|  |
| --- |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** |
| Институт кибербезопасности и цифровых технологий |
| Кафедра КБ-2 «Прикладные информационные технологии» |

**ПРАКТИЧЕСКАЯ РАБОТА №2 по дисциплине**

**«Технологии и методы программирования»**

***(наименование дисциплины)***

**Тема практической работы   
«Модельно-ориентированный подход к проектированию»**

Студент группы: БИСО-01-20 Митрофанов М.С.

*(шифр учебной группы) (Фамилия И.О)*

Преподаватель: Лесько С.А.

*(Фамилия И.О)*

Москва, 2022

**Задание**

По варианту (мой вариант – 3) из Практической работы №1 UML реализовать диаграммы последовательностей и развёртывания.

**Диаграмма последовательности**

Описание

Сначала пациент регистрируется в системе путем отправки запроса на регистрацию в клинику через интерфейс пациента. Затем информация о пациенте передается регистратору, который создает запись о пациенте в системе и создает медицинскую карту.

После этого пациент записывается на прием к врачу через интерфейс пациента, и информация о записи передается регистратору. Регистратор создает запись о приеме и связывает ее с медицинской картой.

Затем врач заполняет медицинскую карту пациента, добавляя диагнозы и рецепты. После этого врач отмечает факт приема в системе, а медсестра помогает при осмотре пациента.

На диаграмме также показаны связи между участниками: пациент взаимодействует с клинической системой и регистратором, врачи заполняют медицинские карты, регистратор управляет записями на прием и медицинскими картами, а клиническая система содержит всех участников и регистратора.

Данный код UML создает диаграмму последовательности, которая описывает основные прецеденты информационной системы поликлиники:

@startuml

actor Patient

actor Doctor

actor Nurse

participant Registrar

participant ClinicSystem

participant MedicalRecord

Patient -> ClinicSystem: регистрация пациента

ClinicSystem -> Registrar: регистрация пациента

Registrar -> Patient: создание записи о пациенте

Patient -> ClinicSystem: запись на прием

ClinicSystem -> Registrar: запись на прием

Registrar -> MedicalRecord: создание медицинской карты

Doctor -> MedicalRecord: заполнение медицинской карты

Doctor -> MedicalRecord: изменение медицинской карты

Doctor -> Patient: осмотр пациента

Nurse -> Patient: помощь при осмотре

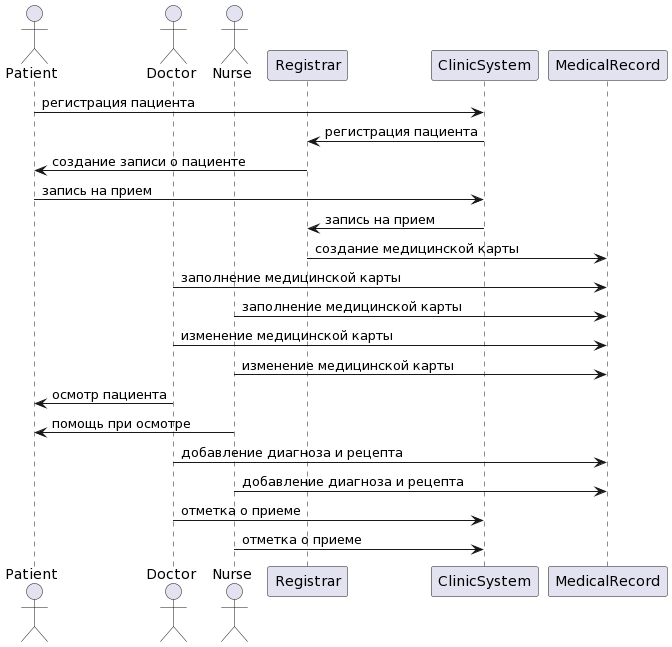
Doctor -> MedicalRecord: добавление диагноза и рецепта

Doctor -> ClinicSystem: отметка о приеме

Nurse -> ClinicSystem: отметка о приеме

@enduml

Диаграмма последовательности:



**Диаграмма развёртывания**

Описание

На диаграмме развертывания изображены основные компоненты компьютерной системы, такие как веб-сервер, сервер приложений, сервер баз данных и несколько клиентских рабочих станций.

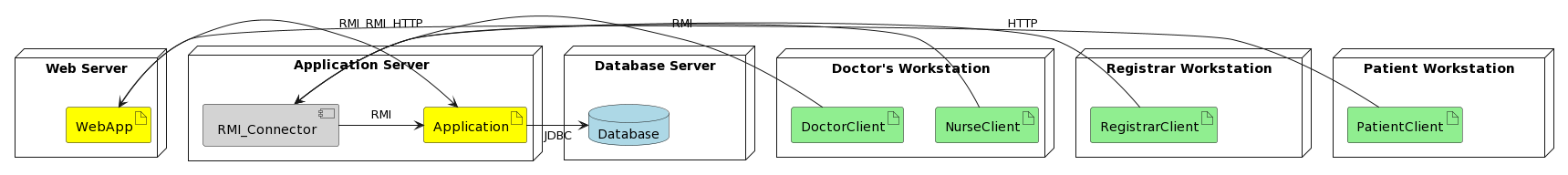
Отношения между компонентами: веб-приложение общается с сервером приложений через протокол HTTP, а сервер приложений использует JDBC для взаимодействия с базой данных. Для связи клиентов (доктора, медсестры, регистратора) с сервером приложений используется RMI (Remote Method Invocation).

Клиент пациента (браузер) отправляет запросы на веб-сервер, который в свою очередь взаимодействует с сервером приложений для обработки запроса и получения данных из базы данных. После этого, сервер приложений возвращает ответ на веб-сервер, который передает его обратно клиенту.

Таким образом, клиент пациента в данной диаграмме связан с WebApp, поскольку это веб-приложение, которое обрабатывает запросы клиентов через протокол HTTP.

Таким образом, данная диаграмма представляет собой типичную компонентную архитектуру клиент-серверного приложения.

Диаграмма развёртывания:



Исходный код UML:

@startuml

node "Web Server" {

artifact WebApp #yellow

}

node "Application Server" {

artifact Application #yellow

component RMI\_Connector #lightgray

}

node "Database Server" {

database Database #LightBlue

}

node "Doctor's Workstation" {

artifact DoctorClient #lightgreen

artifact NurseClient #lightgreen

}

node "Registrar Workstation" {

artifact RegistrarClient #lightgreen

}

node "Patient Workstation" {

artifact PatientClient #lightgreen

}

WebApp -> Application : HTTP

Application -> Database : JDBC

NurseClient -> RMI\_Connector : RMI

DoctorClient -> RMI\_Connector : RMI

RegistrarClient -> RMI\_Connector : RMI

RMI\_Connector -> Application : RMI

PatientClient -> WebApp : HTTP

@enduml