пациент"]

actor_R["Регистратор"]

actor_D["Врач"]

actor_H["Заведующий отделением"]

actor_F["Бухгалтерия"]

actor_A["Администратор системы"]

subgraph System[ИС регистрации пациента]

UC_Login["Авторизация пользователя"]

UC_Register["Регистрация пациента"]

UC_Find["Поиск пациента"]

UC_Edit["Редактирование данных пациента"]

UC_Admission["Госпитализация пациента"]

UC_Appointment["Запись на приём"]

UC_Treatment["Назначить лечение"]

UC_MedRec["Создать запись в медкарте"]

UC_Prescription["Выписать рецепт/назначения"]
UC_Discharge["Выписка пациента"]
UC_Invoice["Выставить счёт"]
UC_Calc["Рассчитать стоимость услуг"]
UC_Payment["Принять оплату"]
UC_Report["Просмотр отчётов"]
UC_ReportGen["Сформировать отчёт"]
UC_UserAdmin["Управление пользователями"]
end

actor_R --- UC_Login

actor_D --- UC_Login

actor_H --- UC_Login

actor_F --- UC_Login

actor_A --- UC_Login

actor_R --- UC_Register

actor_R --- UC_Find

actor_R --- UC_Edit

actor_R --- UC_Admission

actor_R --- UC_Appointment

actor_D --- UC_Treatment

actor_D --- UC_MedRec

actor_D --- UC_Prescription

actor_D --- UC_Discharge

actor_F --- UC_Invoice

actor_F --- UC_Payment

actor_H --- UC_Report

actor_A --- UC_UserAdmin

UC_Register -. включает .-> UC_Find

UC_Treatment -. включает .-> UC_MedRec

UC_Treatment -. включает .-> UC_Prescription

UC_Invoice -. включает .-> UC_Calc

UC_Report -. включает .-> UC_ReportGen

UC_Admission -. расширяет .-> UC_Register

UC_Discharge -. расширяет .-> UC_Treatment

actor_P --- UC_Appointment

actor_P --- UC_Payment

Диаграмма классов uml
(отправлена отдельным файлом)

Диаграмма классов отражает статическую структуру системы: основные сущности предметной области, их атрибуты и взаимосвязи. Она показывает, как реализованы варианты использования на уровне проектирования.

Классы системы:

- User — базовый класс пользователей системы, содержит учётные данные.
 - Специализации: Registrar, Doctor, HeadOfDepartment, Admin.
- Patient — содержит данные пациента.
- Department — отделение больницы.
- Admission — факт госпитализации пациента.
- Appointment — запись на приём.
- MedicalRecord — медицинская карта пациента.
- Diagnosis — диагноз с кодом и описанием.
- TreatmentPlan — план лечения.
- Prescription — назначение лекарства.
- Procedure — назначенная медицинская процедура.
- Invoice — счёт на оказанные услуги.
- InvoiceItem — отдельная позиция счёта.
- Payment — платеж.
- Report — отчёт за определённый период.

Основные связи и кратности:

- Пациент имеет одну медицинскую карту и может иметь много приёмов и госпитализаций.
- Приём связан с врачом и может содержать диагноз.
- Медкарта включает диагнозы и планы лечения; план лечения содержит назначения.
- Госпитализация создаётся регистратором, привязана к пациенту и отделению; может включать план лечения.
- Счёт связан с пациентом и госпитализацией, состоит из позиций; к счёту относятся оплаты.
- Заведующий отделением формирует отчёты по отделению.

Роль диаграммы:

Диаграмма классов обеспечивает переход от анализа требований (use case) к проектированию структуры системы. Она фиксирует объекты предметной области и их взаимодействия, что является основой для реализации базы данных и бизнес-логики.

```
classDiagram
```

```
class User {  
    +id: UUID  
    +login: string  
    +passwordHash: string  
    +fullName: string  
    +role: Role  
}
```

```
class Registrar

class Doctor {
  +specialty: string
}

class HeadOfDepartment

class Admin

User <|-- Registrar
User <|-- Doctor
User <|-- HeadOfDepartment
User <|-- Admin
```

```
class Patient {
  +id: UUID
  +fullName: string
  +birthDate: date
  +sex: string
  +phone: string
  +policyNumber: string
}
```

```
class Department {
  +id: UUID
  +name: string
```

}

```
class Admission {  
    +id: UUID  
    +admitDate: datetime  
    +dischargeDate: datetime  
    +status: AdmissionStatus  
}
```

```
class Appointment {  
    +id: UUID  
    +dateTime: datetime  
    +status: ApptStatus  
}
```

```
class MedicalRecord {  
    +id: UUID  
    +createdAt: datetime  
}
```

```
class Diagnosis {  
    +code: string  
    +name: string
```

```
+notes: string
}

class TreatmentPlan {
  +id: UUID
  +createdAt: datetime
  +notes: string
}
```

```
class Prescription {
  +id: UUID
  +drugName: string
  +dosage: string
  +schedule: string
}
```

```
class Procedure {
  +id: UUID
  +name: string
  +dateTime: datetime
  +result: string
}
```

```
class Invoice {  
  
    +id: UUID  
  
    +issueDate: date  
  
    +total: money  
  
    +status: InvoiceStatus  
  
}
```

```
class InvoiceItem {  
  
    +name: string  
  
    +qty: int  
  
    +price: money  
  
    +amount(): money  
  
}
```

```
class Payment {  
  
    +id: UUID  
  
    +dateTime: datetime  
  
    +amount: money  
  
    +method: PayMethod  
  
}
```

```
class Report {  
  
    +id: UUID
```

```
+title: string
+periodStart: date
+periodEnd: date
+generatedAt: datetime
}
```

```
Patient "1" --> "0..*" Appointment
Doctor "1" --> "0..*" Appointment
Appointment "*" --> "0..1" Diagnosis
```

```
Patient "1" --> "1" MedicalRecord
MedicalRecord "1" --> "0..*" Diagnosis
MedicalRecord "1" --> "0..*" TreatmentPlan
TreatmentPlan "1" --> "0..*" Prescription
TreatmentPlan "1" --> "0..*" Procedure
```

```
Registrar "1" --> "0..*" Admission
Patient "1" --> "0..*" Admission
Department "1" --> "0..*" Admission
Admission "1" --> "0..1" TreatmentPlan
```

```
Invoice "1" --> "1..*" InvoiceItem
```

Patient "1" --> "0..*" Invoice

Admission "0..1" --> "0..*" Invoice

Invoice "1" --> "0..*" Payment

HeadOfDepartment "1" --> "0..*" Report

Department "1" --> "0..*" Report

Вывод.

В ходе выполнения лабораторной работы были рассмотрены основы моделирования информационных систем с использованием uml.

- Определены акторы и построена диаграмма вариантов использования (use case), отражающая взаимодействие пользователей с системой и её функциональные возможности.
- Построена диаграмма классов uml, описывающая статическую структуру системы, основные сущности предметной области и взаимосвязи между ними.
- Выполнено сопоставление вариантов использования и классов, что позволило показать, каким образом функциональные требования реализуются на уровне проектирования.

