«Санкт-Петербургский государственный электротехнический университет «ЛЭТИ» им. В.И.Ульянова (Ленина)»
СПбГЭТУ «ЛЭТИ»

Кафедра вычислительной техники

Пояснительная записка к курсовой работе по дисциплине «Объектно-ориентированное программирование» Создание программного комплекса для поликлиники средствами объектно-ориентированного программирования

Студент гр.8306 _____ Слепов А.Э.

Преподаватель _____ Павловский М.Г.

Санкт-Петербург

2020

1.Техническое задание

1.1 Введение

Программный комплекс (ПК) администрирования поликлиники предназначен для использования в составе системы программно-информационного обеспечения учета и администрирования врачей и пациентов поликлиники.

1.2 Основание для разработки

Основанием для разработки ПК «Регистрация, учет, администрирование, редактирования и выдача сведений о врачах и пациентах» является курсовой проект по дисциплине «Объектно-ориентированное программирование».

1.3 Назначение разработки

ПК «Регистрация, учет, администрирование, редактирование и выдача сведений о врачах и пациентах» должен входить в состав автоматизированной системы учета и администрирования информации, и предназначен для автоматизации деятельности лица (ОЛ), ответственного за администрирование поликлиники.

ПК «Регистрация, учет, администрирование, редактирование и выдача сведений о врачах и пациентах» предназначен для автоматизации следующих процессов:

- учет и администрирование информации о врачах и пациентах поликлиники;
- запись пациентов на прием к врачу;
- получение справочной информации о занятости врачей;

1.4 Требования к программе

1.4.1 Требования к функциональным характеристикам

1.4.1.1 Перечень функций

ПК «Регистрация, учет, администрирование, редактирование и выдача сведений о врачах и пациентах» должен обеспечивать выполнение следующих функций:

- просмотр, добавление, удаление и изменение базы данных (БД);
- выдача справочной информации, хранимой в БД, по запросам ОЛ.

1.4.1.2 Требования к составу выполняемых функций

*1.4.1.2.1 Функция «*просмотр, добавление, удаление и изменение в базы данных (БД)*»*

Ввод, просмотр, добавление, удаление и изменение в БД должны обеспечивать ведение и хранение следующих данных:

- данные о врачах;
- данных о пациентах;
- данные о записи пациентов к врачу.

1.4.1.2 Требования к организации и форме представления выходных данных

Выходные данные должны быть представлены в виде таблицы содержащий описание необходимых информационных объектов, выполненного посредствам представления его характеристик.

1.4.1.3 Требования к организации и форме представления входных данных

Входная информация для задачи «Регистрация, учет, администрирование, редактирование и выдача сведений о врачах и пациентах» содержится в приходно-расходной документации. Ввод исходных данных должен осуществляется ОЛ в режиме диалога. Вводимые данные являются значениями характеристик (атрибутов) информационных объектов.

1.4.2 Требования к надежности

ПК «Регистрация, учет, администрирование, редактирование и выдача сведений о врачах и пациентах» должен устойчиво функционировать при соблюдении гарантии устойчивого функционирования операционной системы и системы управления базой данных. Под устойчивой работой ПК понимается непрерывное функционирование программы в отсутствии критических сбоев, приводящих к аварийному завершению. Кроме того, должен быть обеспечен контроль входных данных на предмет соответствия предполагаемому типу.

1.4.3 Условия эксплуатации

Выполнение ПК «Регистрация, учет, администрирование, редактирование и выдача сведений о врачах и пациентах» своих функций должно быть обеспечено для однопользовательского режима работы с монопольным доступом к базе данных.

1.4.4 Требование к составу и параметрам технических средств

Задача должна решаться на ПЭВМ типа IBM PC или совместимой с ней с процессором Pentium III 500 и выше, ОЗУ не менее 128Мб, HDD не менее 4 Гб, монитор SVGA (цветной) 15", видеокарта 64 Мб, клавиатура 102 кл., манипулятор типа "мышь".

1.4.5 Требование к информационной и программной совместимости

Выходная и входная информация ПК «Регистрация, учет, администрирование, редактирование и выдача сведений о врачах и пациентах» должна быть удобна для визуального восприятия. ПК должен быть выполнен на языке программирования высокого уровня Java и должен быть совместим с операционной системой Windows.

Обязательными требованиями при разработке кода ПК являются использование следующих конструкций языка Java:

- закрытые и открытые члены классов;
- наследование;

- конструкторы с параметрами;
- абстрактные базовые классы;
- виртуальные функции;
- обработка исключительных ситуаций;
- динамическое создание объектов.

1.5 Требования к программной документации

Программная документация (ПД) должна удовлетворять требованиям стандартов ЕСПД.

Документация должна быть представлена в следующем составе:

1. описание процесса проектирования ПК;
2. руководство оператора;
3. исходные тексты ПК.

1.6 Стадии и этапы разработки

1. Разработка технического задания;
2. Описание вариантов использования ПК;
3. Создание прототипа интерфейса пользователя;
4. Разработка объектной модели ПК;
5. Построение диаграмм программных классов;
6. Описание поведения ПК;
7. Построение диаграмм действий;

1.7 Порядок контроля и приемки

В процессе приема работы устанавливается соответствие ПК и прилагаемой документации требованиям, обозначенным в техническом задании.

2 Проектирование ПК

2.1 Описание вариантов использования ПК

Развернутое описание функциональных требований осуществляется на этапе проектирования комплекса. Для того чтобы детализировать требования, необходимо выделить процессы, происходящие в заданной предметной области. Описание таких процессов на UML выполняется в виде прецедентов (use case). Прецеденты являются сценарием или вариантом использования ПК при взаимодействии с внешней средой. Они являются продолжением описаний требований и функциональных спецификаций, указанных в техническом задании. Прецедент изображается в виде эллипса, в котором содержится имя прецедента. Название прецедента обязательно включает в себя глагол, выражающий суть выполняемой функции. С помощью прецедентов описывается функционирование ПК с точки зрения внешнего пользователя, который называется в UML актором (actor). Актор представляет собой любую внешнюю по отношению к моделируемой системе сущность (человек, программная система, устройство), которая взаимодействует с системой и использует ее функциональные возможности для достижения определенных целей или решения частных задач. Актор на диаграмме изображается пиктограммой в виде человечка, под которым указано его имя. Совокупность функций, реализуемых ПК, изображается в виде диаграммы (use case diagram). Для построения диаграммы необходимо определить акторы, прецеденты (функции) и взаимоотношение между акторами и прецедентами, и между прецедентами, если один прецедент расширяет или использует другой. В языке UML для вариантов использования и действующих лиц поддерживается несколько типов связей. Это связи коммуникации (communication), использования (uses) и расширения (extends).

Связь коммуникации — это связь между прецедентом и актором. На языке UML связь коммуникации изображают в виде стрелки. Направление стрелки показывает, кто инициирует коммуникацию. При задании коммуникации необходимо указать данные, которые вводит или получает пользователь. Кроме

данных на концах стрелки можно указать кратности отношения, которые характеризуют количество взаимодействующих между собой акторов и прецедентов. На диаграммах прецедентов наиболее распространенными являются две формы записи кратности 1 и 1 .. *. Первая форма записи означает, что один актор (прецедент) участвует во взаимодействии, а вторая форма записи, что один или несколько акторов (прецедентов) участвуют во взаимодействии.

Связь использования предполагает, что один прецедент всегда применяет функциональные возможности другого. С помощью таких связей структурируют прецеденты, показывая тем самым, какой прецедент является составной частью другого прецедента. Такой включаемый прецедент является абстрактным прецедентом в том смысле, что он не может исполняться независимо от других прецедентов, а лишь в их составе. Связь использования изображается с помощью стрелок и слова «uses» (использование). Направление стрелки указывает, какой прецедент используется для реализации функциональности другого прецедента.

Связь расширения задается в том случае, если необходимо показать родственные отношения между двумя прецедентами. Один из них является базовым, а другой его расширением. Базовый прецедент не зависит от расширяющих прецедентов и способен функционировать без них. С другой стороны, расширяющие прецеденты без базового прецедента функционировать не могут. Связи расширения изображают в виде стрелки со словом «extends» (расширение), которая имеет направление от базового прецедента к расширяемому.

Прецеденты необходимо ранжировать, чтобы в начальных циклах разработки реализовать наиболее приоритетные из них. Разбиение функциональности системы на отдельные прецеденты служит примерно той же цели, что и разбиение сложного алгоритма на подпрограммы. Основная стратегия должна заключаться в том, чтобы сначала сконцентрировать внимание на тех прецедентах, которые в значительной мере определяют базовую архитектуру ПК.

Диаграмма прецедентов представлена на рис. 2.1.

Рисунок 2.1 – Диаграмма прецедентов

2.2 Создание прототипа интерфейса пользователя

Описание прецедента выражает общую сущность процесса без детализации его реализации. Проектные решения, связанные с интерфейсом пользователя, при этом опускаются. Для разработки пользовательского интерфейса необходимо описать процесс в терминах реальных проектных решений, на основе конкретных технологий ввода-вывода информации. Когда речь идет об интерфейсе пользователя, прецеденты разбиваются на экранные формы, которые определяют содержимое диалоговых окон и описывают способы взаимодействия с конкретными устройствами. Для каждой экранной формы указываются поля ввода и перечень элементов управления, действия пользователя (нажать кнопку, выбрать пункт меню, ввести данные, нажать правую/левую кнопку мыши) и отклики системы (отобразить данные, вывести

подсказку, переместить курсор). Такое описание интерфейса представляется в виде таблицы экранных форм.

Дизайн экранных форм представлен на рисунках 2.2-2.3:

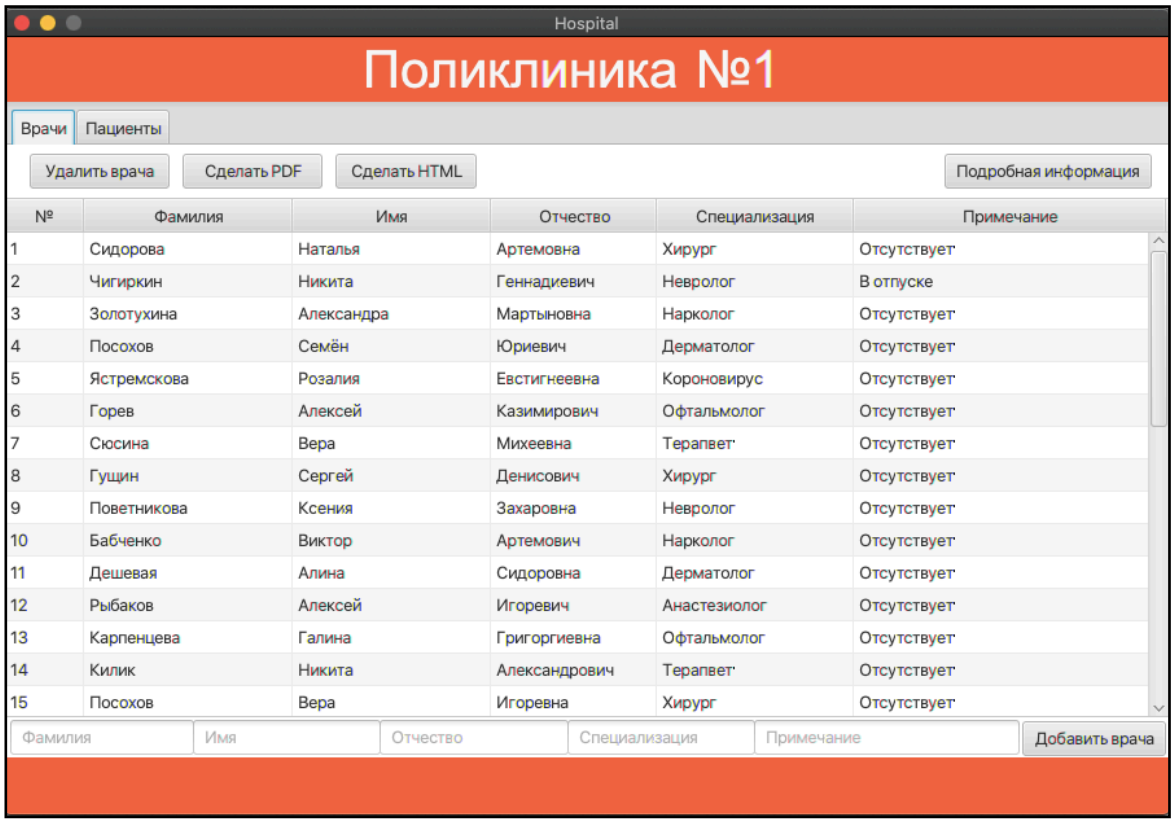


Рисунок 2.2 – Вкладка «Врачи» формы Hospital

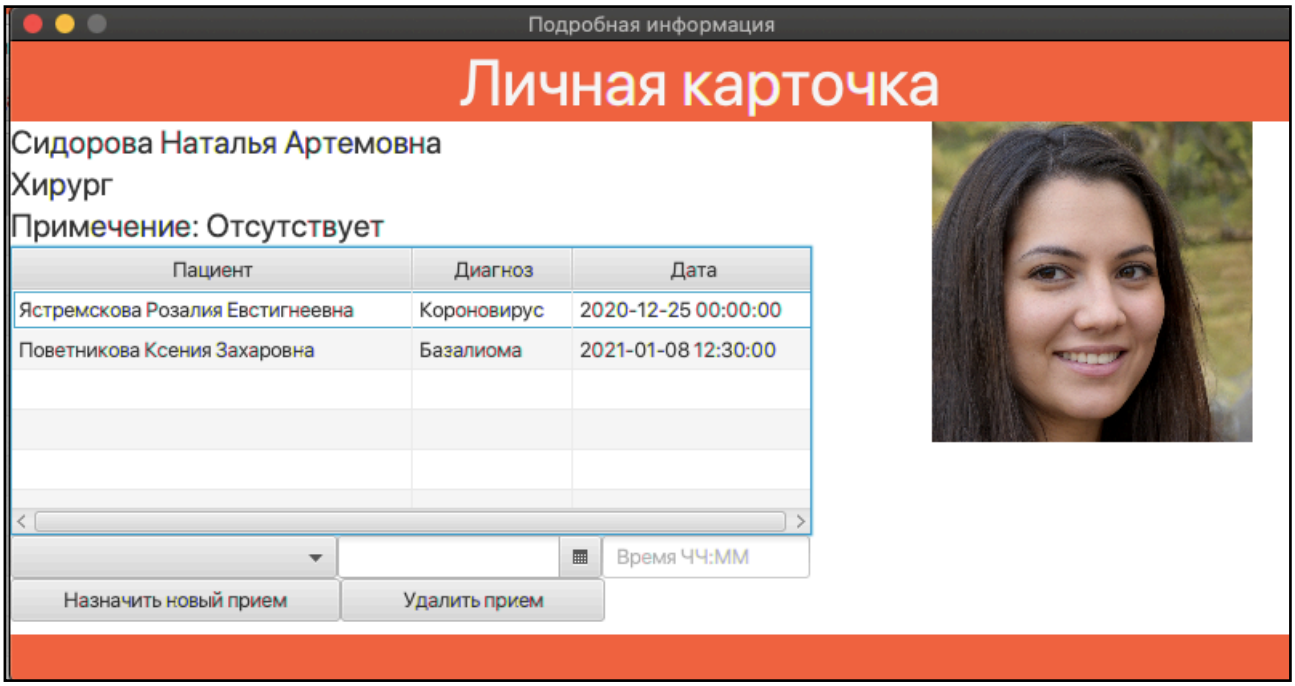


Рисунок 2.3 – Форма «Подробная информация»

Таблица 2.1 - Таблица экранных форм

Экранная форма	**Элементы управления**	**Действия пользователя**	**Отклик системы**
Вкладка «Врачи» формы Hospital	• Кнопки: - Удалить врача - Сделать PDF - Сделать HTML - Добавить врача - Подробная информация • Редактируемая таблица с данными врачей • Поля для ввода информации о новом враче	Выбор врача и кнопка «Удалить врача»	Информационное окно «Врач удален»
		Нажатие кнопки «Удалить» без выбора врача	Окно с предупреждением об отсутствии выбора
		Сделать PDF	Информационное окно «Отчет построен»
		Сделать HTML	Информационное окно «Отчет построен»
		Добавить врача	Окно с предупреждением об отсутствии входных данных
		Ввод информации в поля для ввода текста и «Добавить врача»	Информационное окно «Врач добавлен»
		Двойное нажатие на ячейку в таблице	Возможность изменить информацию в ячейке
		Выбор врача и кнопка «Подробная информация»	Запуск экранной формы «Подробная информация»
Вкладка «Пациенты» формы Hospital	• Кнопки: - Удалить пациента - Сделать PDF - Сделать HTML - Добавить пациента • Редактируемая таблица с данными врачей • Поля для ввода информации о новом пациента	Выбор пациента и кнопка «Удалить пациента»	Информационное окно «Пациент удален»
		Нажатие кнопки «Удалить» без выбора пациента	Окно с предупреждением об отсутствии выбора
		Ввод информации в поля для ввода текста и «Добавить пациенты»	Информационное окно «Пациент добавлен»

Продолжение таблицы 2.1

Экранная форма	Элементы управления	Действия пользователя	Отклик системы
		Добавить врача	Окно с предупреждением об отсутствии входных данных
		Сделать HTML	Информационное окно «Отчет построен»
		Сделать PDF	Информационное окно «Отчет построен»
		Двойное нажатие на ячейку в таблице	Возможность изменить информацию в ячейке
Форма «Подробная информация»	• Таблица с приемами врача • Кнопки: - Назначить новый прием - Удалить прием • ChoiseBox с пациентами • DataPicker - дата нового приема • Поле ввода времени	Ввод информации о приеме и нажатие кнопку «Назначить новый прием»	Добавление приема в таблицу доктора
		Кнопка «Назначить новый прием»	Окно с предупреждением об отсутствии входных данных
		Выбор приема и кнопка «Удалить прием»	Удаление приема из таблицы

2.3. Разработка объектной модели ПК

Объектная модель не описывает структуру ПК, она отображает основные понятия предметной области в виде совокупности типов объектов (сущностей). Сущности строятся путем выделения их из предметной области и анализа прецедентов. На диаграмме сущность обозначается прямоугольником, внутри которого записывается имя сущности, ее атрибуты и операции.

Атрибуты описывают свойства сущности. В объектную модель включаются те атрибуты, для которых определены соответствующие требования или для которых предполагается хранить определенную

информацию. Атрибут характеризуется именем и типом. Для атрибута рекомендуется использовать простые типы данных (число, строка, дата, время и другие).

Описание операций помогает определить поведение объектов сущности. На этом этапе, прежде всего, определяется внутреннее поведение каждого объекта сущности, без учета взаимодействия с другими объектами предметной области. На диаграмме обычно указывается только имя операции, а ее подробное описание приводится в отдельной таблице. В таблице должно содержаться краткое описание назначения операции, ее имя и список входных и выходных параметров.

Ассоциация между сущностями отражает некоторое бинарное отношение между ними. Ассоциация обозначается проведенной между сущностями линией, с которой связывается определенное имя. Имя записывается в глагольной форме, и оно должно отражать семантический смысл отношения. Стрелка на линии указывает, в каком направлении нужно читать имя. На концах линии могут содержаться выражения, определяющие количественную связь между экземплярами сущности (кратность). Кратность определяет, сколько экземпляров одной сущности может быть ассоциировано с одним экземпляром другой сущности. Примеры кратностей:

- 0 ..* - нуль или больше,
- 1 .. * - один или больше,
- 1 – ровно один.

Необходимо устанавливать отношения ассоциации между двумя сущностями в том случае, если объект одной сущности должен знать об объекте другой. Прежде всего, следует включать в модель те ассоциации, которые отражают структурные отношения («содержит», «включает», «хранит» и т.д.), или те, которые должны сохраняться в течение некоторого времени.

Диаграмма сущностей представлена на рис. 2.4. Детальное описание операций представлено в табл. 2.2.

Рисунок 2.4 – Диаграмма сущностей

Таблица 2.2 – Описание операций с сущностями

Сущность	**Имя операции**	**Параметры операции**			**Результат**	**Назначение**
		Вид	**Название**	**Тип**		
Врач	Добавить нового	Вх.	ФИО	Строка	Добавление записи в БД	Добавление нового доктора
		Вх.	Специальность	Строка		
		Вх.	Примечание	Строка		
	Редактировать	Вх.	ФИО	Строка	Изменение записи в БД	Изменение существующей записи в БД
		Вх.	Специальность	Строка		
		Вх.	Примечание	Строка		
	Удалить	Вх.	Врач	Doctor	Удаление записи из БД	Удаление врача из приложения
Пациент	Добавить нового	Вх.	ФИО	Строка	Добавление записи в БД	Добавление нового пациента
		Вх.	Диагноз	Строка		
		Вх.	Примечание	Строка		
	Редактировать	Вх.	ФИО	Строка	Изменение записи в БД	Изменение существующей записи в БД
		Вх.	Диагноз	Строка		
		Вх.	Примечание	Строка		
	Удалить	Вх.	Пациент	Patient	Удаление записи из БД	Удаление врача из приложения
Прием	Добавить новый	Вх.	Пациент	Patient	Создание новой записи	Назначение приема
		Вх.	Дата и время	Строка		
	Удалить	Вх.	Прием	Meeting	Удаление Meeting	Удаление приема

2.4 Построение диаграммы программных классов

Диаграмма классов (class diagram) иллюстрирует спецификации будущих программных классов и интерфейсов. Она строится на основе объектной модели. В описание класса указываются три раздела: имя класса, состав компонентов класса и методы класса. Графически класс изображается в виде прямоугольника. Имя программного класса может совпадать с именем сущности или быть другим. Но поскольку для записи идентификаторов переменных в языках программирования используют латинские буквы, то и имена программных классов и имена их атрибутов, как правило, записываются латинскими буквами. Атрибуты и операции класса перечисляются в горизонтальных отделениях этого прямоугольника. Атрибутам и методам классов должны быть присвоены права доступа. Права доступа помечаются специальными знаками:

\+ - означает открытый (public) доступ;

— - означает скрытый (private) доступ;

- означает наследуемый (protected) доступ.

При описании атрибутов после двоеточия указывается их тип, а при описании методов класса возвращаемое значение (для конструкторов возвращаемое значение не указывается).

В диаграмме классов могут вводиться дополнительно новые атрибуты, операции и связи или осуществляться конкретизация ассоциаций, указанных в объектной модели. На диаграмме классов могут быть три вида отношений: ассоциация, агрегация и наследование.

На диаграмме классов ассоциация имеет такое же обозначение, как и в объектной модели. На линиях ассоциации может присутствовать стрелка. Это стрелка видимости, которая показывает направление посылки запросов в ассоциации. Стрелка видимости также показывает, какой из классов содержит компоненты для реализации отношения ассоциации, иными словами, кто является инициатором посылки запроса к другому объекту. Ассоциация без стрелки является двунаправленной.

Агрегирование - это отношение между классами типа целое/часть. Агрегируемый класс в той или иной форме является частью агрегата. На практике это может быть реализовано по-разному. Например, объект класса-агрегата может хранить объект агрегируемого класса, или хранить ссылку на него. Агрегирование изображается на диаграмме полым ромбом на конце линии со стороны агрегирующего класса (агрегата). Если агрегируемый объект может быть создан только тогда, когда создан агрегат, а с уничтожением агрегата уничтожаются и все агрегируемые объекты, то такое агрегирование называется сильным и отображается в виде закрашенного ромба.

Наследование - это отношение типа общее-частное между классами. Его следует вводить в том случае, когда поведение и состояние различных классов имеют общие черты. Наследование связывает конкретные классы с общими или в терминологии языков программирования производные классы (подклассы) с базовыми классами (суперклассами). На диаграммах наследование изображается в виде стрелки с полым треугольником, идущей от производного класса к базовому. Если один производный класс наследует несколько базовых, то такое наследование называется множественным.

Диаграмма классов представлена в приложении А.

2.5 Описание поведения ПК

Поведение ПК представляет собой описание того, какие действия выполняет ПК, без определения механизма их реализации. Одной из составляющей такого описания является диаграмма последовательностей (seguence diagram). Диаграмма последовательностей является схемой, которая для определенного сценария прецедента показывает генерируемые пользователями и объектами события (запросы) на выполнение некоторой операции и их порядок. Диаграммы последовательности имеют две размерности: вертикальная представляет время, горизонтальная - различные объекты. Чтобы построить диаграмму последовательностей необходимо выполнить следующие действия:

1. Идентифицировать пользователей и объекты программных классов, участвующие в начальной стадии реализации сценария прецедента, и их изображения в виде прямоугольников расположить наверху в одну линию. Для каждого пользователя и объекта нарисовать вертикальную пунктирную линию, которая является линией их жизни. Внутри прямоугольника указываются подчеркнутое имя объекта и имя класса, к которому принадлежит объект.

2. Из объектной модели выбрать те операции, которые участвуют в реализации сценария. Если такие операции не были определены при построении диаграммы программных классов, то необходимо их описать и внести в модель.

3. На диаграмме последовательностей каждому запросу на выполнение операции должна соответствовать горизонтальная линия со стрелкой, начинающаяся от вертикальной линии того пользователя или объекта, который вызывает операцию, и заканчивающаяся на линии жизни того пользователя или объекта, который будет ее выполнять. Над стрелкой указывается номер операции, число итераций, имя операции и в скобках ее параметры. После описания операции может следовать комментарий, поясняющий смысл операции и начинающийся со знака "//".

Операция, которая реализует запрос, на линии жизни объекта обозначается прямоугольником. Порядок выполнения операций определяется ее номером, который указывается перед именем, и положением горизонтальной линии на диаграмме. Чем ниже горизонтальная линия, тем позже выполняется операция. В диаграммах последовательности принято применять вложенную систему нумерации, так как это позволяет отобразить их вложенность. Нумерация операций каждого уровня вложенности должна начинаться с 1.

На диаграмме последовательностей можно описать вызов операции по условию (конструкция if-else) и показать моменты создания и уничтожения объектов. Если объект создается или уничтожается на отрезке времени, представленном на диаграмме, то его линия жизни начинается и заканчивается

в соответствующих точках, в противном случае линия жизни объекта проводится от начала до конца диаграммы. Символ объекта рисуется в начале его линии жизни; если объект создается не в начале диаграммы, то сообщение о создании объекта рисуется со стрелкой, проведенной к символу объекта. Если объект уничтожается не в конце диаграммы, то момент его уничтожения помечается большим крестиком "X".

Диаграмма последовательностей для операции копирования строк полученных в БД в список объектов представлена на рис. 2.5.

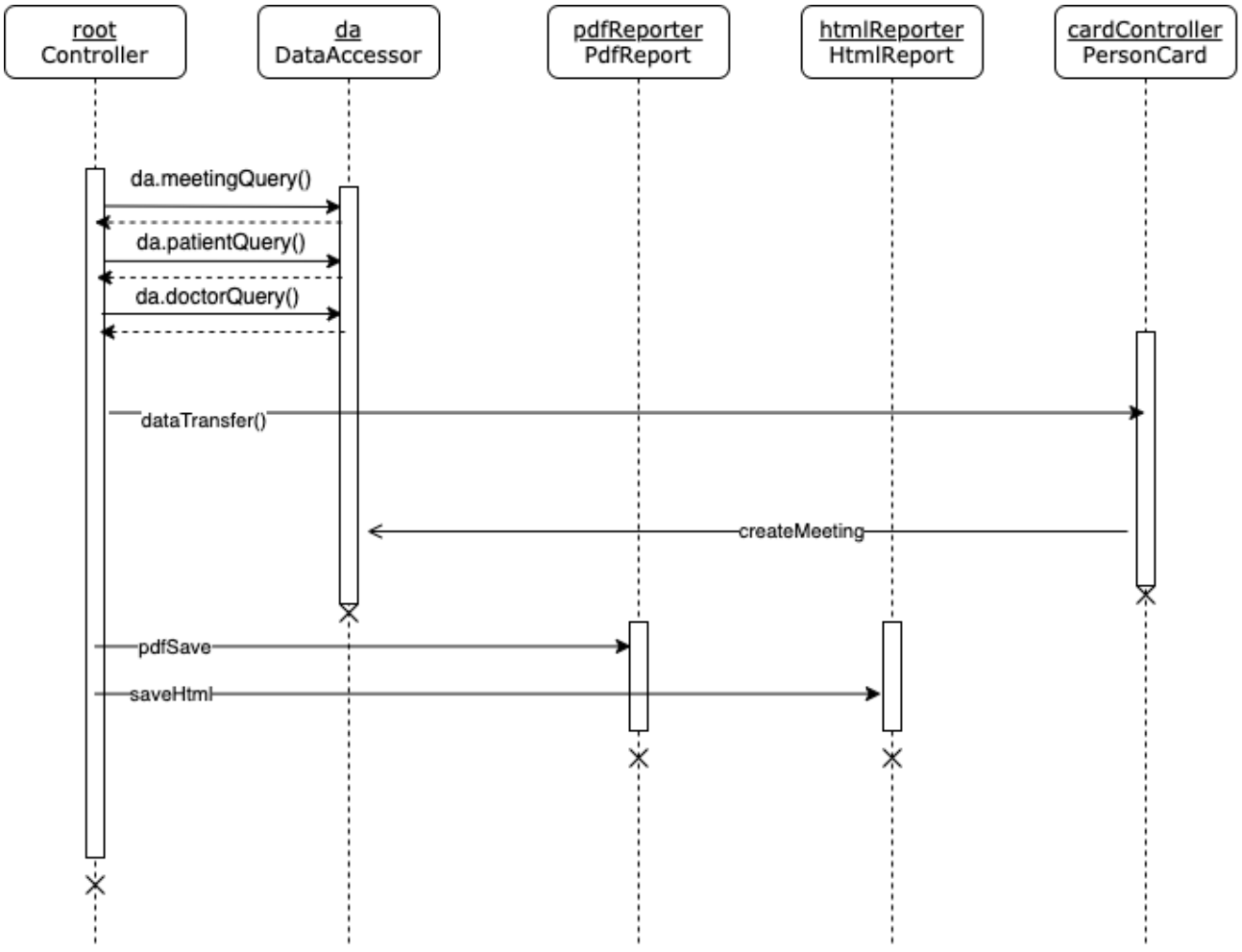


Рисунок 2.5 – Диаграмма последовательностей

2.6 Построение диаграммы действий

Диаграмма действий (activity diagram) строится для сложных операций. Основным направлением использования диаграмм деятельности является визуализация особенностей реализации операций классов, когда необходимо

представить алгоритмы их выполнения. Графически диаграмма деятельности представляется в форме графа деятельности, вершинами которого являются действия, а дугами — переходы от одного действия к другому. Она очень похожа на блок-схемы алгоритмов. Каждая диаграмма деятельности должна иметь единственное начальное и единственное конечное состояние. Диаграмму деятельности принято строить таким образом, чтобы действия следовали сверху вниз. Отличительной чертой диаграммы действий является то, что в ней можно отобразить параллельные процессы. Для этой цели используется специальный символ (линия синхронизации), который позволяет задать разделение и слияние потоков управления. При этом разделение имеет один входящий переход и несколько выходящих, а слияние, наоборот, имеет несколько входящих переходов и один выходящий.

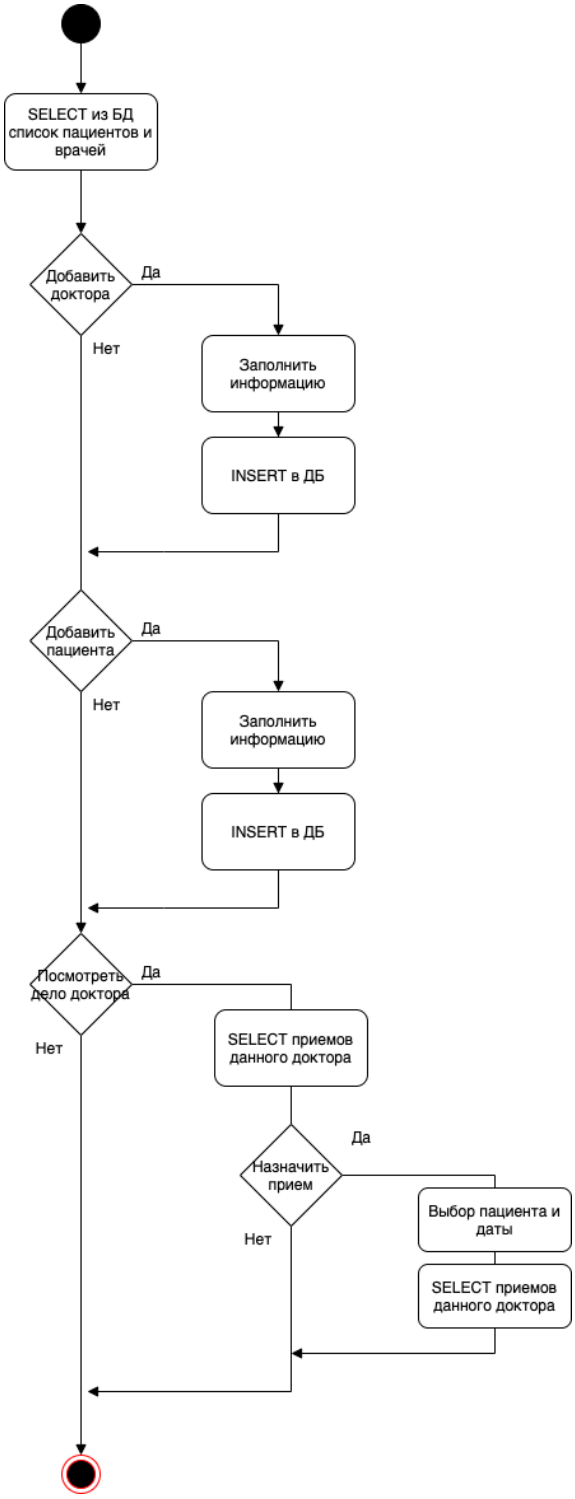


Рисунок 2.6 Диаграмма действий

В общем случае действия на диаграмме деятельности выполняются над теми или иными объектами. Эти объекты либо инициируют выполнение действий, либо определяют некоторый результат этих действий. При этом действия специфицируют вызовы, которые передаются от одного объекта графа деятельности к другому. Чтобы связать объекты с действиями, необходимо явно

указать их на диаграмме деятельности. Для графического представления объектов, используются прямоугольник, в котором указывается подчеркнутое имя класса. Подчеркнутое имя означает, что на диаграмме задается объект, а не его класс. Далее после имени можно указать в прямых скобках значения атрибутов объекта после выполнения предшествующего действия. Такие прямоугольники объектов присоединяются к переходам отношением зависимости с помощью пунктирной линией со стрелкой.

3. Руководство оператора

3.1 Назначение программы

ПК «Регистрация, учет, администрирование, редактирование и выдача сведений о врачах и пациентах» должен входить в состав автоматизированной системы учета и администрирования информации, и предназначен для автоматизации деятельности ОЛ, ответственного за учет прихода и расхода лекарств.

В рамках ПК «Регистрация, учет, администрирование, редактирования и выдача сведений о книгах» ОЛ может:

- добавлять, править и удалять информацию о врачах;
- добавлять, править и удалять информацию о пациентах;
- добавлять и удалять о приемах пациентов у врачей;

3.2 Условия выполнения программы

ПК предназначен для функционирования под операционной средой Windows 10, OS X Catalina при поддержке PostgreSQL.

Персональная электронно-вычислительная машина (ПЭВМ) должна обладать следующими характеристиками:

1. тип процессора Pentium III 500 и выше;
2. объем ОЗУ – не менее 128Мб;
3. объем жесткого диска – не менее 4Гб;
4. видеокарта – 64Мб;
5. стандартная клавиатура;
6. манипулятор типа "мышь".

3.3 Описание задачи

В ПК должны храниться сведения о врачах, пациентах и приемах. Администратор библиотеки может добавлять, изменять и удалять эти сведения. Ему может потребоваться следующая информация:

- права доступа пользователя;
- имеется ли книга в библиотеки;
- до какого числа можно выдать книгу;
- предыдущие задолженности пользователя.

Обязательными требованиями при разработке кода ПК являются использование следующих конструкций языка Java:

- закрытые и открытые члены классов;
- наследование;
- конструкторы с параметрами;
- абстрактные базовые классы;
- виртуальные функции;
- обработка исключительных ситуаций;
- динамическое создание объектов.

С целью выполнения поставленной задачи в процессе проектирования разработана общая модель ПК с выявлением основных объектов и связей между ними. На основании полученной модели разработаны программные классы. Данные об информационных объектах хранятся в базе данных.

Требования к коду ПК учтены при создании программных классов и непосредственном написании программы.

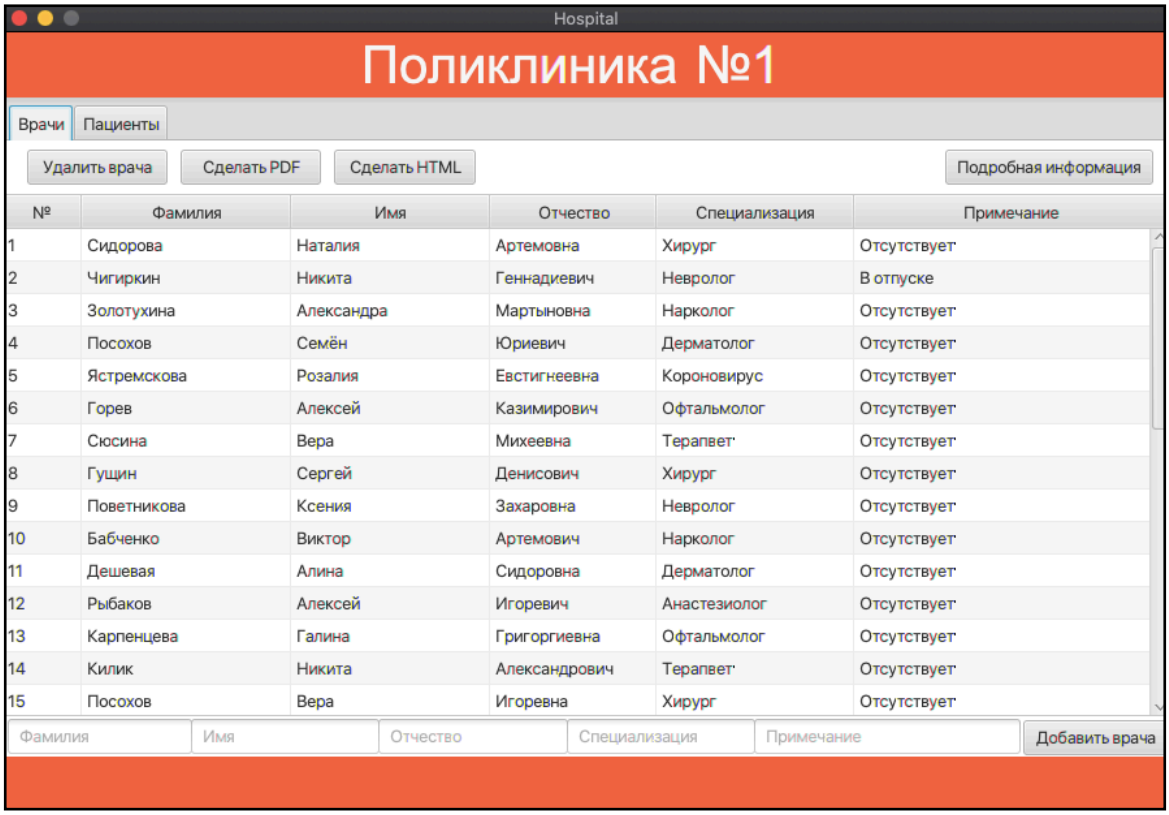
3.4 Входные и выходные данные

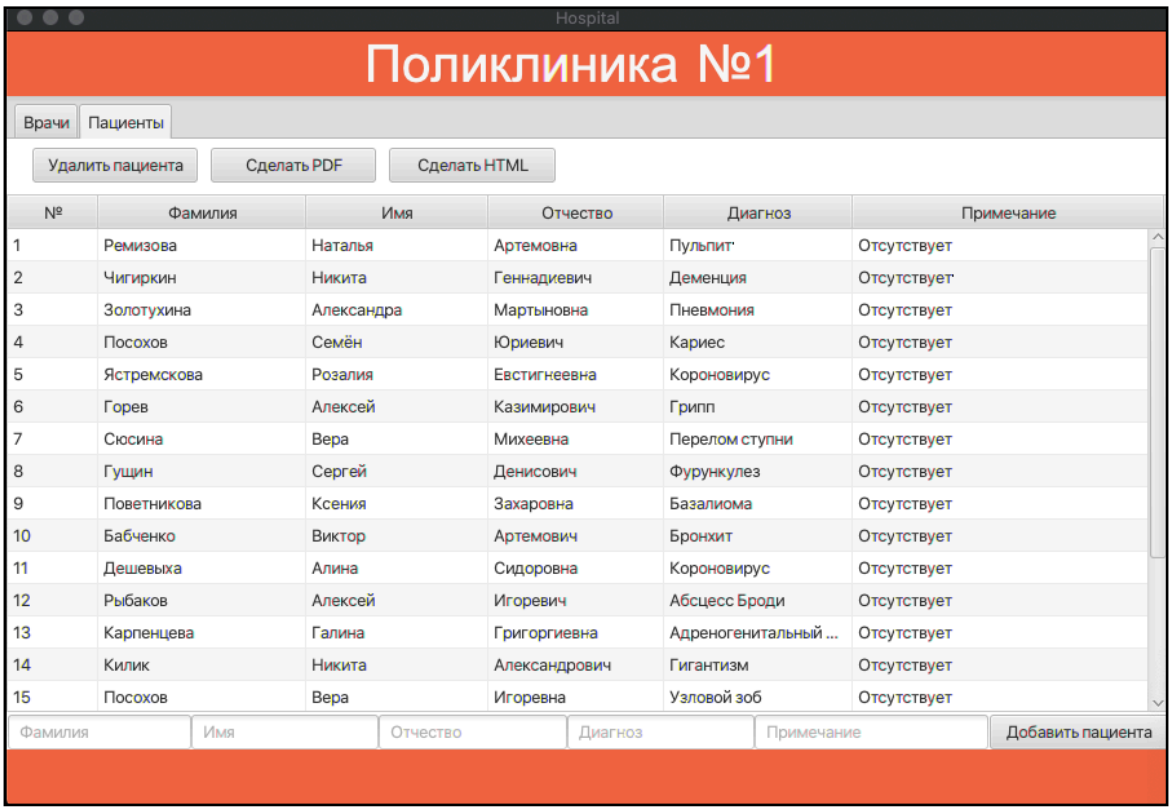
Выходные данные должны быть представлены в виде таблице содержащий описание необходимых информационных объектов, выполненного посредствам представления его характеристик.

Входная информация для задачи «Регистрация, учет, администрирование, редактирование и выдача сведений о врачах и пациентах» содержится в приходно-расходной документации. Ввод исходных данных должен осуществляется ОЛ в режиме диалога. Вводимые данные являются значениями характеристик (атрибутов) информационных объектов. Вводимая информация может выбираться или набираться из списка предлагаемых значений.

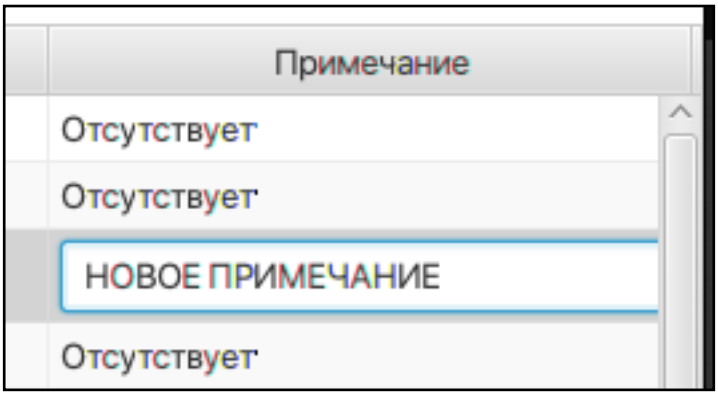
3.5 Выполнение программы

При запуске программы выводится список врачей и пациентов из базы данных. Данные о врачах и пациентах доступны на разных вкладках.

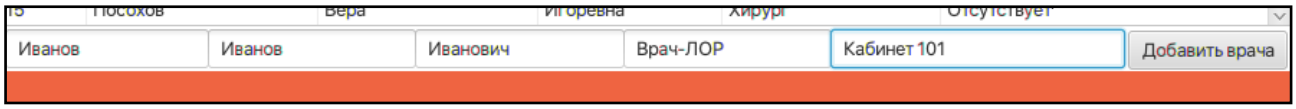


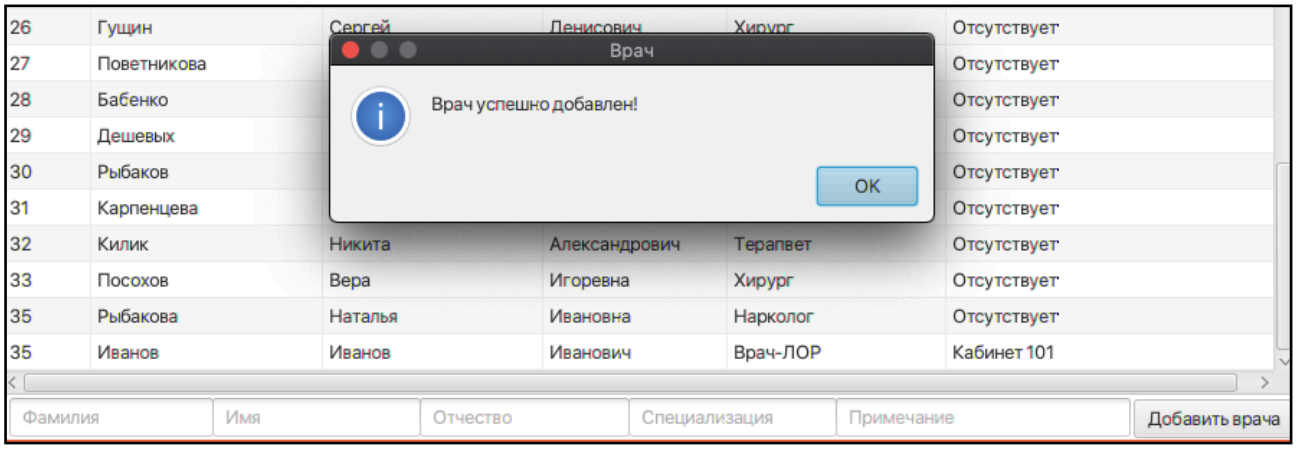


При двойном нажатии можно редактировать информацию в таблице. Все изменения сразу фиксируются в базе данных.

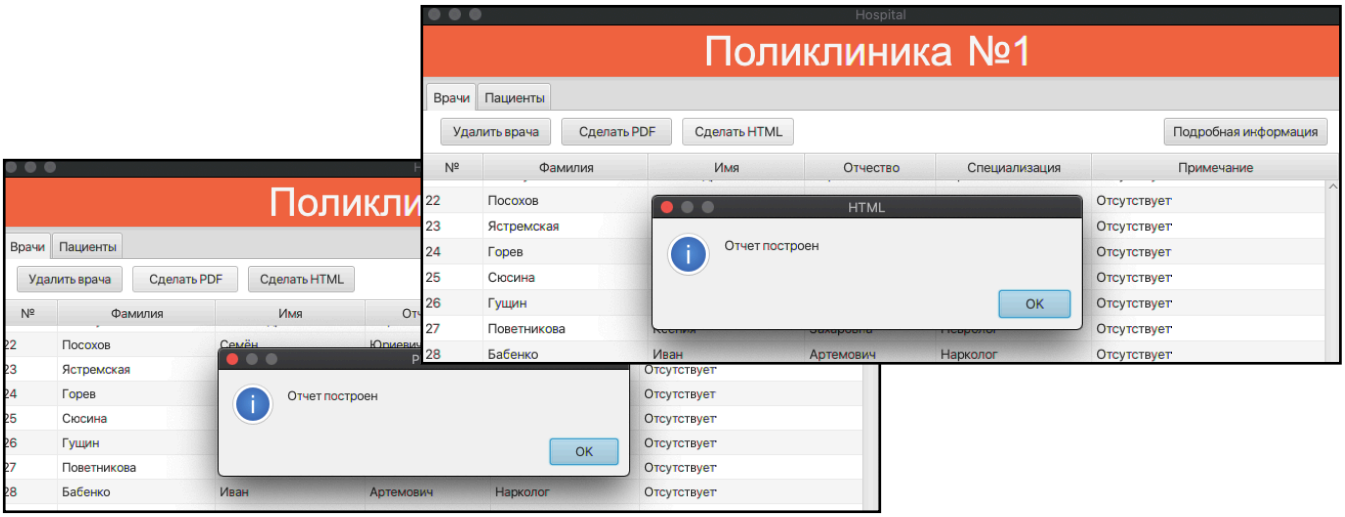


Имеется возможность добавлять новых врачей и пациентов. Новые люди сразу добавляются в базу данных.

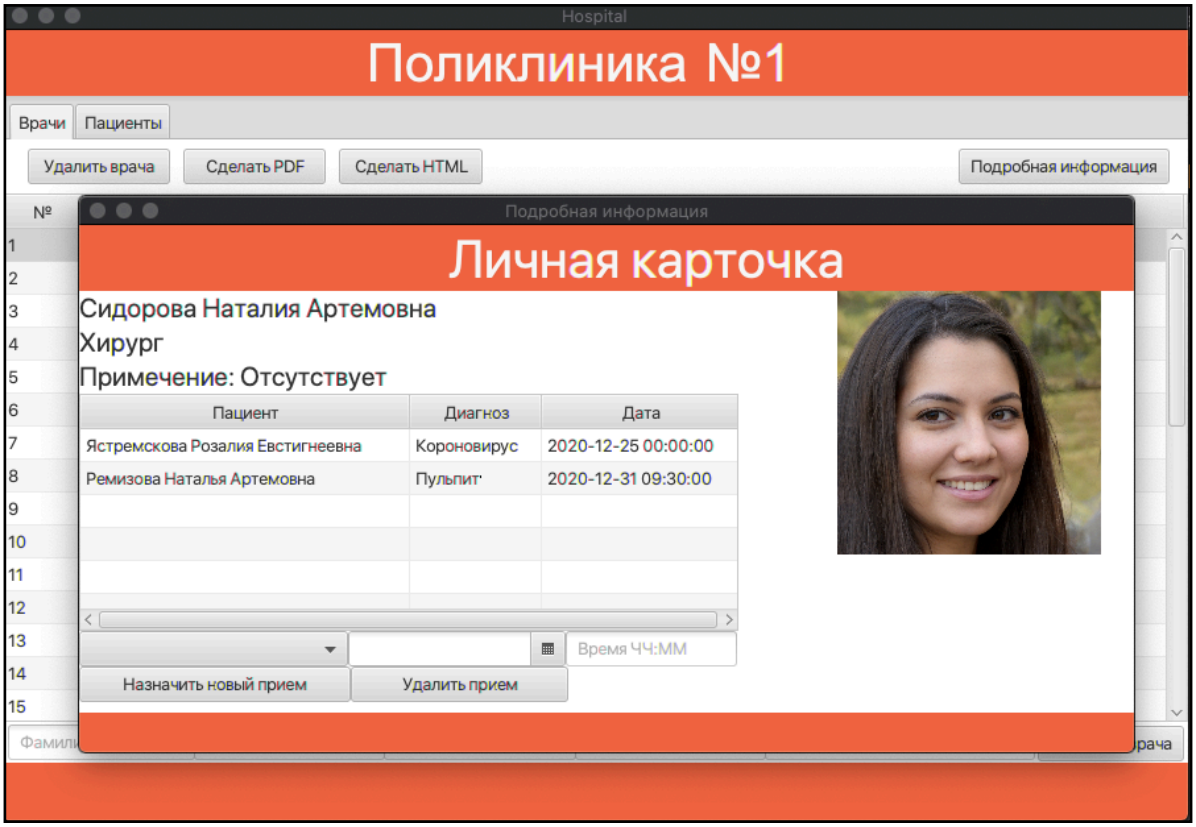




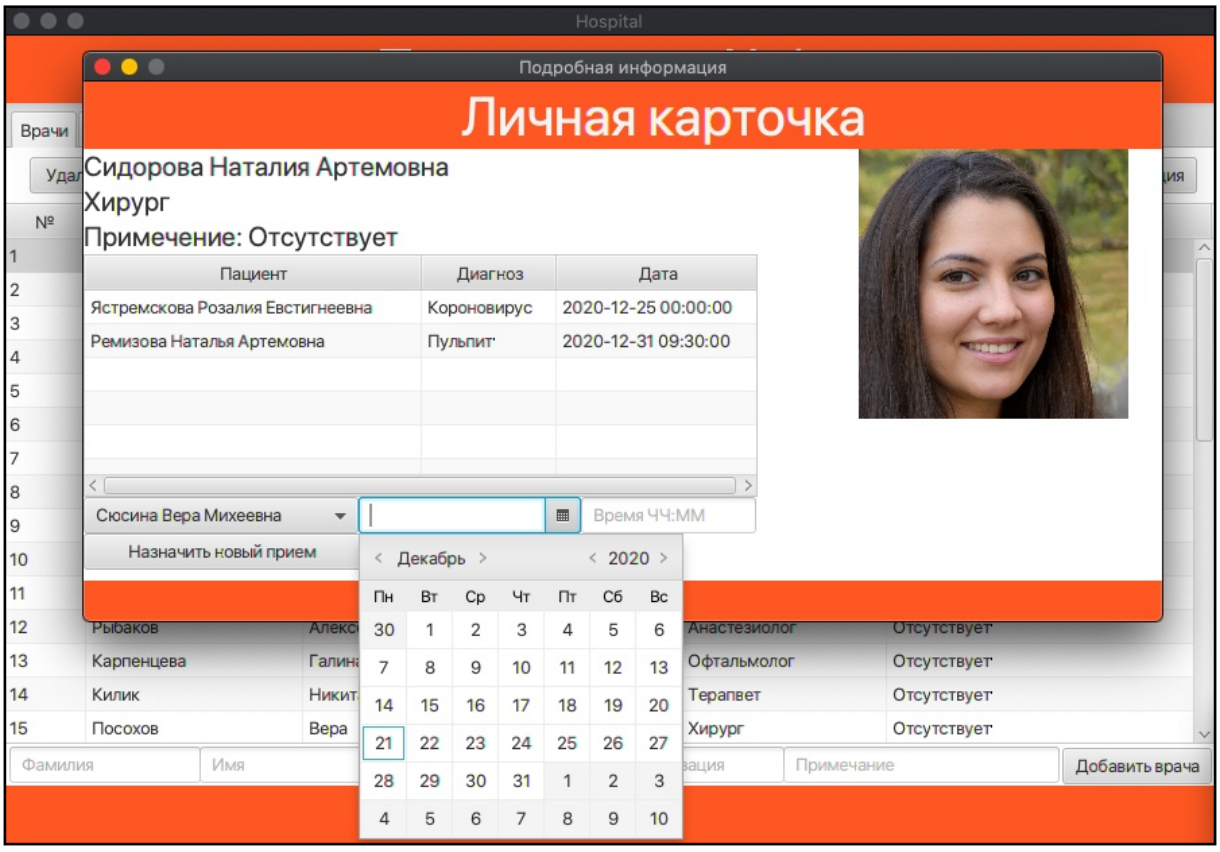
Из лабораторных работ оставлена возможность строить отчеты в PDF и HTML.



При выборе врача и нажатии на кнопку можно посмотреть подробную информацию о враче, которая включает журнал приемов врача



Приемы можно удалять, а можно назначать новые:



Заключение

В результате проделанной работы разработан ПК «Регистрация, учет, администрирование, редактирования и выдача сведений о врачах и пациентах», предназначенный для администрирования и учета информации о врачах и пациентах поликлиники, разработано руководство оператора.

В процессе проектирования созданы описание вариантов использования ПК, прототип интерфейса пользователя, объектная модель ПК, диаграмма классов, описание поведения ПК, диаграмма действия ПК.

ПРИЛОЖЕНИЕ А