Лабораторная работа №14

Тема «Разработка ТП на создаваемую систему»

ВЫПОЛНИЛ

Олишкевич Игорь Русланович

Группа: 25-ПО

Задание: разработка ТП на создаваемую систему «АРМ администратора офтольмологической клиники «Хорошее зрение».

Решение:

**1 ОБЩИЕ СВЕДЕНИЯ**

Используя данные о работе администратора клиники «Хорошее зрение»: управление данными о работниках: ФИО, должность, дата приема на работу, график работы, номер кабинета, специализация, пол, трудовая книжка, фиксированная заработная плата, процентные начисления, количество приемов; личная карточка клиента: ФИО, дата приема, цель приема, заболевания, медицинское заключение, оплата; график приема: день недели, время, принимающий врач; табель учета рабочего времени; оборудование: состояние, количество, стоимость; услуги: информация, стоимость, принимающие врачи; разработать программу на языке C#, которая автоматизирует работу администратора клиники и позволит осуществить управление данными об услугах и клиентах.

В программе необходимо предусмотреть: базу данных, с возможностью добавления, редактирования и удаления содержимого; выборку по дате записи, оплате, данных клиента, услуги, оборудования, начислениям работников, учету состояния в табеле рабочего времени; подбор врача по услуге; создание медицинской карточки пациента; регистрацию на прием с учетом услуги и врача; ведение табеля учета рабочего времени и экспорт в текстовый документ; слежение за ходом лечения пациента; ведение журнала жалоб и предложений, в который будут записываться отзывы клиентов.

Как результат работы администратора клиники предусмотреть формирование следующих документов: финансовый отчет по прибыли, выплате сотрудникам заработной платы и процентных начислений, выполненных врачебных манипуляциях, износу оборудования, статистика посещений пациентов, техническая документация по состоянию оборудования, медицинские заключения и карточки пациентов, отчет по состоянию работников в табеле рабочего времени, статистика выполненных услуг, поиск по заданным критериям: дате записи, оплате, данных клиента, услуги, оборудования, начислениям работников. Предусмотреть вывод документации в Excel и Word, печать данных.

**2 ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ**

Офтальмолог – это врач, который занимается диагностикой и лечением заболеваний, связанных с глазами и зрением. Офтальмология подразделяется на детскую и взрослую, обе специальности имеют свои особенности.

Диаграмма вариантов использования описывает организацию поведения системы: совокупность прецедентов и актеров, а также отношения между ними. Иллюстрируется статический вид системы с точки зрения прецедентов, что необходимо для организации и моделирования поведения.

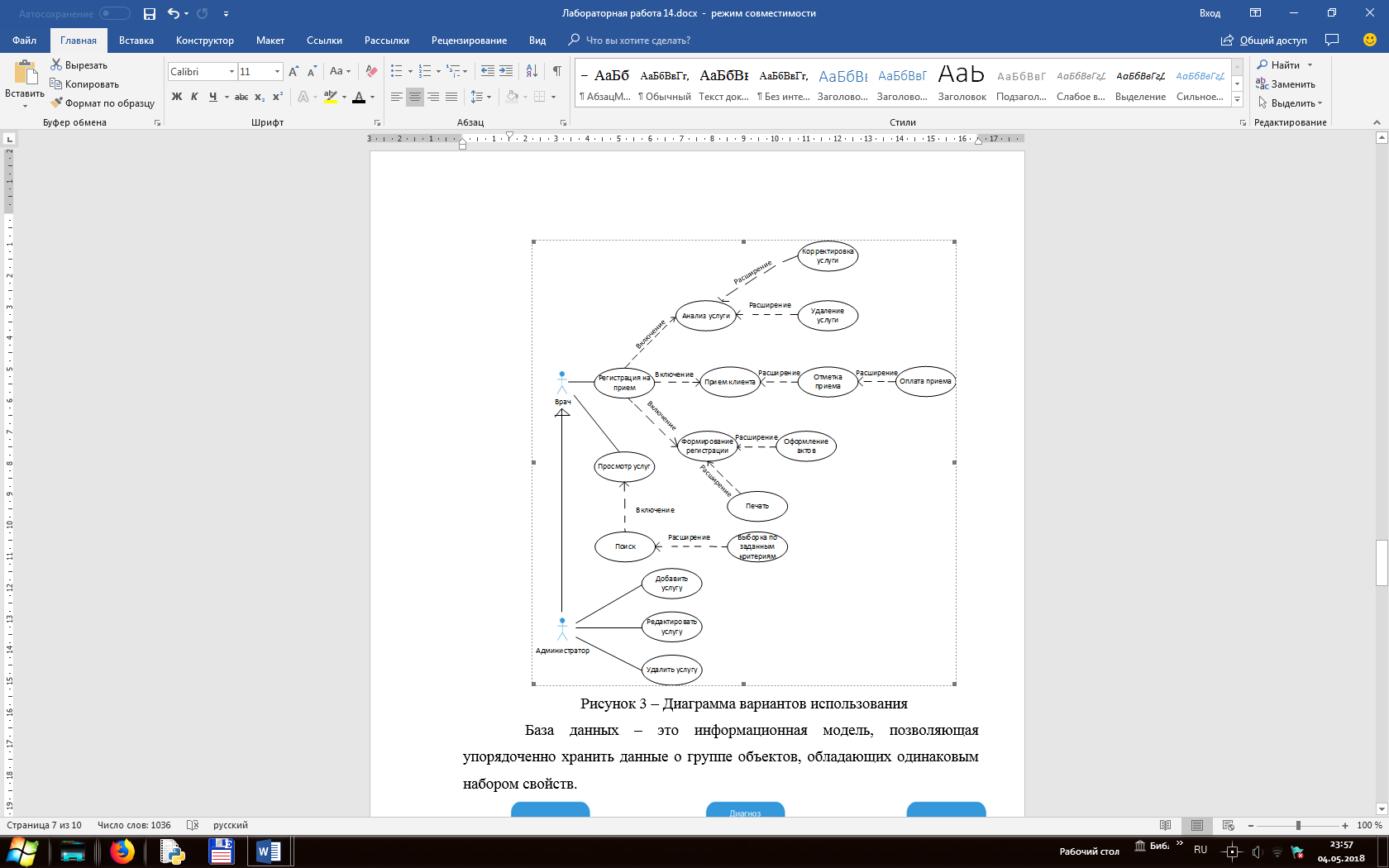


Рисунок 1 – Диаграмма вариантов использования

Цели деятельности можно охарактеризовать таким образом:

1. ведение врачебной документации;
2. регистрация клиентов;
3. заведение медицинских карт;
4. прием клиентов по различным услугам;
5. квалифицированная врачебная помощь.

Основными задачами являются:

1. диагностика;
2. лечение;
3. профилактика заболеваний, связанных с нарушением зрительной системы.

**3 ОПИСАНИЕ ПРОЕКТНЫХ РЕШЕНИЙ**

При проектировании системы использованы следующие схемы решений, влияющих на архитектуру системы:

1. принцип открытости, состоящий в способности системы к расширению состава предоставляемых сервисов и технологий, увеличению числа набора данных предметной области в связи с расширением сервера и сегментации информации, ее пользователей без нарушения регламентов внутреннего функционирования системы и без ухудшения эксплуатационных характеристик;
2. принцип стандартизации (унификации), состоящий в необходимости рационального применения типовых, унифицированных или стандартизированных проектных решений и технологий, внутренних и внешних интерфейсов и протоколов, конфигурационных файлов и используемых библиотек, что закладывает фундамент для блочного и модульного построения компонентов, подсистем и системы в целом, а также обеспечивает согласованность процедур сбора и обработки информации участниками информационного взаимодействия в рамках выполнения закрепленных за ними функций и имеющихся полномочий;
3. принцип преемственности, позволяющий при развитии системы обеспечить сохранение, развитие и эффективное использование существующей информации в сочетании с эффективными и рациональными методами и способами ее сбора, хранения и предоставления, а также максимально использовать при развитии возможности существующих материальных и нематериальных ресурсов системы.

Информационный обмен между компонентами системы обеспечивается с использованием открытых технологий и стандартов информационного взаимодействия.

**3.1  Решения по архитектуре**

Архитектура программного обеспечения – совокупность важнейших решений об организации программной системы. Документирование архитектуры программного обеспечения упрощает процесс коммуникации между разработчиками, позволяет зафиксировать принятые проектные решения и предоставить информацию о них эксплуатационному персоналу системы.

На диаграмме классов отображено множество классов, интерфейсов, коопераций и их отношений. Используется для иллюстрации статического вида системы с точки зрения проектирования. Диаграммы, на которых показаны активные классы, применяются для работы со статическим видом системы с точки зрения процессов.

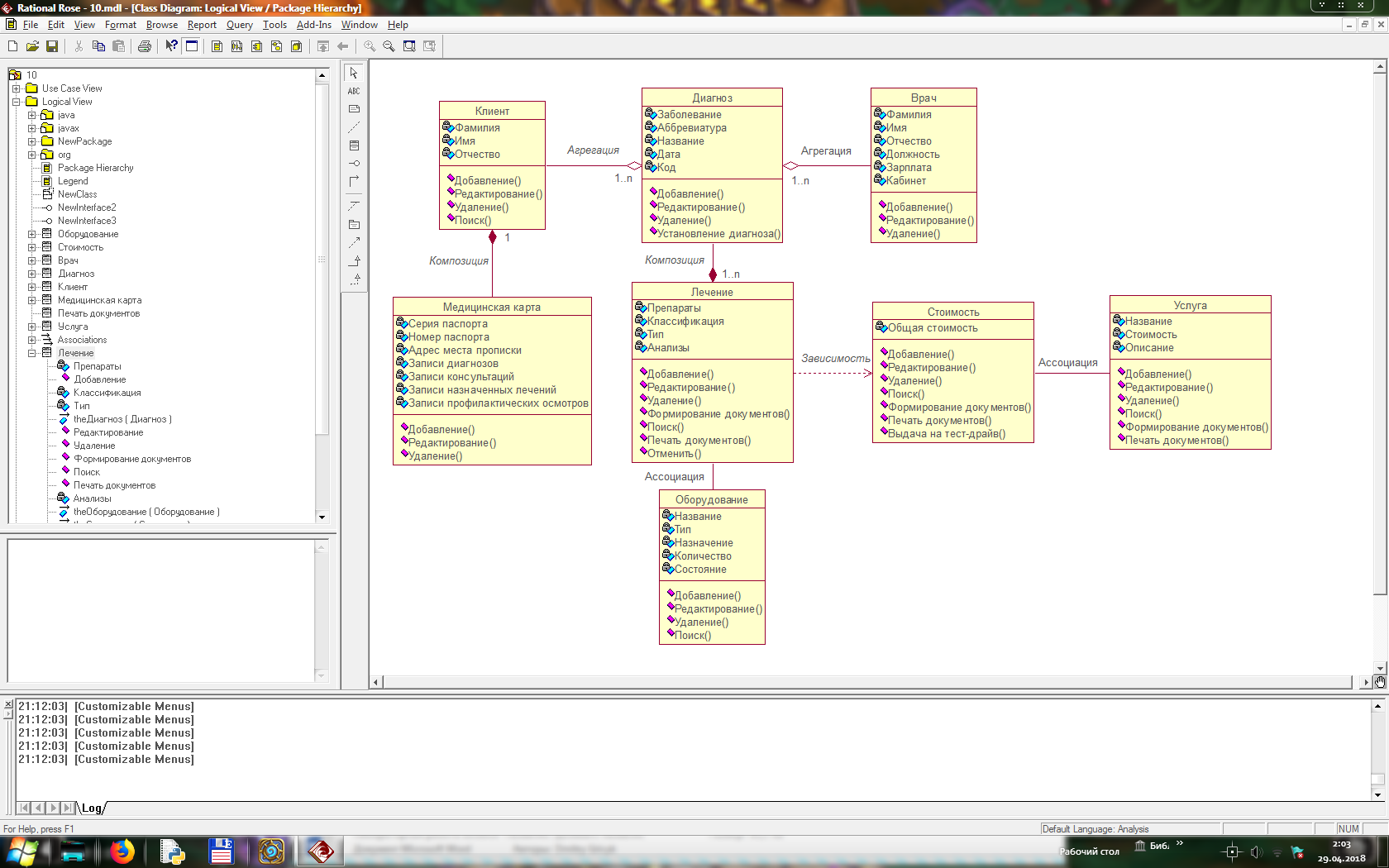


Рисунок 2 – Диаграмма классов

На диаграммах компонентов показаны множества компонентов и отношения между ними. Иллюстрируется статический вид системы с точки зрения реализации. Диаграммы компонентов соотносятся с диаграммами классов.

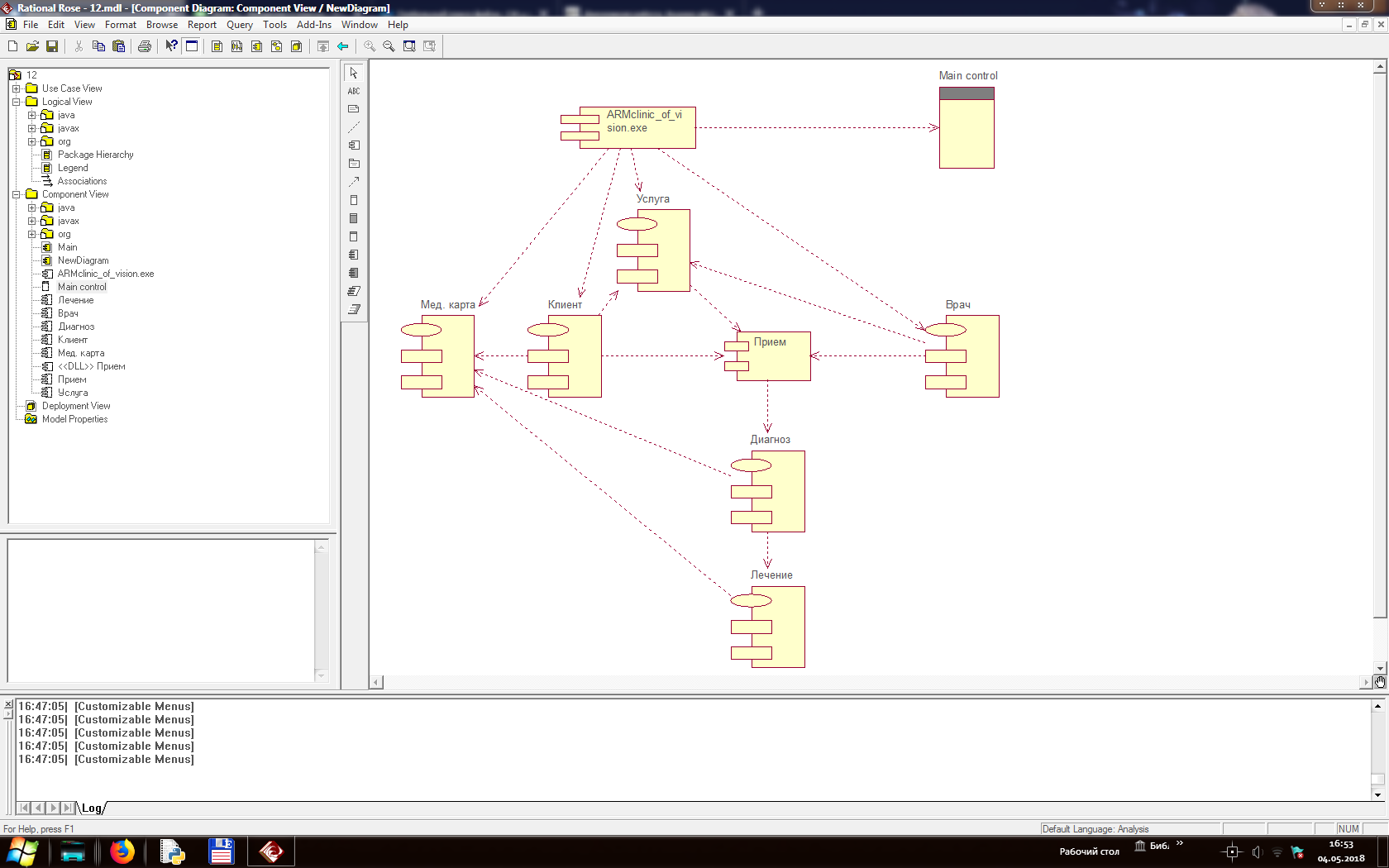


Рисунок 3 – Диаграмма компонентов

На диаграммах развертывания представлены узлы и отношения между ними. С помощью таких изображений иллюстрируют статический вид системы с точки зрения развертывания. Они соотносятся с диаграммами компонентов, так как узел обычно содержит один или несколько компонентов.

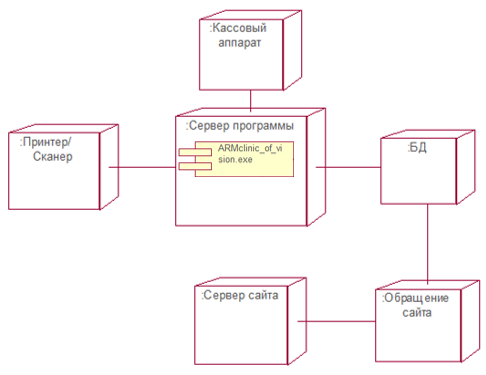


Рисунок 4 – Диаграмма развертывания

На диаграмме объектов отображено множество объектов и отношений между ними. Используется для иллюстрации структуры данных экземпляров сущностей, представленных на диаграмме классов. Диаграммы объектов относятся к статическому виду системы с точки зрения процессов, заостряют внимание на реальных или модельных прецедентах.

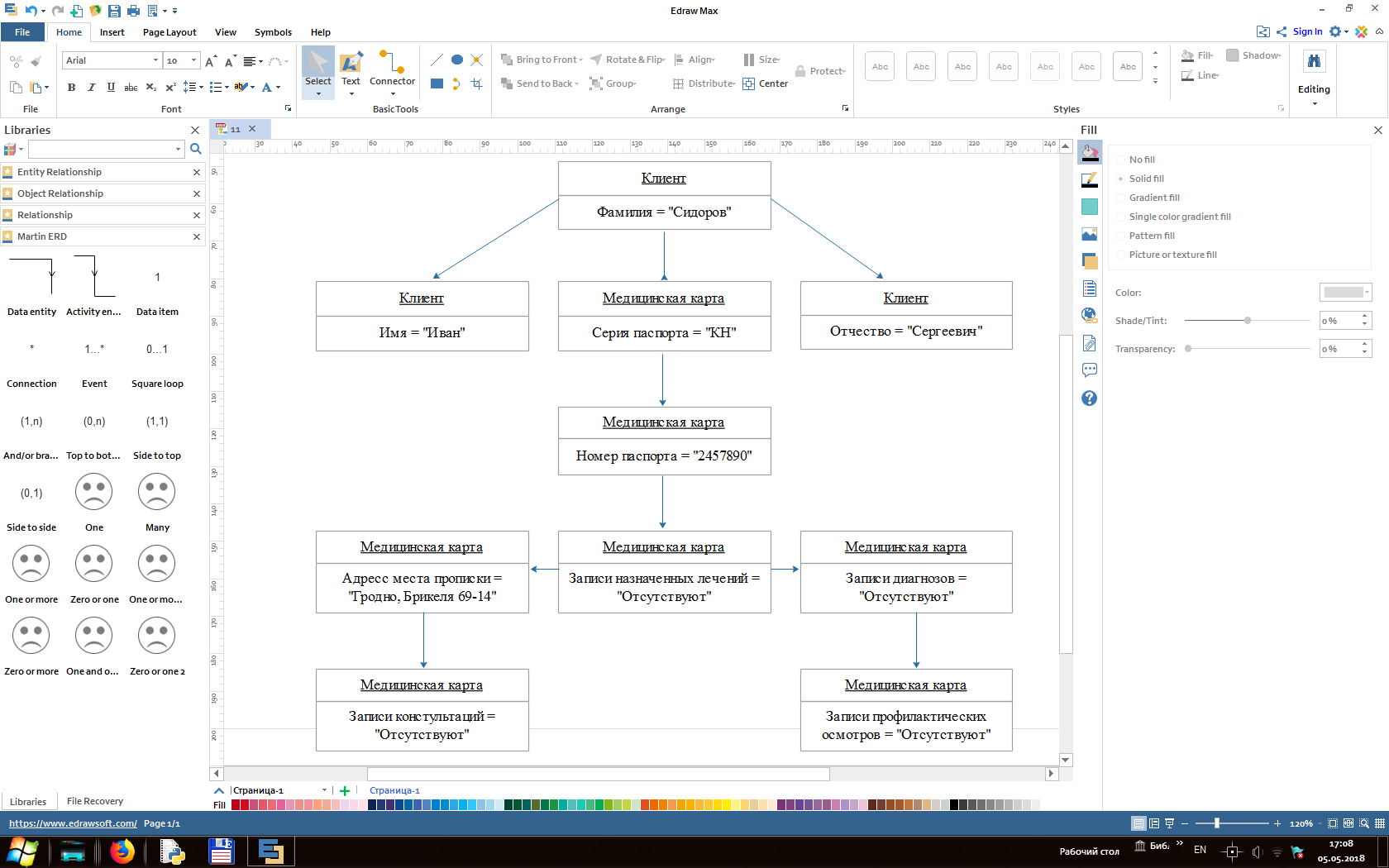


Рисунок 5 – Диаграмма объектов

**3.2 Информационная модель и структура базы данных**

Информационная модель – модель объекта, представленная в виде информации, описывающей существенные для данного рассмотрения параметры и переменные величины объекта, связи между ними, входы и выходы объекта и позволяющая путём подачи на модель информации об изменениях входных величин моделировать возможные состояния объекта.

База данных – это информационная модель, позволяющая упорядоченно хранить данные о группе объектов, обладающих одинаковым набором свойств.

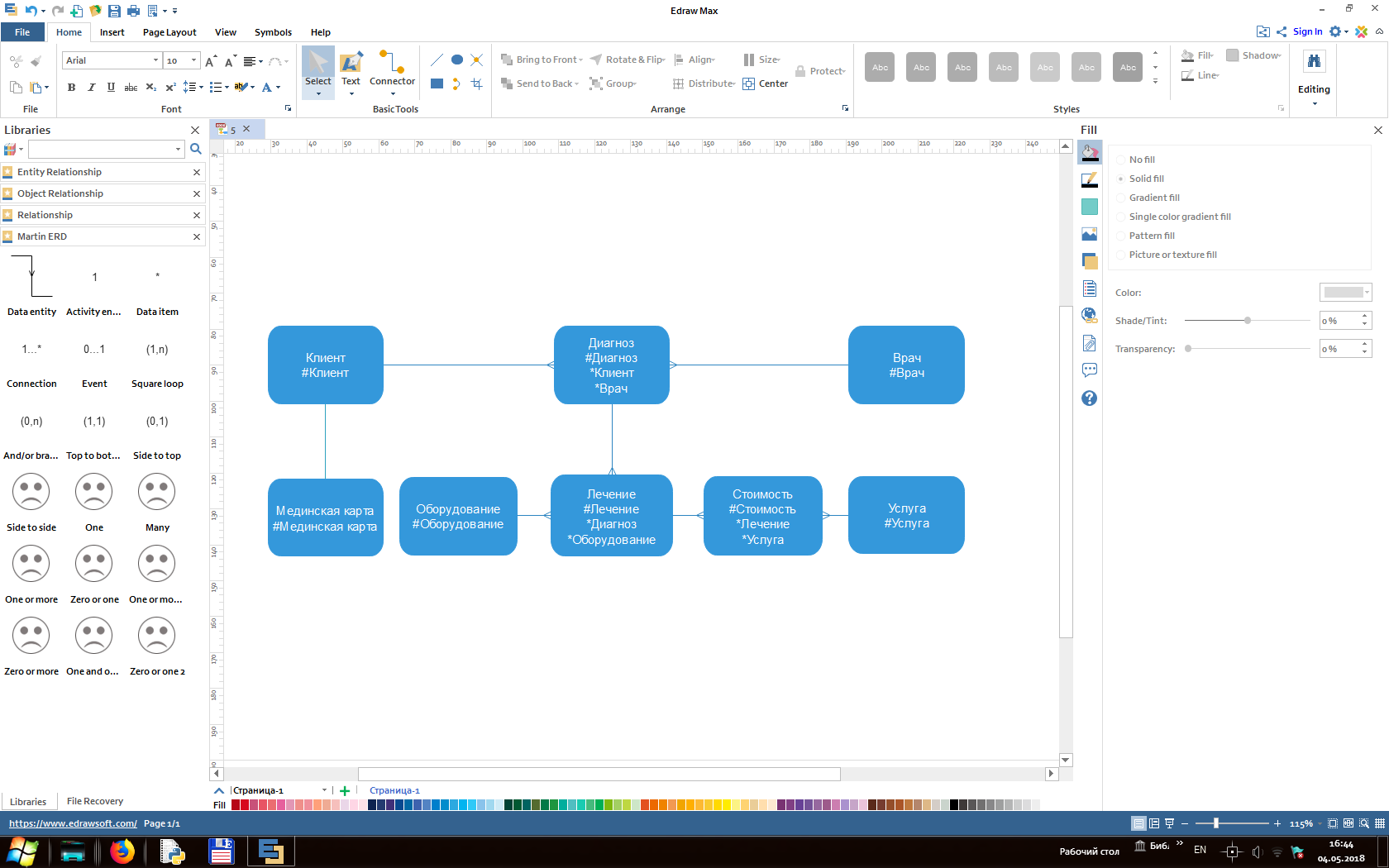


Рисунок 6 – Схема базы данных

Структурная таблица «Медицинская карта»:

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип данных | Ключевое поле | Описание поля |
| #id\_med\_record | integer | + | Код медицинской карты |
| passport\_series | varchar (255) |  | Серия паспорта |
| passport\_numbers | integer |  | Номер паспорта |
| adress | varchar (255) |  | Адрес места прописки |
| diagnosis | text |  | Записи диагнозов |
| consultation | text |  | Записи консультаций |
| treatment | text |  | Записи назначенных лечений |
| inspection | text |  | Записи профилактических осмотров |

Структурная таблица «Клиент»:

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип данных | Ключевое поле | Описание поля |
| #id\_client | integer | + | Код клиента |
| \*id\_med\_record | integer |  | Код мед. карты |
| last\_name | varchar (255) |  | Фамилия клиента |
| first\_name | varchar (255) |  | Имя клиента |
| sure\_name | varchar (255) |  | Отчество клиента |

Структурная таблица «Диагноз»:

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип данных | Ключевое поле | Описание поля |
| #id\_diagnosis | integer | + | Код диагноза |
| \*id\_client | integer |  | Код клиента |
| \*id\_doctor | integer |  | Код врача |
| disease | varchar (255) |  | Заболевание клиента |
| abbreviation | varchar (255) |  | Аббревиатура диагноза |
| name | varchar (255) |  | Наименование диагноза |
| date | date |  | Дата постановления диагноза |
| code | integer |  | Код диагноза |

Структурная таблица «Врач»:

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип данных | Ключевое поле | Описание поля |
| #id\_doctor | integer | + | Код врача |
| last\_name | varchar (255) |  | Фамилия врача |
| first\_name | varchar (255) |  | Имя врача |
| sure\_name | varchar (255) |  | Отчество врача |
| position | varchar (255) |  | Должность врача |
| salary | decimal |  | Зарплата врача |
| room | integer |  | Кабинет врача |

Структурная таблица «Лечение»:

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип данных | Ключевое поле | Описание поля |
| #id\_ treatment | integer | + | Код лечения |
| \*id\_diagnosis | integer |  | Код диагноза |
| drugs | text |  | Препараты |
| classification | varchar (255) |  | Классификация лечения |
| type | varchar (255) |  | Тип лечения |
| analyzes | text |  | Анализы |

Структурная таблица «Стоимость»:

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип данных | Ключевое поле | Описание поля |
| #id\_cost | integer | + | Код стоимости |
| \*id\_treatment | integer |  | Код лечения |
| total\_cost | decimal |  | Общая стоимость |

Структурная таблица «Услуга»:

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип данных | Ключевое поле | Описание поля |
| #id\_service | integer | + | Код услуги |
| \*id\_cost | integer |  | Код стоимости |
| cost | decimal |  | Стоимость услуги |
| description | text |  | Описание услуги |

Структурная таблица «Оборудование»:

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип данных | Ключевое поле | Описание поля |
| #id\_equipment | integer | + | Код оборудования |
| \*id\_ treatment | integer |  | Код лечения |
| name | varchar (255) |  | Наименование оборудования |
| type | varchar (255) |  | Тип оборудования |
| appointment | varchar (255) |  | Назначение оборудования |
| count | integer |  | Количество оборудования |
| state | integer |  | Состояние оборудования |

**3.3 Структура входных данных**

Данные, вводимые пользователем при работе с функциями:

1. числовые значения, вводимые с клавиатуры;
2. текстовые данные;

Данные, хранящиеся в базе данных системы:

1. сведения медицинских карт;
2. сведения о клиентах;
3. информация о диагнозах;
4. сведения о врачах;
5. информация об оборудовании;
6. сведения о лечениях;
7. информация о стоимости;
8. информация об услугах.

Врач имеет возможность добавить нового клиента, услугу или совершить другое действие.

**3.5 Структура выходных данных**

Выходные данные программы являются:

1. печать документов:
2. финансового отчета по прибыли;
3. выплат сотрудникам заработной платы и процентных начислений;
4. выполненных врачебных манипуляций;
5. износа оборудования;
6. статистики посещений клиентов;
7. технической документации по состоянию оборудования;
8. медицинских заключений;
9. состояния работников в табеле рабочего времени
10. статистики выполненных услуг.
11. выборка данных о клиентах;
12. подсчёт заработной платы работников;
13. статистика посещений.

**3.6 Описание алгоритмов**

Используемые алгоритмы:

1. Алгоритм выборки - один из способов найти такую запись состоит в том, чтобы отсортировать список в порядке убывания; тогда запись с k-ым по величине значением окажется на k-ом месте. На это уйдет гораздо больше сил, чем необходимо: значения, меньшие искомого, нас, на самом деле, не интересуют. Может пригодиться следующий подход: затем находим наибольшее значение в списке и помещаем его в конец списка. Затем можем найти наибольшее значение в оставшейся части списка, исключая уже найденное. В результате получаем второе по величине значение списка, которое можно поместить на второе с конца место в списке. Повторив эту процедуру К раз, затем найдем k-ое по величине значение.
2. Линейный, последовательный поиск — алгоритм нахождения заданного значения произвольной функции на некотором отрезке. Данный алгоритм является простейшим алгоритмом поиска и, в отличие, например, от двоичного поиска, не накладывает никаких ограничений на функцию и имеет простейшую реализацию. Поиск значения функции осуществляется простым сравнением очередного рассматриваемого значения (как правило, поиск происходит слева направо, то есть от меньших значений аргумента к большим) и, если значения совпадают (с той или иной точностью), то поиск считается завершённым.
3. Алгоритм Стемминга — это процесс нахождения основы слова для заданного исходного слова. Основа слова не обязательно совпадает с морфологическим корнем слова.

**Прочие проектные решения**

Оптимизация в основном фокусируется на одиночном или повторном времени выполнения, использовании памяти, дискового пространства, пропускной способности или некотором другом ресурсе. Увеличение размера программного кэша улучшило производительность времени выполнения.

На основании методов обнаружения ошибок были разработаны следующие средства повышения надёжности ПО.

Средства, использующие временную избыточность: авторизация доступа пользователей к системе, анализ доступных пользователю ресурсов, выделение ресурсов согласно ролям и уровням подготовки пользователей, разграничение прав доступа пользователей к отдельным задачам, функциям управления, записям и полям баз данных.

Средства обеспечения надёжности, использующие информационную избыточность: ссылочная целостность баз данных обеспечивается за счёт системы внутренних уникальных ключей для всех информационных записей системы.

**4 ПЛАН ТЕСТИРОВАНИЯ**

Тестирование программного обеспечения – процесс исследования, испытания программного продукта, имеющий своей целью проверку соответствия между реальным поведением программы и её ожидаемым поведением на конечном наборе тестов, выбранных определенным образом.

Список тестируемых функций:

1. заведение мед. карты;
2. подтверждение оплаты;
3. отметка приема;
4. выбор наилучшего оборудования;
5. оформление документации.

Метод тестирования перечисленных функций – метод «черного ящика», также известное как тестирование, основанное на спецификации или тестирование поведения – техника тестирования, основанная на работе исключительно с внешними интерфейсами тестируемой системы.

Целью этой техники является поиск ошибок в таких категориях:

1. неправильно реализованные или недостающие функции;
2. ошибки интерфейса;
3. ошибки в структурах данных или организации доступа к внешним базам данных;
4. ошибки поведения или недостаточная производительности системы.

Преимущества:

1. тестирование производится с позиции конечного пользователя и может помочь обнаружить неточности и противоречия в спецификации;
2. тестировщику нет необходимости знать языки программирования и углубляться в особенности реализации программы;
3. тестирование может производиться специалистами, независимыми от отдела разработки, что помогает избежать предвзятого отношения;
4. можно начинать писать тест-кейсы, как только готова спецификация.

Недостатки:

1. тестируется только очень ограниченное количество путей выполнения программы;
2. без четкой трудно составить эффективные тест-кейсы;
3. некоторые тесты могут оказаться избыточными, если они уже были проведены разработчиком на уровне модульного тестирования.