МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Российский технологический университет»
МИРЭА

Физико-технологический институт
Кафедра оптических и биотехнических систем и технологий

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА НА ТЕМУ:

«АВТОМАТИЗАЦИЯ КЛЮЧЕВЫХ БИЗНЕС-ПРОЦЕССОВ САНАТОРИЯ НА ОСНОВЕ ИТЕРАЦИОННОЙ МОДЕЛИ ВНЕДРЕНИЯ ИНФОРМАЦИОННЫХ СИСТЕМ»

Студент:
Симанков А.В.

Научный руководитель:
к.т.н., доц. МИРЭА Степанов Д.Ю.

Москва – 2019

АННОТАЦИЯ

Выпускная квалификационная работа на тему «Автоматизация ключевых бизнес-процессов санатория на основе итерационной модели внедрения информационных систем». Данная бакалаврская работа включает в себя введение, шесть разделов, заключение, список использованных литературных источников. Также работа включает 48 рисунков, 11 таблиц, которые наглядно отражают суть выпускной работы.

В разделе «Введение» доказывается актуальность выбранной темы, ставится цель и обозначаются задачи работы. В «Методология внедрения итерационной модели программных продуктов» описывается методология внедрения, ее преимущества и недостатки. Формируется понятие жизненного цикла информационной системы.

В разделе «Идентификация требований, формирование списка требований» формируются пользовательские и функциональные требования. Создается цикл разработки по итерационной модели (4 итерации). В «Проектирование ключевого бизнес-процесса в моделях AS-IS и TO-BE» описываются нотации IDEF0 и DFD. Внедряя эти нотации, проектируются процессы в моделях AS-IS и TO-BE для ключевых бизнес-процессов.

В разделе «Разработка приложения» реализуется создание приложения в среде MS Access в рамках четырех итераций. В «Тестирование разработанного приложения» производится функциональное, интеграционное, нагрузочное тестирование приложения.

В пункте «Расчет затрат на разработку и внедрение программного продукта» производится расчет заработной платы исполнителей, затрат на электроэнергию, амортизацию оборудования, прочих затрат, а также сметы затрат. В «Заключение» подведены итоги проделанной работы, сформированы выводы о работоспособности приложения.

ОГЛАВЛЕНИЕ

Введение

Здоровье граждан является одним из основных приоритетов общественного, экономического и культурного развития любой страны. В крепком здоровье граждан страны – залог роста трудовых резервов, повышения культурного уровня, общенационального и международного ее престижа. В качестве приоритетного направления повышения уровня здоровья граждан в РФ избраны – профилактика и предупреждение заболеваний, качественное медицинское обслуживание граждан и их реабилитация. Трудно переоценить положительный эффект от профилактических мероприятий, лечебных и восстановительных процедур, осуществляемых в заведениях санаторно-курортного типа. Данная практика дает возможность значительно снизить количество возникновения и обострения хронических и иных заболеваний. В связи с чем, все большее количество граждан предпочитают проводить время отдыха в санаторных учреждениях. Помимо применения восстановительных и лечебных свойств и условий природного характера в учреждениях санаторно-курортного типа начали значительное время отводить под процедуры, являющиеся действенными способами системной терапии многих заболеваний, которые оказывают также неоценимый и профилактический эффект. На данный момент большая часть санаторно-курортных учреждений осуществляет хранение медицинской документации (карт) по пациентам в бумажном виде (картотека). Ведение картотеки пациентов на бумажных носителях затрудняет работу персонала по поиску и обработке необходимых данных. Требует значительных затрат по хранению, созданию условий и обеспечению их сохранности. Ведение базы по пациентам учреждения требует ее систематического обновления. Внесение изменений, дополнений и

обновлений в бумажные носители в ручном режиме – трудоемкая процедура, требующая нерационального отвлечения труда сотрудников санаторно-курортного учреждения от непосредственного обслуживания пациентов/клиентов учреждения. Для структуризации всех сведений о клиентах санатория, их диагнозе и процедурах необходимо внедрить специализированные автоматизированные системы. Такой подход даст возможность ощутимо повысить скорость обработки и обмена данными, увеличить качество предоставляемых услуг и сократить количество ошибок, Целью данного проекта является автоматизация ключевых бизнес-процессов санатория на основе итерационной модели внедрения информационных систем и создание таких механизмов, которые позволят упростить трудоемкость работы, ускорить время ее выполнения и сделать процесс работы врача более комфортным, благодаря внедрению автоматизированной системы. Это позволит врачу-курортологу выбрать тот перечень процедур, который необходим пациенту для оздоровления и профилактики. Таким образом, чтобы реализовать вышеупомянутую цель нужно реализовать следующие задачи:

- детальный анализ итерационной методологии внедрения систем;
- идентификация требований и формирование бэклога;
- проектирование процессов и оргструктуры в моделях AS-IS и TO-BE нотации IDEF0 и DFD до 3-4 уровней детализации.

Для каждой итерации требуется выполнить:

- моделирование разрабатываемых пользовательских интерфейсов;
- проектирование структуры данных и нормализация таблиц данных;
- реализация операции ключевого процесса в среде MS Access;
- тестирование и количественная оценка результатов тестирования;
- подготовка блок-схемы алгоритма работы программы.

Раздел 1. Методология внедрения итерационной модели программных продуктов

1.1. Формулировка жизненного цикла и его модели

Понятие жизненного цикла (Life Cycle) информационной системы (ИС) подразумевает основополагающую базу методологии проектирования информационных систем. Жизненный цикл можно представить, как последовательность решений и операций, которые устанавливают путь развития любой системы, программы или иного проекта, начиная с принятия решения о создании ИС, этапов работы над ее созданием, тестированием, внедрением в эксплуатацию и заканчивая окончательным завершением ее применения и отказом от ее использования. А также показывающие происходящие с системой в процессе создания и использования изменения.

Модель жизненного цикла информационной системы – это определенная упорядоченная структура, означающая не только последовательность выполнения процессов, целей и действий, происходящих в процессе эксплуатации, но и определяющая взаимосвязанность между процессами, целями и действиями. Модель жизненного цикла – структура, состоящая из процессов, работ и задач, включающих в себя разработку, эксплуатацию и сопровождение программного продукта, охватывающая жизнь системы от установления требований к ней до прекращения ее использования [1].

1.2. Понятие итерационная модель, ее основные черты и качества специфика

Итеративная модель представляет собой процесс разработки программного обеспечения, реализуемый небольшими этапами, в ходе каждой из которых исследуются и анализируются полученные итоги работы,

выдвигаются новые требования и вносятся исправления в более ранние стадии работы.

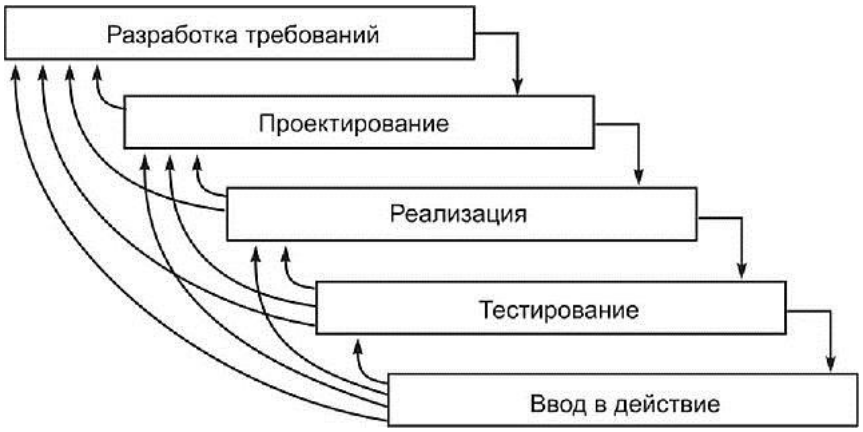


Рисунок 1.1 – *Итерационная схема внедрения информационных систем*

Жизненный цикл проекта при итерационной разработке разделен на поочередные итерации, при этом каждая из них включает в себя все процессы разработки ПО:

- определение и спецификация требований;
- проектирование;
- непосредственную реализацию;
- тестирование;
- введение в эксплуатацию.

При этом в течение одной итерации происходит разработка не проекта в целом, а только отдельного составляющего элемента. Каждая итерация преследует цель разработки и получения в качестве результата такой версии программного продукта, который обладает функционалом, полученным на всех прежних стадиях, и при этом приобретает новый, который был разработан в ходе последней итерации. В каждой итерации необходимо принимать во внимание самые значимые риски и осуществлять требования, которым присвоен наиболее высокий приоритет. Когда разрабатываемые

элементы приходят к уровню, который соответствует запросам заказчика, их включают в единый проект, после чего происходит его общая сборка и тестирование. Таким образом, функциональность разрабатываемого программного продукта увеличивается по ходу выполнения итераций, общая цель которых устанавливается на стадии планирования. Сроки разработки и бюджет, которые требуются для создания окончательной версии продукта, как правило, не оговариваются заранее, поскольку невозможно сразу установить весь будущий объем работы и требования заказчика могут меняться в процессе разработки [2]. Использование инкрементного подхода в процессе создания проекта имеет ряд положительных сторон:

- применение данного подхода избавляет от необходимости сразу расходовать весь объем средств, который требуется для реализации проекта в целом (это связано с тем, что на первых этапах осуществляется разработка главных возможностей ПО или таких, которые относятся к группе повышенного риска);
- в результате выполнения каждого инкремента получается функциональный продукт;
- данный подход позволяет заказчику контролировать разработку каждой последующей версии продукта и по ходу вносить новые требования и предложения;
- разделение появляющихся трудностей на элементы, посредством чего устраняются большие объемы требований, которые ставятся перед командой разработчиков;
- данный подход позволяет непрерывно стимулировать прогресс в процессе реализации разработки;
- сокращается вероятность возникновения рисков или перемен в запросах заказчика;

- инкрементная модель позволяет заказчикам идентифицировать наиболее значимые и приоритетные функциональности программного продукта на самых первых стадиях создания;
- при использовании данного подхода есть возможность распределения рисков на ряд меньших инкрементов;
- растет понимание требований для более поздних версий, благодаря тому, что пользователь имеет возможность ознакомиться с более ранними версиями продукта;
- применение поочередных инкрементов дает возможность консолидировать наработанный опыт в виде оптимизированного продукта, израсходовав при этом гораздо меньшее количество ресурсов, чем было бы необходимо для реализации повторной разработки проекта;
- запросы заказчика более эффективно регулируются, так как период создания каждого инкремента не занимает много времени;
- заказчик последовательно вникает в суть разрабатываемого проекта.

Если инкрементная модель применяется для разработки проекта, но при этом не является подходящей для него в полной мере, она может проявить ряд отрицательных качеств:

- установку полной функциональной системы необходимо производить на первой стадии жизненного цикла, чтобы обеспечить определение инкрементов;
- заказчику необходимо понимать, что суммарные расходы на реализацию проекта не будут сокращены;
- требуется четкое определение интерфейсов, так как разработка ряда составных элементов будет окончена на много раньше других;

- есть вероятность возникновения такого явления, как откладывание решения более сложных вопросов на потом, с целью показать руководству достигаемые на первых стадиях успехи.

Целесообразность использования в разработке итерационной модели можно проследить по таким условиям:

- если большая часть требований может быть выражена на первых этапах разработки, но при этом их возникновение предполагается через какое-то время;
- есть необходимость в быстрые сроки создать продукт с основными базовыми качествами, для его выставления для продажи;
- для долговременных проектов, для реализации которых необходим большой период (год и более);
- когда при анализе рисков, финансирования, графика реализации проекта, масштабов программы, уровня ее сложности или возникновения потребности в ее осуществления на ранних этапах выясняется, что наиболее эффективным способом будет использование принципа поэтапного создания;
- при реализации проекта с использованием новых методик, что дает возможность пользователю приспосабливаться к программному продукту посредством осуществления менее крупных инкрементных операций, не приступая сразу к использованию основного функционала [3].

За последовательность итераций должны быть реализованы такие действия, как:

- анализ требований;
- моделирование ключевого бизнес-процесса с помощью нотаций IDEF0 и DFD (разработка структуры, декомпозиция компонентов);

- реализация требований (разработка программы в среде MS Access): моделирование пользовательского интерфейса, проектирование структуры данных и нормализация таблиц данных, реализация ключевого бизнес-процесса;
- тестирование программы (анализ программного средства и сопутствующей документации с целью выявления дефектов и повышения качества продукта);
- исправление ошибок;
- демонстрация прототипа программы заказчику.

Раздел 2. Идентификация требований, формирование списка требований

2.1. Идентификация требований

Определение первичных требований к разрабатываемому ПО представляет собой процесс, который устанавливает качества и свойства нового программного продукта согласно требованиям заказчика, которые основываются на анализе рыночных условий и динамики их развития. По ходу выполнения данного процесса компания, на которую возложена ответственность за планирование и проектирование приложения, определяет наиболее оптимальные направления для реализации проекта. Данные направления могут соответствовать определенным свойствам и качествам программного обеспечения, которое производит конкурирующая компания, могут соответствовать существующему рыночному спросу или иметь определенные свойства, которые конечный пользователь желает добавить к приложению, уже используемому им. Требования к продукту определяются согласно параметрам, обеспечивающим их адекватность и верную интерпретацию, толкование или перевод с языка пользователя. Данные требования касаются не только функциональные качества продукта и его визуального оформления, но также качество, стабильность работы, безопасность, соответствие требованиям нормативно-правовых актов, а также разного рода нормам и стандартам, для удовлетворения требований и запросов заказчика в полной мере. В процессе определения первичных требований к разрабатываемому продукту, нужно детальное изучение каждого компонента на этапе тестирования приложения. Документация, регламентирующая данные требования, должна создаваться заказчиком в неразрывной коммуникации с ним.

Первичные требования к разрабатываемому ПО закрепляются в документации функциональной спецификации и отображают главные свойства и качества продукта, возможности его взаимодействия с приложениями от других производителей, дают возможность сравнивать его с продуктами конкурирующих фирм, анализировать информацию о состоянии рынка, стоимости, цене, масштабы производства, временных требований. Функциональная спецификация продукта – это справочная информация, на основе которой все фирмы строят свое подробное описание продукта, спецификации предоставляемых услуг и сервиса. Они всегда устанавливают задачи и уровень качества, который необходимо сравнивать с полученным, если есть такая возможность, или со сведениями о продуктах-аналогах (собственного производства или произведенных конкурирующими фирмами). В финале процесса определения функциональных спецификаций происходит утверждение обзора качества. В ходе реализации данного процесса присутствуют и принимают участие все подразделения организации, которые будут заниматься разработкой данного продукта:

- полноту и законченность описания продукта;
- адекватность процесса идентификации требований к продукту;
- адекватность результата, то есть то, что продукт соответствует поставленной цели [4].

Анализ требований – элемент процесса создания программного продукта, который состоит в получении требований к продукту, их обработке и фиксации на материальных носителях. В ходе получения требований необходимо учитывать вероятность того, что отдельные требования разных заинтересованных субъектов могут противоречить друг другу [5]. Полное и качественное исследование требований имеет основополагающее значение для успеха всего проекта. Требования, предъявляемые к продукту должны

быть осуществимые, проверяемые, детализированные, зафиксированные на материальных носителях. Анализ требований включает три типа деятельности:

- сбор требований – общение с клиентами и пользователями, чтобы определить, каковы их требования; анализ предметной области;
- анализ требований – определение, являются ли собранные требования неясными, неполными, неоднозначными или противоречащими; решение этих проблем; выявление взаимосвязи требований;
- документирование требований – требования могут быть задокументированы в различных формах, таких как простое описание, сценарии использования, пользовательские истории, или спецификации процессов.

Анализ требований может представлять из себя долгий и трудоемкий процесс. Новые системы преобразовывают окружающую действительность и взаимоотношения между людьми, следовательно необходимо установить всех заинтересованных субъектов, учесть все их запросы и обеспечить осознание ими важности новых систем [6].

2.2. Пользовательские требования

Пользовательские требования – это требования, которые формулируются пользователями к конечному продукту. Эти требования представляют собой задачи и идеи, которые пользователям дадут возможность для комфортного пользования разработанным программным средством. Ниже определены требования, предъявляемые пользователями к проектируемой системе:

- База данных должна хранить:

- личные данные о пациенте;
- данные о диагнозе;
- данные о проводимых процедурах;
- данные о врачах.

- Установка процедуры по диагнозу. Автоматический выбор процедуры по определенному диагнозу.
- Возможность осуществлять действия с электронной медицинской картой:
  - добавление (создание новой записи о пациенте);
  - редактирование (изменение некорректной информации);
  - удаление (удаление неактуальных данных или полное удаление информации о пациенте);
- Печатать и выводить эпикриз с назначенными процедурами.
- Осуществлять поиск пациента.
- Программа должна иметь возможность составлять отчеты по необходимым категориям и за определенное время.
- Легкость в освоении медицинским персоналом (простота и легкость в пользовании интерфейса).
- Вся информация хранится на персональных компьютерах санатория.

2.3. Функциональные требования

Функциональные требования – это требования, объясняющие, что должно быть в разрабатываемом ПО, а также, какие действия должны выполняться системой:

- База данных должна включать в себя таблицы:

- «Пациенты». Содержится информация о ФИО пациента, дате рождения, пол и др.;
- «Диагноз». Содержится информация о диагнозе;
- «Процедуры». Содержится информация о процедуре, количестве, кратности, дате начала и окончания процедуры, а также комментарий для работника процедурного кабинета;
- «Врачи». Содержится информация о враче (ФИО и др.).

- Автоматизировать выбор процедуры по поставленному диагнозу.
- Создать возможность вывода на экран электронной карточки пациента, в которой можно внести или изменить данные о пациенте.
- Обеспечивать:
  - создание новых данных в таблицах «Пациенты», «Диагноз», «Процедуры», «Врачи»;
  - изменение данных в таблицах «Пациенты», «Диагноз», «Процедуры», «Врачи»;
  - удаление данных в таблицах «Пациенты», «Диагноз», «Процедуры», «Врачи».
- Выводить данные на экран монитора:
  - полная личная информация о пациенте;
  - информация о диагнозе;
  - информация о процедурах;
  - информация о процедурах для диагноза.
- Пользовательский интерфейс программы должен обеспечивать наглядное, интуитивно понятное представление структуры размещенной на нем информации, быстрый и логичный переход к разделам и страницам.

- Печать отчета, содержащего личные данные пациента и список проведенных процедур.

2.4. Приоритизация требований

Важнейшей стадией работы является приоритизация требований. После того, как осуществлена оценка запросов пользователей важно установить, в какой очередности необходимо осуществлять выполнение задач, какой функционал должен присутствовать в первоначальной версии приложения, что должно быть в следующих сборках и чем можно пренебречь до последнего этапа, на тот случай если в конце еще останется время.

Установить определенную очередность выполнения целей можно, основываясь на разных параметрах и отталкиваясь от разных соображений. С этой целью было выработано большое количество методик приоритизации, каждая из которых основывается на разных параметрах важности цели. Применим модель Кано (Kano model). Главная задача данной модели состоит в гарантировании удовлетворенности требований заказчика и конечного пользователя, иными словами это одно из средств регулирования уровня качества. Цель данной модели состоит в установке всех существующих требований и расстановке их в порядке убывающего приоритета. Его природа состоит в дифференциации требований на четыре группы, отображающие степень значимости каждой возможности программы для пользователя:

- необходимые (must-be) – при их отсутствии пользователь не рассматривает данный продукт. Однако их наличие не приводит к высокой удовлетворенности пользователя, поскольку воспринимается им как нечто само собой разумеющееся;

- одномерные (one-dimensional) – чем в большей степени такие характеристики присутствуют или лучше их качество, тем выше удовлетворенность ими пользователя;
- привлекательные (attractive) – отсутствие данного атрибута воспринимается пользователям нейтрально и не препятствует выбору, тем не менее его присутствие резко увеличивает удовлетворенность потребителя;
- безразличные (indifferent) – эти характеристики не имеют значения для пользователя и не влияют на его выбор [7].

2.5. Список требований

Полноценное представление о функциональных возможностях продукта – полный список требований. Это таблица, которая связывает требования к создаваемому продукту. В ней сверяются пользовательские и функциональные требования, происходит представление программного компонента создаваемой системы к каждому из требований, выстраивается приоритизация.

Таблица 2.1 – *Список требований*

№	Пользовательское требование	Программный компонент	Функциональное требование	Приоритет требования согласно модели Кано
1	Хранение личных данных пациента, диагноз, процедуры, врачи	Программа для работы с БД	Таблица данных «Пациенты», «Процедуры», «Диагноз» включающая сведения о пациенте, его диагнозе, процедурах и датах проведения. Таблица данных «Врачи», содержащая сведения о лечащем враче	Необходимые (Must be)

№	Пользовательское требование	Программный компонент	Функциональное требование	Приоритет требования согласно модели Кано
2	Автоматический выбор процедуры		Список всех возможных диагнозов, из которого будут выбираться определенные процедуры	
3	Добавление редактирование, удаление данных пациента, врача, диагноза, процедуры		Функция исправления удаления и внесения новой информации о пациенте в таблицу данных «Пациенты», «Процедуры», «Диагноз». Добавление нового персонала в таблицу данных «Врачи»	Одномерные (one-dimensio-nal)
4	Создание электронной медицинской карты		Функция создания, редактирования личной информации в электронной медицинской карте	
5	Просмотр записей диагнозы пациентов, процедуры пациентов, врачи	Программа по выводу на экран запрашиваемых данных	Функция просмотра данных из БД «Пациенты», «Процедуры», «Диагноз», «Врачи»	Привлекательн ые (attractive)
6	Вывод эпикриза с назначенными процедурами		Функция просмотра данных из БД «Пациенты», «Процедуры», «Диагноз», «Врачи» (вывод медицинской карты на экран)	
7	Осуществление поиска пациента	Программа по поиску	Возможность поиска записей в таблице	Безразличные (indifferent)
8	Легкость в освоении медицинским персоналом	Разрабатываемое ПО	Функция доступности (общепонятность, простота, легкость)	
9	Печать эпикриза	Программа вывода информации	Функция печати отчета на бумажный носитель	

При создании программного продукта по итерационной модели рекомендуется производить разработку по следующим шагам.

Таблица 2.2 – *Циклы разработки итерационной модели*

Название цикла	Содержание
Начало	▪ анализ требований (см. табл. 2.1) ▪ моделирование ключевого бизнес-процесса с помощью нотаций IDEF0 и DFD ▪ моделирование данных и экранов программы
Итерация 1	▪ составление плана данного цикла (выполнить требования 1-2 в MS Access ▪ реализация требований 1-2 в MS Access ▪ тестирование текущего потенциала программы ▪ исправление ошибок ▪ демонстрация прототипа программы заказчику (обратная связь)
Итерация 2	▪ составление плана данного цикла (выполнить требования 3-4 в MS Access) ▪ реализация требований 3-4 в MS Access ▪ тестирование текущего потенциала программы ▪ исправление ошибок ▪ демонстрация прототипа программы заказчику (обратная связь)
Итерация 3	▪ составление плана данного цикла (выполнить требования 5-6 в MS Access) ▪ реализация требований 5-6 в MS Access; ▪ тестирование текущего потенциала программы ▪ исправление ошибок ▪ демонстрация прототипа программы заказчику (обратная связь)
Итерация 4	▪ составление плана данного цикла (выполнить требования 7-9 в MS Access) ▪ реализация требований 7-9 в MS Access ▪ тестирование текущего потенциала программы ▪ исправление ошибок ▪ демонстрация прототипа программы заказчику (обратная связь)
Окончание	▪ исправление ошибок и финальная подготовка к релизу ▪ тестирование финальной версии программы ▪ выход в релиз конечного продукта

Раздел 3. Проектирование ключевых бизнес-процессов в моделях «AS-IS» и «TO-BE»

Моделирование – это замена реального объекта, процесса, явления его определенным аналогом, который конструктивно проще своего оригинала, но при этом в нем отображаются все специфические черты с позиции задачи моделирования, что может содействовать в исследовании «исходника». Моделирование используется в различных областях жизнедеятельности, в частности в экономике.

Моделирование бизнес-процесса – процесс, который заключается в отображении рассмотрения хода работ в построении формальной модели, которая складывается из связанных между собой действий. Моделирование бизнес-процессов дает возможность исследовать не только процесс деятельности организации в общем, характер его коммуникаций с другими организациями и субъектами, но также как построен механизм деятельности на каждом отдельном рабочем месте.

Задача проектирования состоит в упорядочивании совокупности сведений об организации и ее бизнес-процессах в графической форме в виде графиков, которая является более простой для исследования и анализа полученных данных.

Подробное проектирование бизнес-процессов осуществляется в такой модели, которая может в достаточной мере реализовать необходимую конкретизацию и гарантировать единое понимание природы деятельности компании. Этот подход дает возможность привести в порядок происходящие в настоящее время процессы, а также применяемые информационные объекты. Отталкиваясь от этого, определяются узкие места в компании и

коммуникации бизнес-процессов, устанавливается, требуются ли какие-либо преобразования в действующей структуре.

Модель «AS-IS» нередко подразумевается в качестве функциональной и реализуется с применением разного рода графических нотаций. На стадии проектирования AS-IS необходимо создавать наиболее близкую к реальности модель, которая базируется на фактических процессах, а не на их идеальных копиях. Проектирование информационных систем и управление процессами подразумевает построение модели AS-IS и дальнейший переход к модели TO-BE, что является залогом автоматизации "правильных", усовершенствованных процессов.

Процесс моделирования какой-либо системы в IDEF0 начинается с определения контекста, т.е. наиболее абстрактного уровня описания системы в целом. Сюда нужно отнести установку субъекта, осуществляющего моделирование, целей и позиции относительно проекта. Под субъектом предполагается сама система. При этом нужно четко определить, что в нее входит, а что находится вне ее границ. Иными словами необходимо установить элементы системы и внешние связи. На установление субъекта системы значимым образом оказывает воздействие точка зрения, с которой рассматривается система и цели моделирования – вопросы, на которые сформированный проект должен ответить. То есть, на первом этапе нужно выяснить сферу проектирования. Характеристика сферы как системы, а также ее элементов служит фундаментом для формирования проекта.

Обычно сначала строится модель существующей организации работы – AS-IS (как есть). На основе модели AS-IS можно прийти к определенному балансу между разными структурными подразделениями организации относительно их вклада в общее дело организации и что каждая единица

бизнеса добавляет в процесс. Модель AS-IS позволяет выяснить, "что мы делаем сегодня" перед тем, как перейти на то, "что мы будем делать завтра".

Анализ функциональной модели позволяет понять, где находятся наиболее слабые места, в чем будут состоять преимущества новых бизнес-процессов и насколько глубоким изменениям подвергнется существующая структура организации бизнеса.

Детализация бизнес-процессов позволяет выявить недостатки организации даже там, где с виду работа налажена должным образом. Признаками неэффективной деятельности могут быть безрезультатные, нерегулируемые и дублируемые процессы, некачественный документооборот, отсутствие налаженных обратных связей в системе управления организации.

Найденные в модели AS-IS недостатки можно исправить при создании модели TO-BE (как будет) – модели, которая отображает новое построение бизнес-процессов. Модель TO-BE необходима для нахождения иных, более оптимальных методов организации и отображения в документации проектов будущей деятельности фирмы [8].

3.1. Способы проектирования

IDEF0 – методология функционального моделирования, снабженная наглядным графическим языком и позволяющая представить моделируемый процесс в виде набора взаимосвязанных функций. Моделирование средствами IDEF0 является первым этапом изучения бизнес-процесса.

Функциональные модели выделяют действия посредством представления в виде специального элемента – функционального блока. Блок, изображающий некоторую бизнес-функцию, является центральным

элементом нотации IDEF0. Элементы, используемые в нотации IDEF0 представлены в таблице 3.1.

Таблица 3.1 – *Условные обозначения нотации IDEF0*

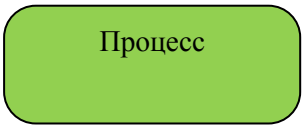
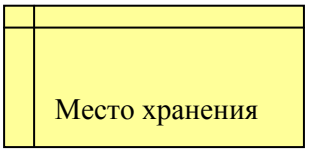
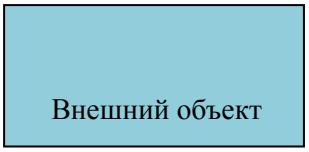
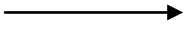
Графический элемент	Описание
A1. Процесс	Функциональный блок. Представляет собой некоторый конкретный процесс в рамках рассматриваемой системы
Вход →	Вход, входящие в левую грань стрелки изображают данные или объекты, изменяемые в ходе выполнения бизнес-процесса
→ Выход	Выход, выходящие из правой грани стрелки изображают данные или объекты, появляющиеся в результате выполнения бизнес-процесса
Управление ↓	Управление, входящие в верхнюю грань стрелки изображают правила и ограничения, согласно которым выполняется бизнес-процесс
Механизм ↑	Механизм, входящие в нижнюю грань стрелки изображают ресурсы, необходимые для выполнения бизнес-процесса, но не изменяемые им

Основным преимуществом данной методологии является возможность описания бизнес-процессов верхнего уровня и возможности отображения регулирования процессов, обратных связей и потоков данных. Среди слабых сторон следует отметить трудность для понимания диаграмм, множество уровней декомпозиции, которое необходимо для полной характеристики процесса, сложность связывания нескольких процессов. Для этих целей используют нотацию DFD (Data Flow Diagram – диаграмма потока данных) [9].

Цель представления DFD нотации — продемонстрировать, как каждый процесс преобразует свои входные данные в выходные. Может отражать не

только информационные, но и материальные потоки. Также как и в других моделях, поддерживается декомпозиция. Диаграмма потока данных является основным средством моделирования функциональных требований к системе – проектируемой или реально существующей. Главной сильной стороной этой нотации являются: возможность описания процессов нижнего уровня, требуемой для разрешения проблемы логической недоработанности модели и формирования полной функциональной спецификации для разрабатываемой системы, возможность моделирования сверху вниз, возможность нотации в точности установить внешние структуры, применяя анализ потоков данных в пределах и снаружи системы [10]. Элементы, используемые в процессе моделирования с помощью нотации DFD, приведены в таблице 3.2 [1].

Таблица 3.2 – *Условные обозначения нотации DFD*

Графический элемент	Описание
	Исходный процесс. Преобразование входных потоков данных в выходные в соответствии с определенным алгоритмом
	Место хранения информации. Структура для хранения информационных объектов
	Внешний объект. Это материальный предмет или физическое лицо, представляющее собой приемник или источник информации
Данные 	Данные. Поток данных определяет информацию, передаваемую через некоторое соединение от источника к приемнику

Основным недостатком данной модели является невозможность анализа временных промежутков в процессе преобразования данных, необходимость ввода управляющих процессов.

3.2. Проектирование процессов в AS-IS с помощью IDEF0 и DFD

Первый уровень описания процесса врача-курортолога в модели AS-IS с помощью нотации IDEF0 заключается в выполнении трех процессов: A1 – «Принять пациента», A2 – «Провести лечение», A3 – «Выписать пациента». Эти процессы показаны на рисунке 3.1.

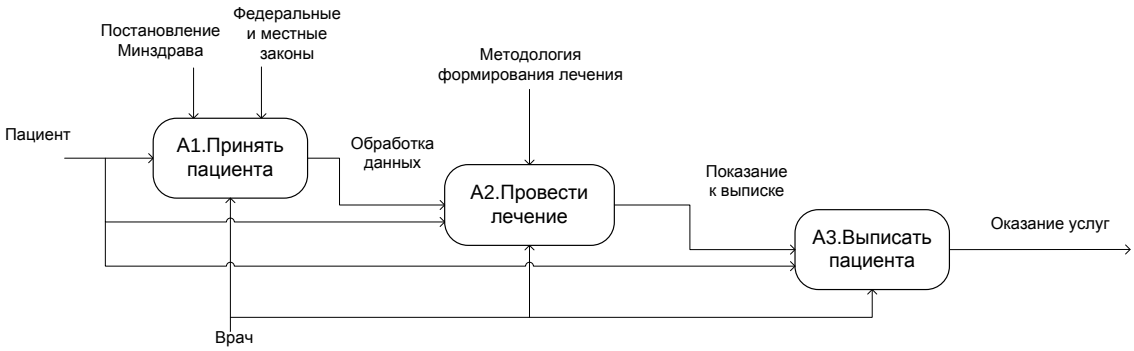


Рисунок 3.1 – *Первый уровень описания ключевых бизнес-процессов в AS-IS нотации IDEF0*

Выполним декомпозицию процессов A1 и A3. Таким образом, получим второй уровень описания процессов IDEF0 (рис. 3.2-3.3).

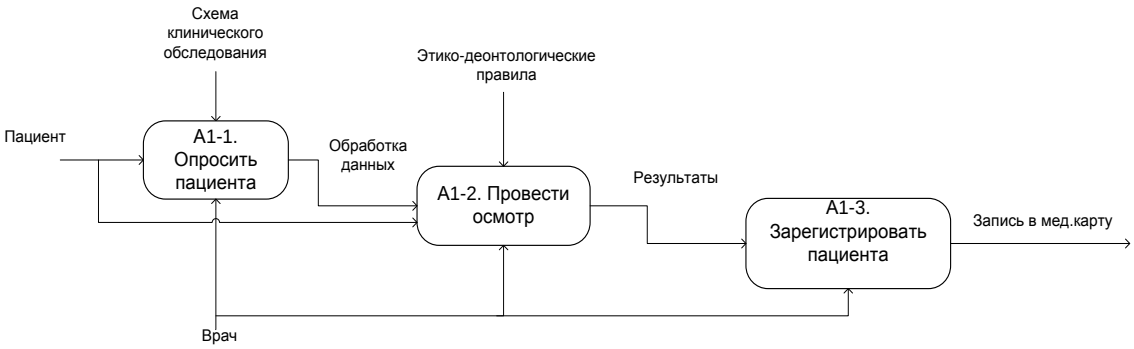


Рисунок 3.2 – *Бизнес-процесс A1 «Принять пациента» на втором уровне описания в AS-IS нотации IDEF0*

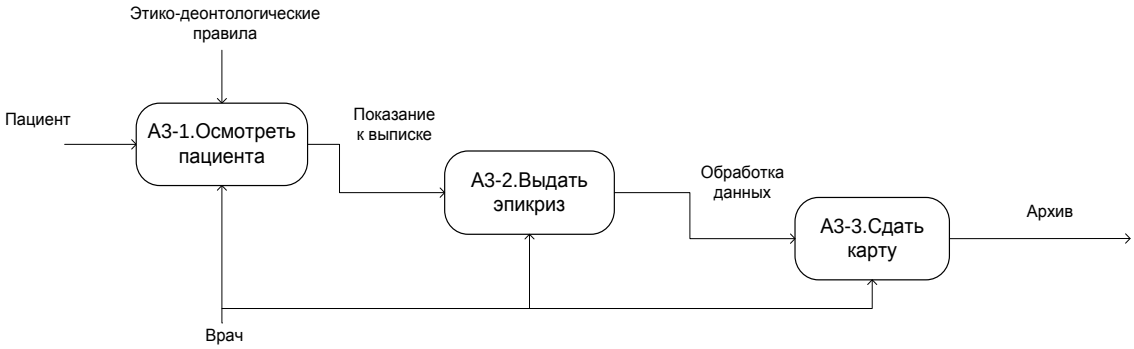


Рисунок 3.3 – *Бизнес-процесс А3 «Выписать пациента» на втором уровне описания в AS-IS нотации IDEF0*

Проведем декомпозицию процесса А2 «Провести лечение», получив второй уровень описания процессов нотации IDEF0 (рис. 3.4).

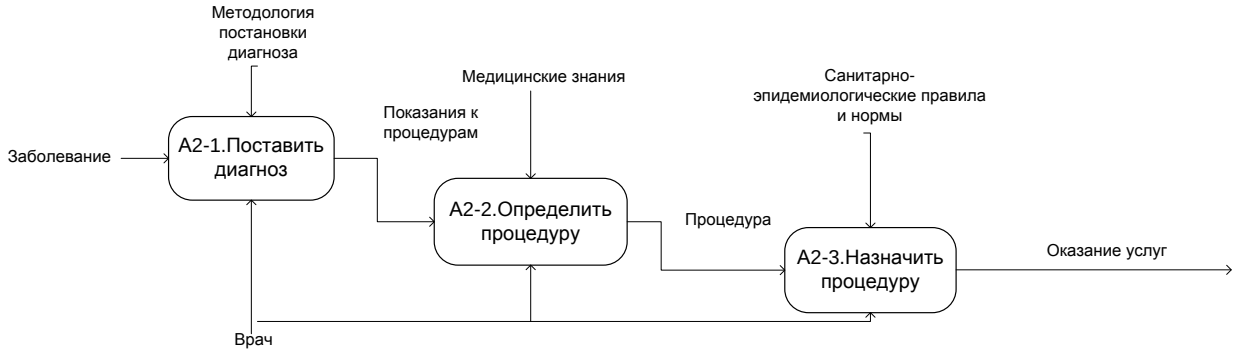


Рисунок 3.4 – *Бизнес-процесс А2 «Провести лечение» на втором уровне описания в AS-IS нотации IDEF0*

Второй уровень описания не несет в себе информации, поэтому следует рассмотреть подпроцесс А2-2 «Определить процедуру» более подробно. Выполнение декомпозиции второго уровня описания с помощью IDEF0 невозможно, так как в дальнейшем вовлекаются хранилища данных. Поэтому целесообразно использовать нотацию DFD. Результат декомпозиции показан на рисунке 3.5.

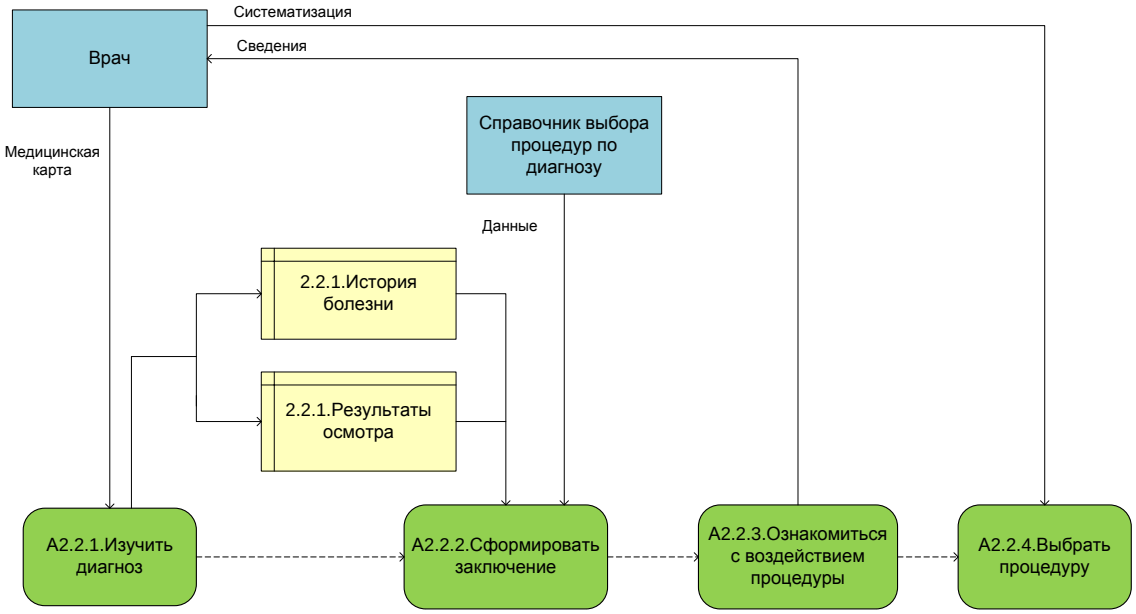


Рисунок 3.5 – *Бизнес-процесс A2-2 «Определить процедуру» на третьем уровне описания в AS-IS нотации DFD*

3.3. Проектирование процессов в TO-BE с помощью IDEF0 и DFD

После рассмотрения структуры процессов в нотации AS-IS, необходимо рассмотреть структуры процессов в модели TO-BE. На рисунке 3.6 изображены процессы в модели «Как будет» нотации IDEF0.

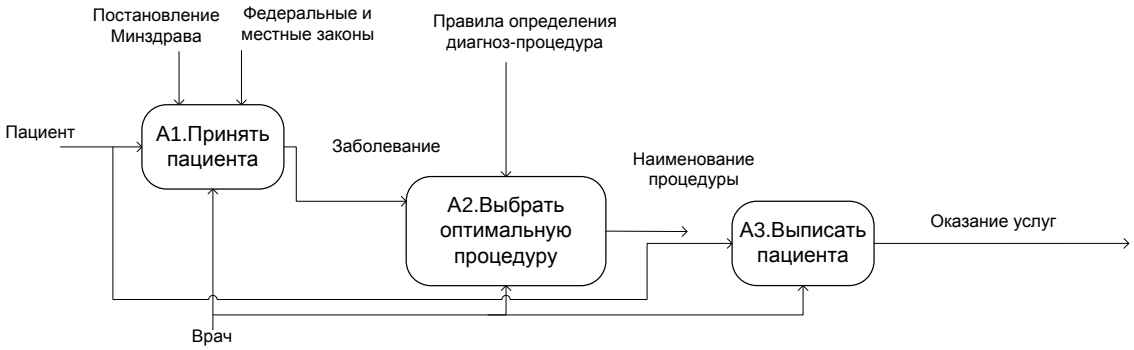


Рисунок 3.6 – *Первый уровень описания ключевых бизнес-процессов в TO-BE нотации IDEF0*

Подобным образом произведем декомпозицию процессов A1 и A3 (рис. 3.7-3.8).

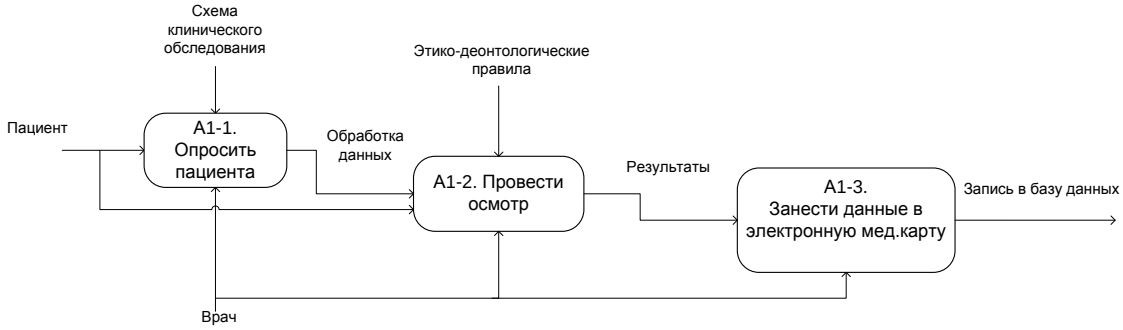


Рисунок 3.7 – *Бизнес-процесс A1 «Принять пациента» на втором уровне описания в TO-BE нотации IDEF0*

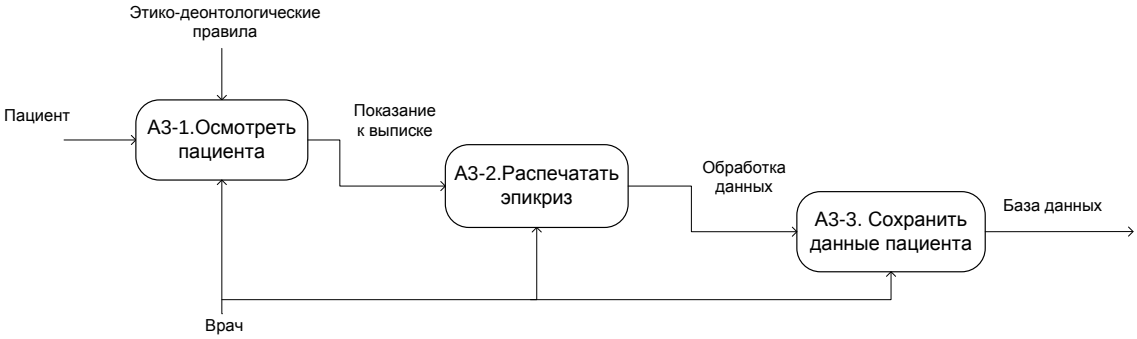


Рисунок 3.8 – *Бизнес-процесс A3 «Выписать пациента» на втором уровне описания в TO-BE нотации IDEF0*

Аналогично, для более подробного рассмотрения подвергнем процесс A2 «Выдать оптимальную процедуру» декомпозиции (рис. 3.9).

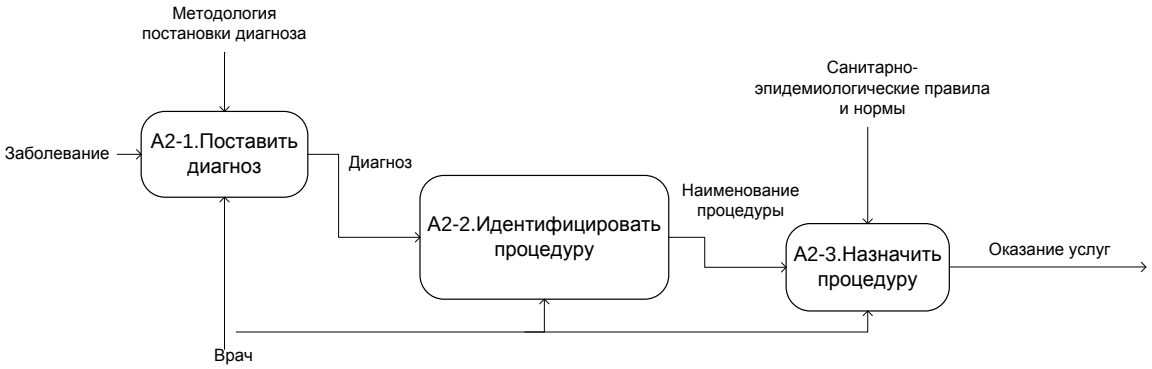


Рисунок 3.9 – *Бизнес-процесс A2 «Выдать оптимальную процедуру» на втором уровне описания в TO-BE нотации IDEF0*

Выполнение декомпозиции второго уровня не представляется возможным из-за фигурирования хранилища данных. Поэтому применим

нотацию DFD. Графическое представление декомпозиции второго уровня представлено на рисунке 3.10.

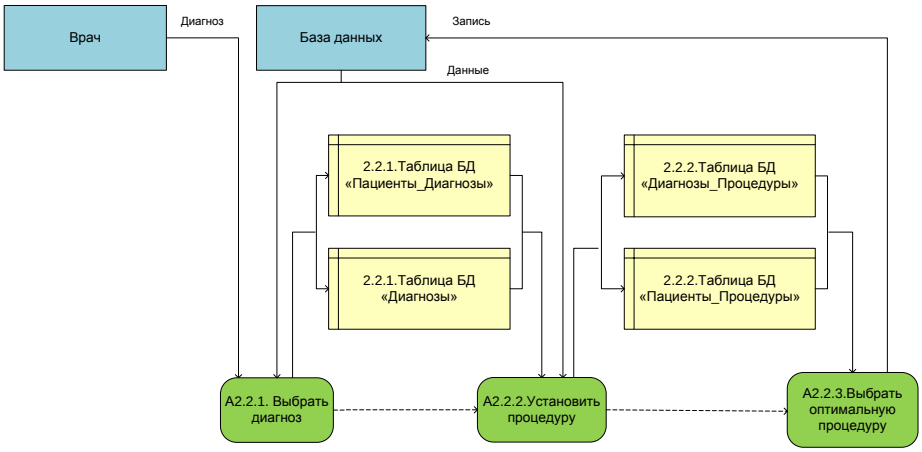


Рисунок 3.10 – *Бизнес-процесс А2-2 «Идентифицировать процедуру» на третьем уровне описания в TO-BE нотации DFD*

Используя описанные ранее бизнес-процесс на 1-3 уровнях, построим карту процессов в нотации ARIS и модели «Как будет» (рис. 3.11).

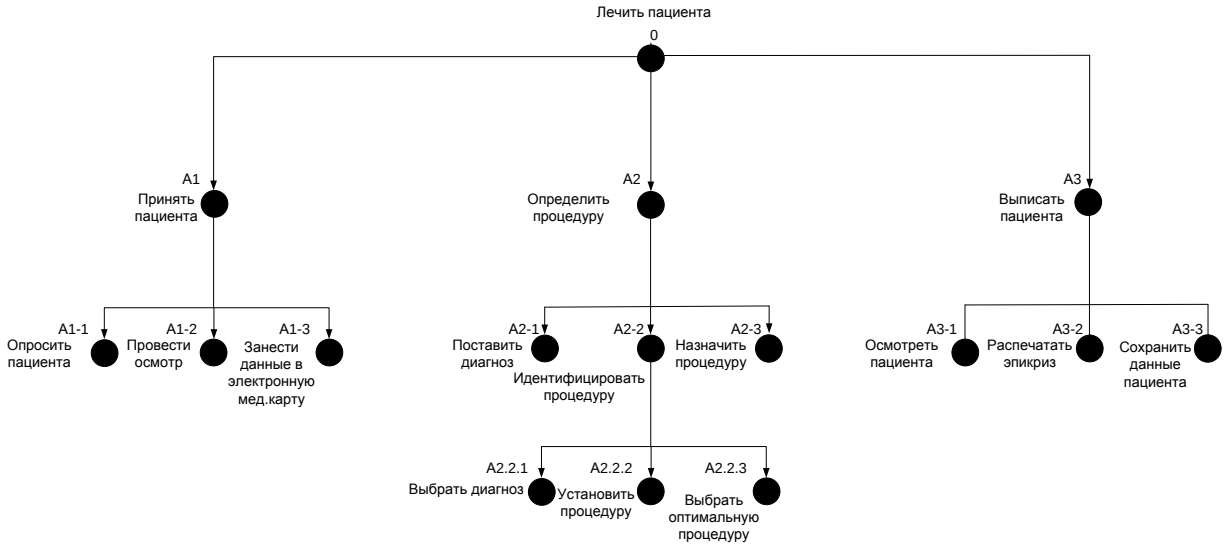


Рисунок 3.11 – *Карта процессов в нотации ARIS модели «Как будет»*

Раздел 4. Разработка приложения

4.1. Моделирование данных разрабатываемой системы

Процесс моделирования основывается на методике нормализации, иными словами декомпозиции связей, которые находятся в предыдущей нормальной форме, в две либо больше связи, которые соответствуют следующей нормальной форме. Данные могут группироваться в таблицы (классы данных) разными способами. При проектировании приложения может использоваться одна универсальная связь, в которую включаются все необходимые атрибуты [11]. Она может содержать все данные, которые предполагается размещать в БД (табл. 4.1).

Таблица 4.1 – *Классы данных*

Название класса	Данные	Тип данных	Размер поля
Пациенты	ФИО	Текстовый	100 символов
	Дата рождения	Дата/время	8 символов
	Пол	Текстовый	4 символа
	Место проживания	Текстовый	255 символов
	Социальный статус	Текстовый	255 символов
	Время пребывания	Дата/время	8 символов
	Врач	Текстовый	100 символов
Диагноз	Название диагноза	Текстовый	50 символов
Процедуры	Название процедуры	Текстовый	80 символов
	Комментарий	Поле MEMO	800 символов

Используем нормализацию данных, чтобы исключить избыточное дублирование данных в базе. Оптимально производить нормализацию до третьей нормальной формы. Согласно первой нормальной форме (1НФ) необходимо:

- определить ключевые поля;
- обеспечить содержание только одного значения на пересечении каждого столбца строки.

Таким образом, зададим ключевые поля для класса «Пациенты», «Диагноз», «Процедуры». Обеспечим содержание только одного значения для каждого атрибута. Для этого в классе «Пациент» разделим поле «ФИО» на три строки: «Фамилия», «Имя», «Отчество»; «Пол», «Место проживания» на строки: «Республика/край/область», «Город», «Район», «Тип местности»; поле «Время пребывания» на строки: «Дата поступления», «Дата выписки», «Проведено дней» (табл. 4.2).

Таблица 4.2 – *Нормализация данных первой нормальной формой*

Название класса	Данные	Тип данных	Размер поля
Пациенты	Код пациента	Счетчик	4 символа
	Фамилия	Текстовый	30 символов
	Имя	Текстовый	30 символов
	Отчество	Текстовый	30 символов
	Дата рождения	Дата/время	8 символов
	Пол	Текстовый	4 символа
	Республика/край/область	Текстовый	20 символов
	Город	Текстовый	30 символов
	Район	Текстовый	30 символов
	Тип местности	Текстовый	10 символов
	Дата поступления	Дата/время	8 символов
	Дата выписки	Дата/время	8 символов
	Проведено дней	Дата/время	3 символа
	Код врача	Счетчик	4 символа
	Фамилия	Текстовый	30 символов
	Имя	Текстовый	30 символов
	Отчество	Текстовый	30 символов
Диагноз	Код диагноза	Счетчик	4 символа

Название класса	Данные	Тип данных	Размер поля
	Название диагноза	Текстовый	50 символов
Процедуры	Код процедуры	Счетчик	4 символа
	Название процедуры	Текстовый	80 символов
	Комментарий	Поле MEMO	800 символов

Произведем нормализацию до второй нормальной формы (2НФ). Для этого необходимо: декомпозировать отношения, в которых неключевые атрибуты будут зависеть от ключа. Следовательно, не ключевые атрибуты ключа «Код пациента» добавляются в промежуточные таблицы: «Пациенты_Диагнозы» (Код_Записи, Код_Пациента, Код_диагноза, Дата), «Пациенты_Процедуры» (Код_Записи, Код_Пациента, Код_Диагноза, Код_Процедуры, Количество, Дата_Начала, Дата_Окончания, Комментарий). Не ключевые атрибуты ключей «Код_Диагноза», «Код_Процедуры» добавляются в промежуточную таблицу: «Диагнозы_Процедуры» (Код_Записи, Код_Диагноза, Код_Процедуры). Данные о врачах вынесем в отдельную таблицу «Врачи» (табл. 4.3).

Таблица 4.3 – *Нормализация данных второй нормальной формой*

Название класса	Данные	Типы данных	Размер поля
Пациенты	Код пациента	Счетчик	4 символа
	Фамилия	Текстовый	30 символов
	Имя	Текстовый	30 символов
	Отчество	Текстовый	30 символов
	Дата рождения	Дата/время	8 символов
	Пол	Текстовый	4 символа
Пациенты_Диагнозы	Код записи	Счетчик	4 символа
	Код пациента	Числовой	2 символа
	Код диагноза	Числовой	2 символа
	Дата	Дата/время	8 символов

Название класса	Данные	Типы данных	Размер поля
Пациенты_Процедуры	Код записи	Счетчик	4 символа
	Код диагноза	Числовой	2 символа
	Код пациента	Числовой	2 символа
	Код процедуры	Числовой	2 символа
	Количество	Числовой	2 символа
	Дата начала	Дата/время	8 символов
	Дата окончания	Дата/время	8 символов
	Комментарий	Поле MEMO	800 символов
Диагноз	Код диагноза	Счетчик	4 символа
	Название диагноза	Текстовый	50 символов
Процедуры	Код процедуры	Счетчик	4 символа
	Название процедуры	Текстовый	80 символов
	Комментарий	Поле MEMO	800 символов
Диагноз_Процедуры	Код записи	Счетчик	4 символа
	Код диагноза	Числовой	2 символа
	Код процедуры	Числовой	2 символа
Врачи	Код врача	Счетчик	4 символа
	Фамилия	Текстовый	30 символов
	Имя	Текстовый	30 символов
	Отчество	Текстовый	30 символов

Для приведения к третьей нормальной форме (3НФ) данные должны соответствовать условиям 2НФ и необходимо, чтобы ни один из его не ключевых атрибутов не связан функциональной зависимостью с любым другим не ключевым атрибутом. Атрибуты, зависящие от других не ключевых атрибутов, нормализуются посредством перемещения зависимого свойства и того, от которого он зависит, в новое отношение. Проведя нормализацию данных, получим архитектуру данных, приведенную к третьей нормальной форме (рис.4.1).

Рисунок 4.1 – *Архитектура данных разрабатываемой системы*

4.2. Описание интерфейса разрабатываемого приложения

Интерфейс пользователя, он же пользовательский интерфейс (UI – англ. user interface) – вид интерфейсов, в котором одна сторона является человеком, а другая компьютером. Представляет собой комплекс приемов и инструментов, посредством которых пользователь осуществляет коммуникацию с разными, как правило, непростыми, совокупностями компонентов, компьютерами и устройствами.

Интерфейс двунаправленный. устройство, получив команды от пользователя и исполнив их, выдаёт информацию обратно, наличествующими у неё средствами (визуальными, звуковыми и т. п.), приняв которую, пользователь выдаёт устройству последующие команды предоставленными в его распоряжение средствами (кнопки, переключатели, регуляторы, сенсоры, голосом, и т. д.) [4]. Интерфейс должен создаваться принимая во внимание параметры качества любого интерфейса: скорость осуществления операций, удобство, уровень производительности, быстрота адаптации и личная удовлетворенность пользователя. Для этого необходимо

создать макет главного экрана приложения, который содержит следующие кнопки: «Справочная информация», «Личные данные пациента», «Поиск по пациентам», «Карточка пациента», «Установить диагноз», «Карта выбывшего из санатория», «Отчёты» (рис. 4.2).

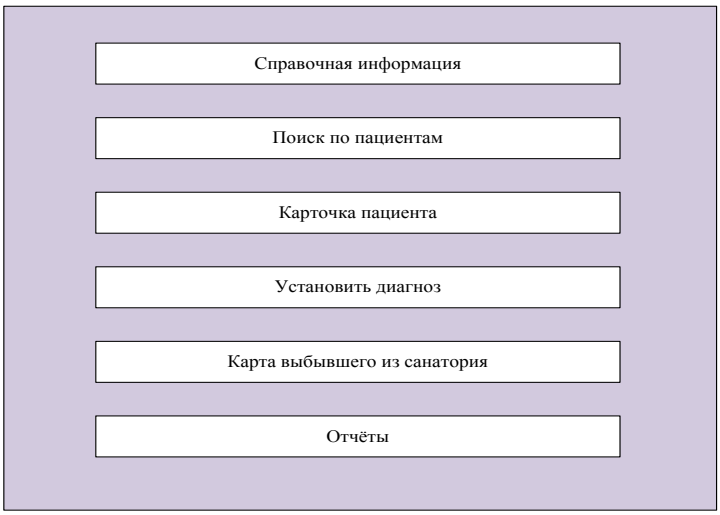


Рисунок 4.2 – *Главный интерфейс приложения*

Перейдя на «Справочную информацию» можно добавить/изменить/удалить записи о диагнозах, процедурах, процедурах для диагноза, а также есть возможность занести информацию о враче (рис. 4.3).

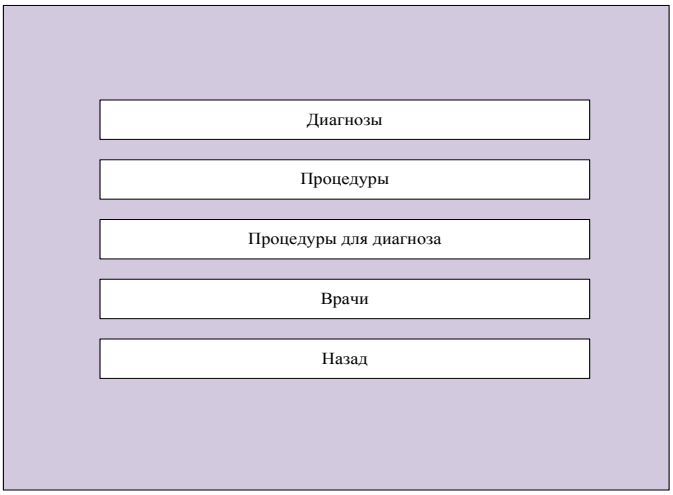


Рисунок 4.3 – *Интерфейс «Справочная информация»*

Если необходимо произвести поиск пациента, достаточно перейти по кнопке «Поиск по пациентам» (рис. 4.4).

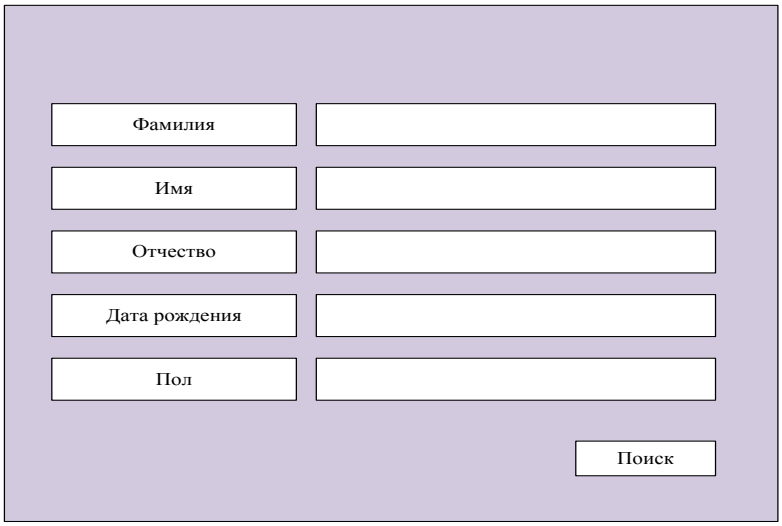


Рисунок 4.4 – *Интерфейс «Поиск по пациентам»*

Чтобы добавить нового пациента, пользователю достаточно выбрать «Карточка пациента» и перейти в таблицу добавления записи о новом пациенте (рис.4.5).

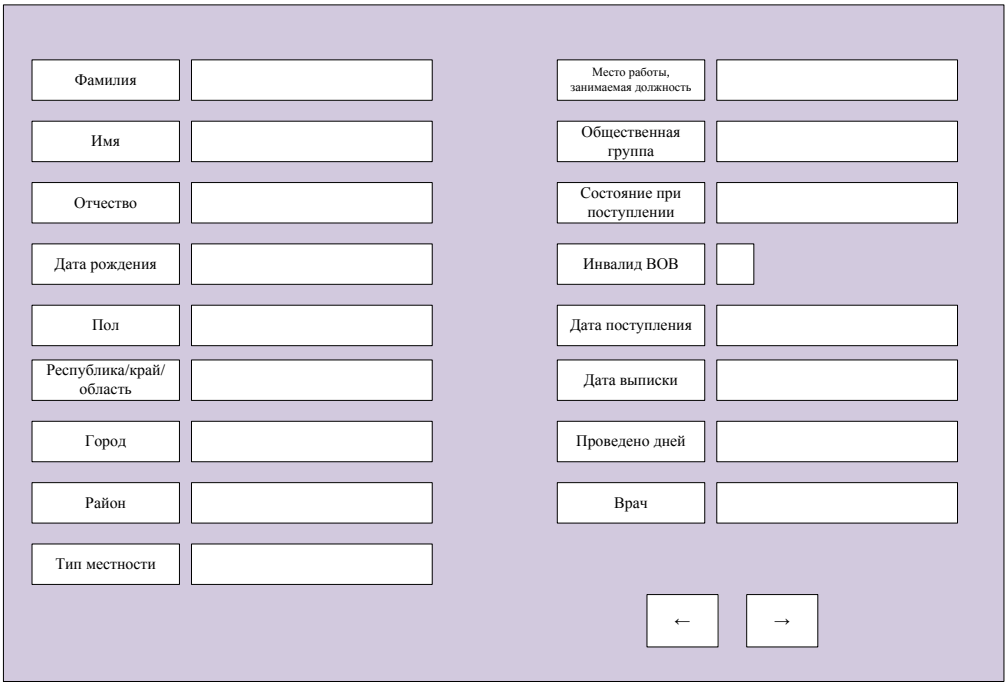


Рисунок 4.5 – *Интерфейс «Карточка пациента»*

Если пользователю необходимо выбрать процедуру для пациента, то требуется выбрать кнопку «Установить диагноз», выбрать диагноз из выпадающего списка, а подходящие процедуры появятся ниже (рис. 4.6).

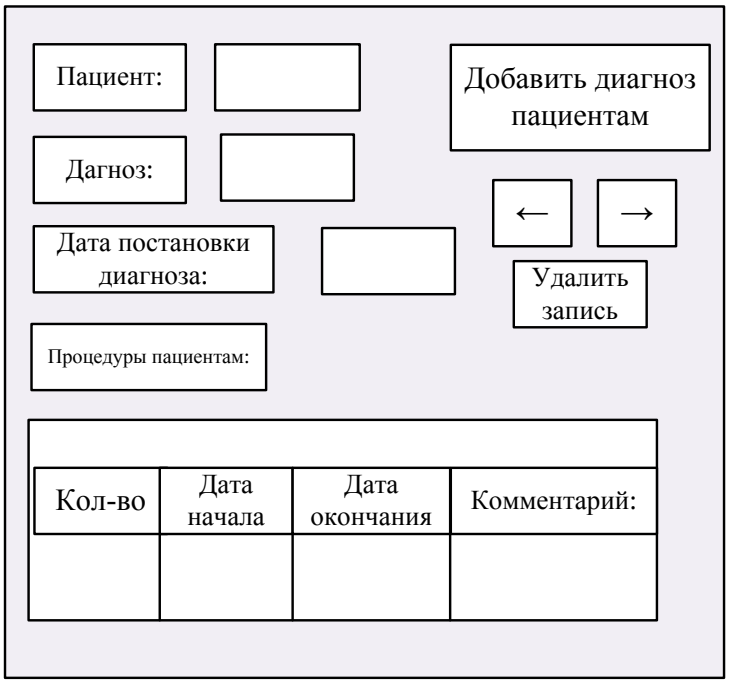


Рисунок 4.6 – *Интерфейс «Установить диагноз»*

После выписки пациенту выдается «Карта выбывшего из санатория» (рис. 4.7-4.8).

Рисунок 4.7 – *Интерфейс «Карта выбывшего из санатория», выбор пациента*

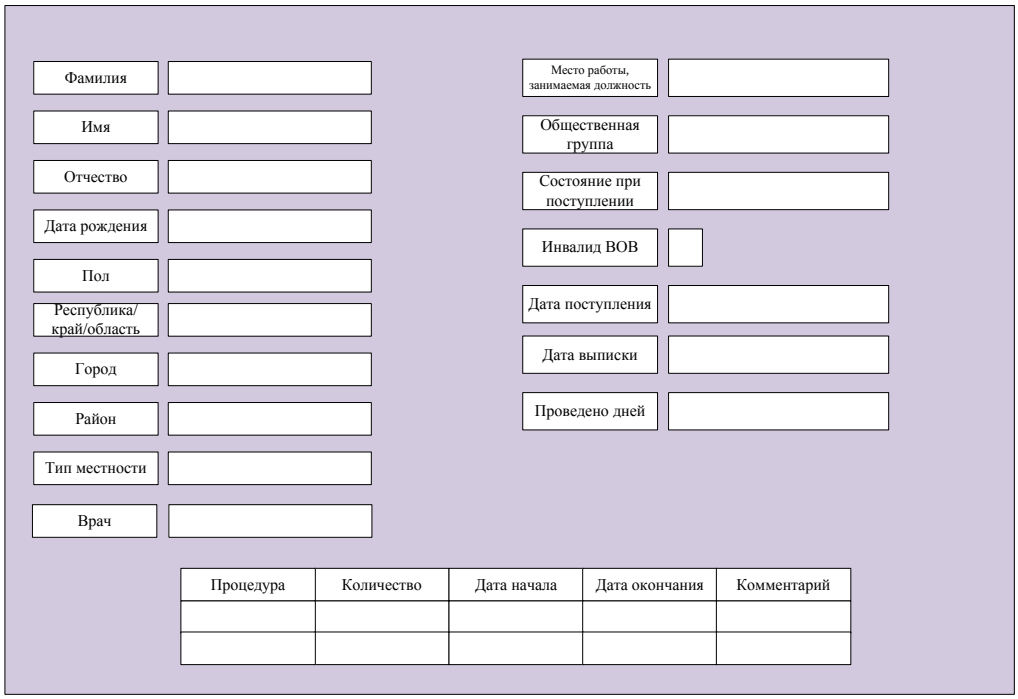


Рисунок 4.8 – *Интерфейс «Карта выбывшего из санатория», информация о пациенте*

Если пользователю необходимо составить отчет, то нажав на кнопку «Отчеты», можно выбрать одну из представленных форм отчетов (рис. 4.9).

Рисунок 4.9 – *Интерфейс «Отчеты»*

Исходя из спроектированных выше интерфейсов, составим схему работы приложения (рис. 4.10 – 4.11).

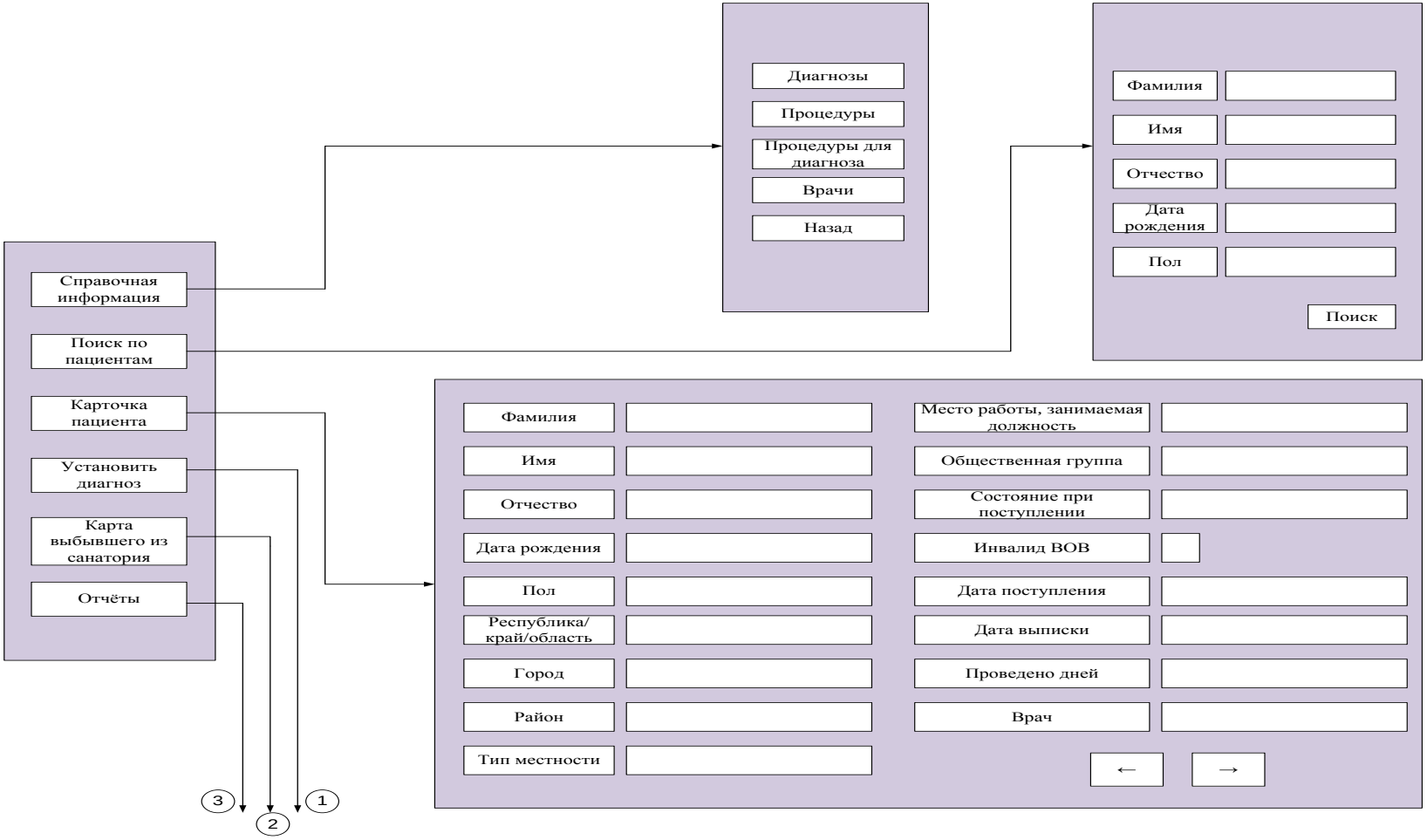


Рисунок 4.10 – *Схема работы приложения (часть 1)*

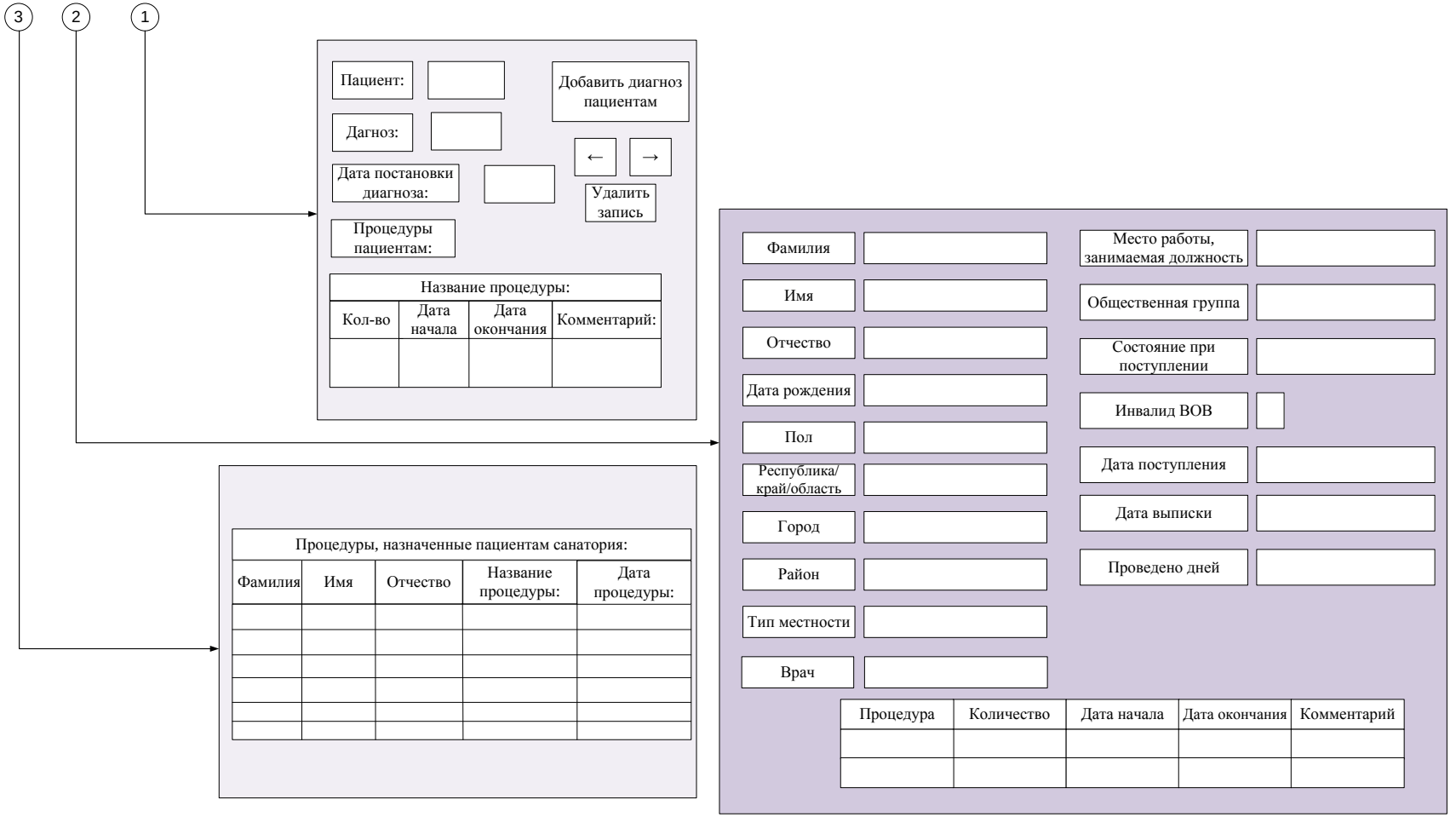


Рисунок 4.11 – *Схема работы приложения (часть 2)*

4.3. *Первый образец программы (Итерация 1)*

Реализация первых двух пользовательских требований «Хранение личных данных пациента, диагноз, процедуры, врачи» и «Автоматический выбор процедуры» подразумевает собой создание таблиц в среде разработки MS Access. В базу данных включены таблицы: «Пациенты», «Диагноз», «Процедуры», «Врачи», в которых хранятся соответствующие данные (рис. 4.12).

Пациенты

Код_пациен	Фамилия	Имя	Отчество	Дата_Рожд	Пол	Республика	Город
1	Алешин	Сергей	Валентинович	19.04.1987	Мужской	Оренбургская	Орск
2	Алексеева	Петр	Сергеевич	06.06.1966	Женский	Рязанская	Тума
3	Греночкин	Алексей	Иванович	12.12.1970	Мужской	Вологодская	Череповец
4	Денисов	Павел	Николаевич	13.03.1977	Мужской	Башкортостан	Салават
5	Иванович	Павел	Дмитриевич	10.11.1990	Женский	Татарстан	Казань
6	Глушаков	Денис	Давыдович	27.01.1987	Мужской	Белгородская	Губкин
7	Карпин	Валерий	Петрович	02.02.1939	Мужской	Ставропольск	Кисловодск
8	Джиоева	Мария	Ивановна	03.04.1941	Женский	Ярославская	Алексеевское
9	Громова	Александра	Павловна	09.09.1999	Женский	Тульская	Тула
10	Пыркин	Андрей	Юрьевич	06.06.1945	Мужской	Тверская	Торжок
12	Иванов	Степан	Владимирович	07.04.1977	Мужской	Мурманская	Североморск
13	Васечкин	Василий	Петрович	12.04.1977	Мужской	Краснодарски	Геленджик

Рисунок 4.12 – *Фрагмент БД таблицы «Пациенты»*

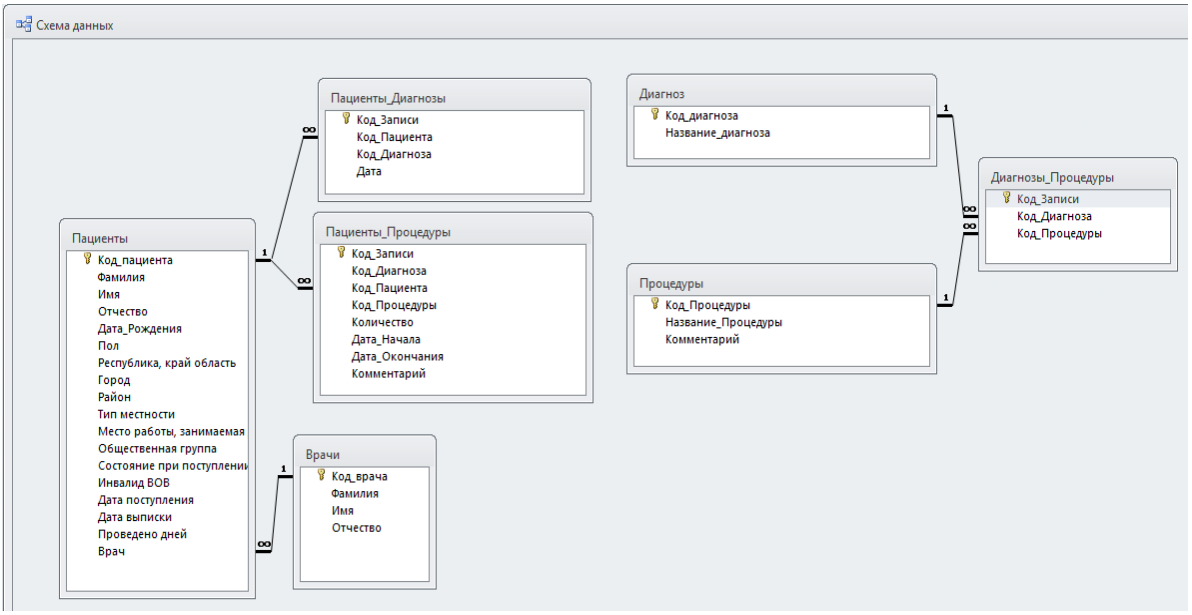


Рисунок 4.13 – *Схема данных приложения*

Далее, необходимо создать «промежуточные» таблицы «Пациенты_Диагнозы», «Пациенты_Процедуры», «Диагнозы_Процедуры», которые потребуются для реализации основного бизнес-процесса «Автоматический выбор процедуры» (рис. 4.13).

Следующим этапом будет создание интерфейса «Установить диагноз», необходимой для установки диагноза из выпадающего списка и выбора процедуры из подчиненной формы (рис. 4.14).

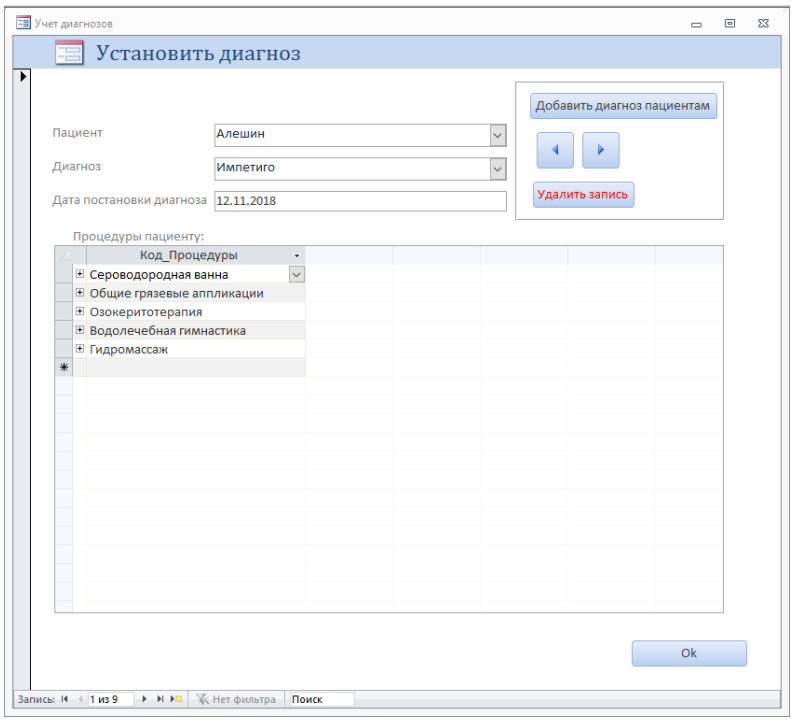


Рисунок 4.14 – *Интерфейс «Установить диагноз» с возможностью выбора диагноза и постановки процедуры*

4.4. Второй образец программы (Итерация 2)

Затем, предоставим врачу возможность добавлять, редактировать, удалять данные, как и о лечащих врачах, так и о процедурах, диагнозах тем самым реализуем требование № 3 «Добавление редактирование, удаление данных пациента, врача, диагноза, процедуры». Появляется необходимость создать интерфейс, требующийся для выполнения выше названных процессов (рис. 4.15).

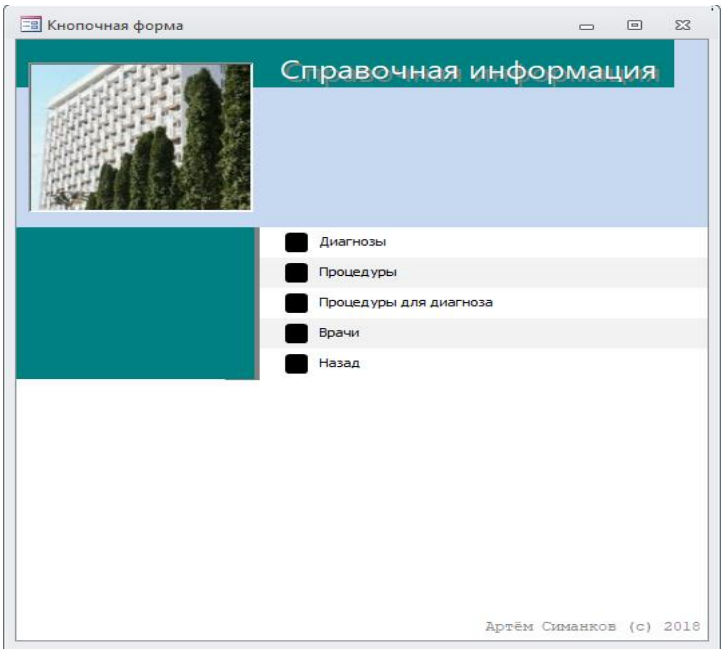


Рисунок 4.15 – *Главный интерфейс приложения для работы с таблицами*

Реализуем электронную медицинскую карту с целью упрощения работы врача и уменьшения времени работы с документацией (рис. 4.16).

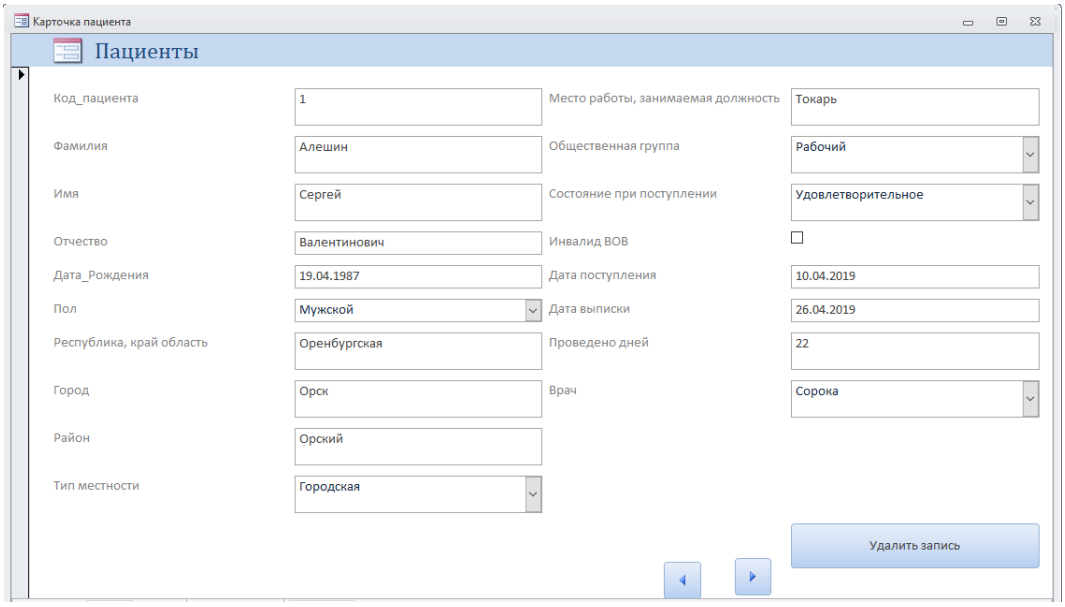


Рисунок 4.16 – *Интерфейс «Карточка пациента» для работы с таблицей «Пациенты»*

4.5. Третий образец программы (Итерация 3)

Третья итерация направлена на реализацию требования № 5 «Просмотр записей диагнозы пациентов, процедуры пациентов, врачи» основная цель которой – формирование отчетов. В среде MS Access они необходимы для вывода на экран определенной информации из базы данных. Разработаем интерфейс для удобной навигации между кнопками отчета (рис.4.17-рис.4.18).

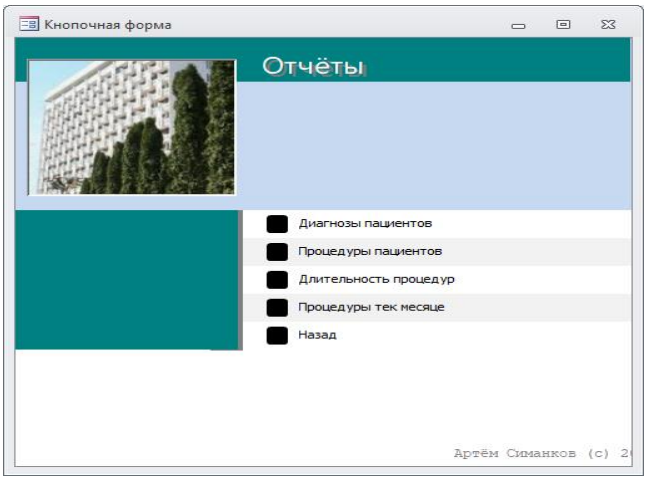


Рисунок 4.17 – *Интерфейс «Отчеты» для работы с отчетами*

Рисунок 4.18 – *Отчет «Процедуры, назначенные пациентам»*

Создание «Карта выбывшего из санатория» подразумевает реализацию требования № 6 «Вывод эпикриза с назначенными процедурами». Это необходимо для взаимного обмена информацией между медицинскими учреждениями (рис.4.19).

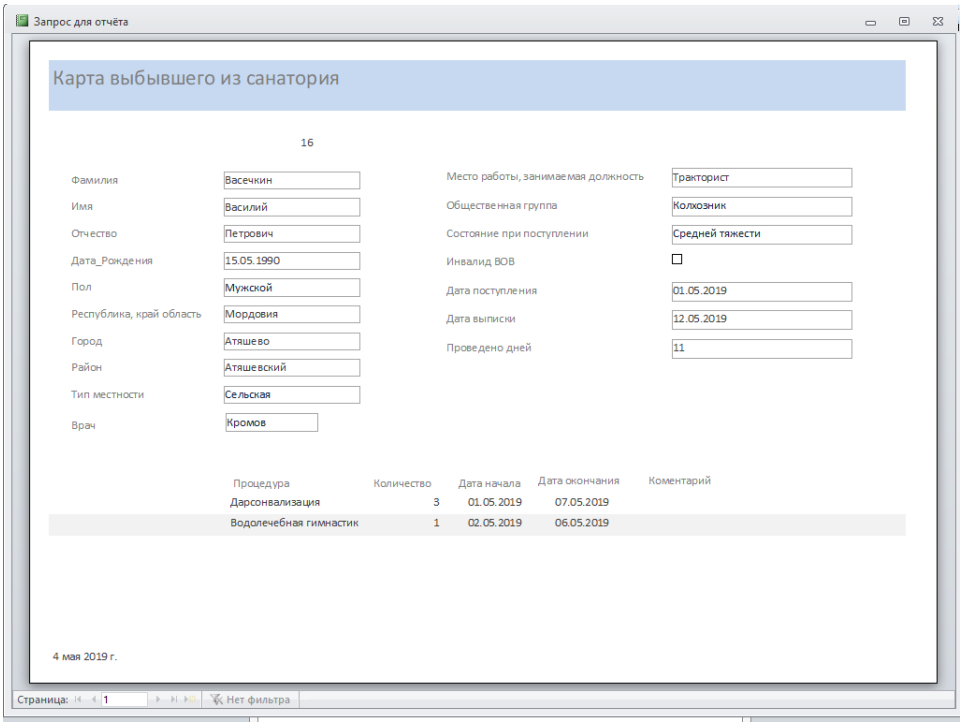


Рисунок 4.19 – *Интерфейс «Карта выбывшего из санатория»*

4.6. Четвертый образец программы (Итерация 4)

Разработка требования № 7 «Осуществление поиска пациента» включает в себя создание запросов для поиска нужной информации среди всей базы данных. Запрос с названием «Поиск» позволяет по любой части фамилии найти конкретного пациента (рис. 4.20).

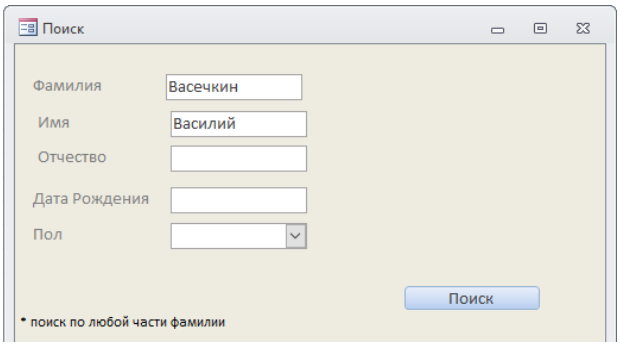


Рисунок 4.20 – *Окно поиска процедур и диагноза пациента*

Отобразим реализацию данного требования программно путем представления VBA кода на рисунке 4.21 [12].

```
Option Compare Database

Private Sub Кнопка3_Click()
DoCmd.OpenQuery ("Поиск")
End Sub

Private Sub Кнопка5_Click()
Dim ssql As String
ssql = "SELECT Пациенты.Фамилия, Пациенты.Имя, Пациенты.Отчество, Пациенты.Дата_Рождения, Пациенты.Пол,
Диагноз.Название_диагноза, Процедуры.Название_Процедуры, Пациенты_Процедуры.Дата_Начала, Пациенты_Процедуры.Дата_Окончания"
ssql = ssql + " FROM Пациенты INNER JOIN ((Пациенты_Процедуры INNER JOIN Диагноз ON Пациенты_Процедуры.Код_Диагноза =
Диагноз.Код_диагноза) INNER JOIN Процедуры ON Пациенты_Процедуры.Код_Процедуры = Процедуры.Код_Процедуры) ON
Пациенты.Код_пациента = Пациенты_Процедуры.Код_Пациента"
ssql = ssql + " WHERE Пациенты.[Код_пациента] <> 0 "
If Nz(Фамилия) > 0 Then ssql = ssql + " and (Пациенты.Фамилия) = " & "'" & Фамилия & "'"
If Nz(Имя) > 0 Then ssql = ssql + " and (Пациенты.Имя) = " & "'" & Имя & "'"
If Nz(Отчество) > 0 Then ssql = ssql + " and (Пациенты.Отчество) = " & "'" & Отчество & "'"
If Nz(Пол) > 0 Then ssql = ssql + " AND (Пациенты.Пол) =" & "'" & Пол & "'"
If Nz([Дата_Рождения]) > 0 Then ssql = ssql + " AND (Пациенты.[Дата_рождения]) =" & Replace("#" & [Дата_Рождения] & "#", ".",
"/")
ssql = ssql + " order by [Фамилия];"
'MsgBox (ssql)
fncShowQ (ssql)
End Sub
```

Рисунок 4.21 – *Листинг программы по поиску пациента*

Составим блок-схему алгоритма работы программы. Это позволит наглядным способом представить работу ключевого бизнес-процесса (рис. 4.22) [13]. Подобный запрос помогает пользователю отследить процедуру, дату ее начала и окончания (рис. 4.23).

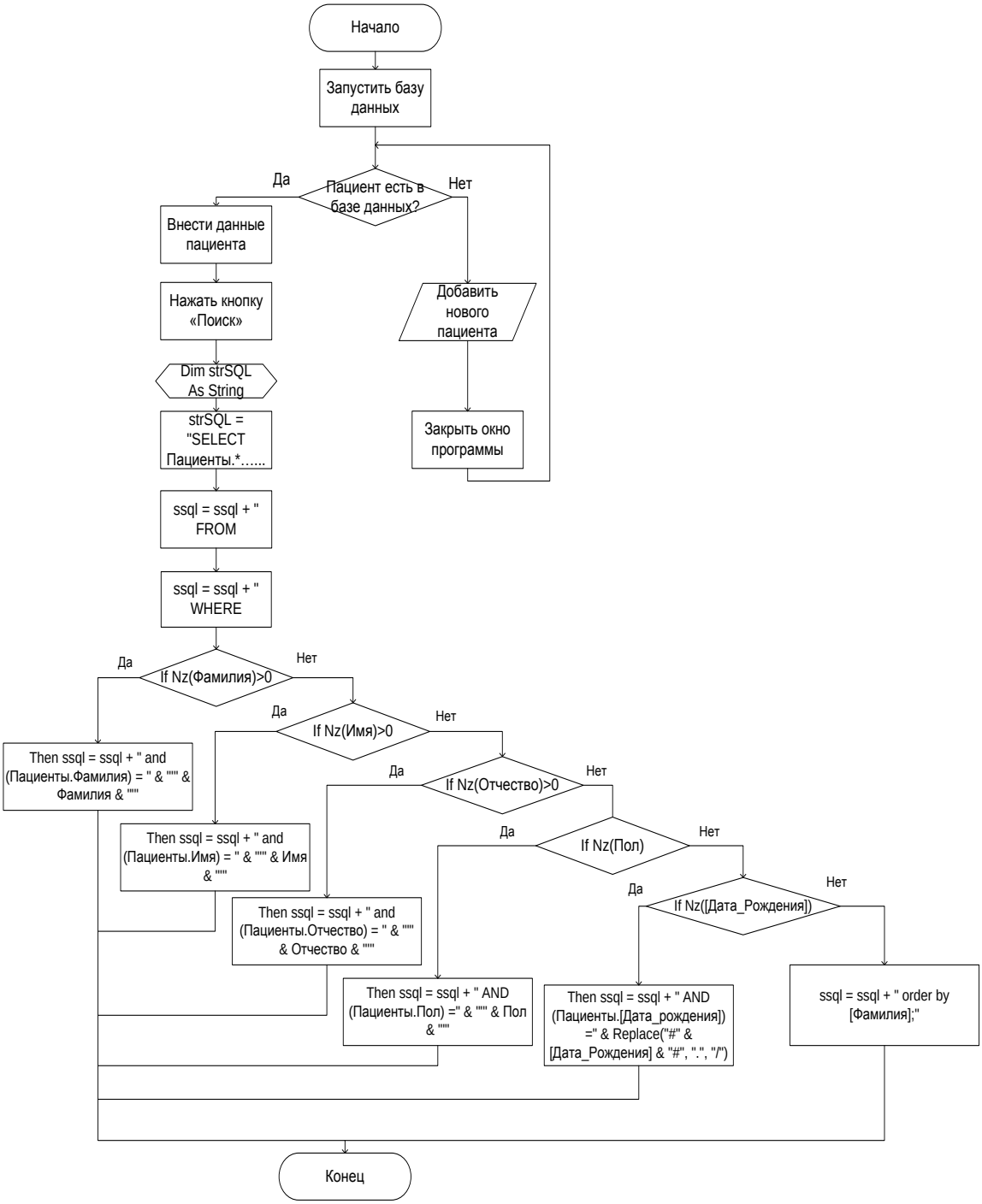


Рисунок 4.22 – *Блок-схема работы «Поиска пациента»*

qtmp

Фамилия	Имя	Отчество	Дата_Рожд	Пол	Название_диагноза	Название_Процедуры	Дата_Начал	Дата_Окон
Васечкин	Василий	Петрович	15.05.1990	Мужской	Экзема	Дарсонвализация	01.05.2019	07.05.2019
Васечкин	Василий	Петрович	15.05.1990	Мужской	Экзема	Водолечебная гимнастика	02.05.2019	06.05.2019

Рисунок 4.23 – *Таблица с выданным запросом по поиску процедур и диагнозу на пациента Васечкина*

Исходя из требования № 8 «Легкость в освоении медицинским персоналом» для удобства работы с программой необходимо сформировать интуитивно понятный интерфейс, который обладает такими функциями доступности: общепонятность, простота, легкость (рис. 4.24).

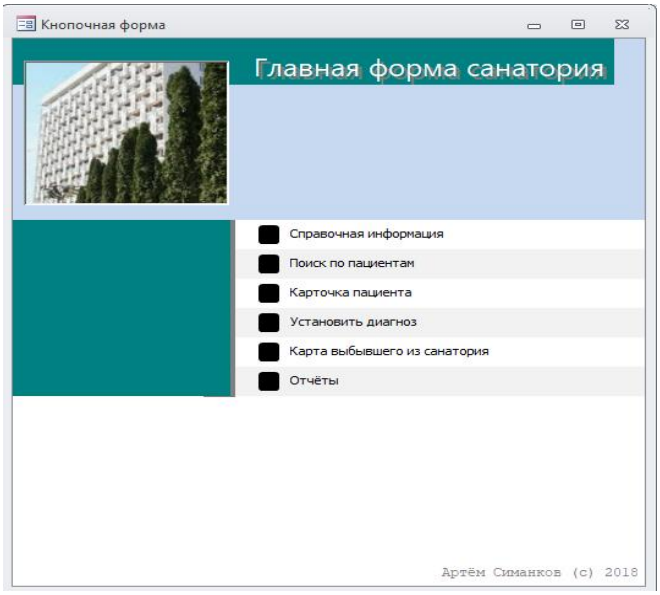


Рисунок 4.24 – *Главное меню приложения*

Реализация требования № 9 «Печать эпикриза» предоставляет возможность вывода на печать необходимой информации. Таким образом отобразить «Карта выбывшего из санатория» на бумажный носитель предоставляется возможным (рис.4.25).

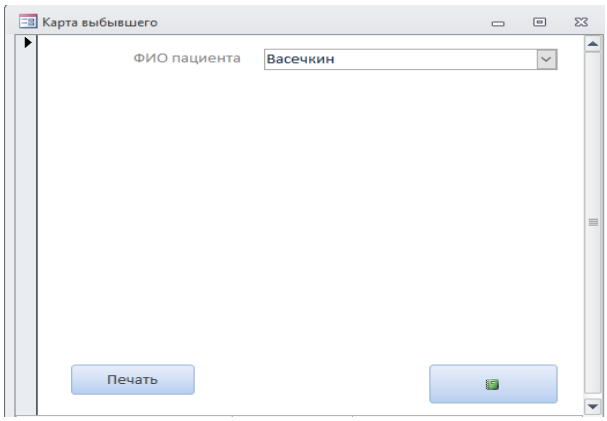


Рисунок 4.25 – *Окно с возможностью печати «Карта выбывшего из санатория»*

Раздел 5. Тестирование разработанного приложения

5.1. Функциональное тестирование

Функциональное тестирование – вид тестирования, целью которого является проверка правильности работы функционала разрабатываемого приложения (корректность реализации функциональных требований). Его задача состоит в определении того, какие возможности продукта были разработаны и в проверке того, что они функционируют должным образом [14]. Для проверки реализации функциональных требований проведем их анализ.

Первое функциональное требование гласит: «База данных должна содержать таблицы «Пациенты», «Процедуры», «Диагноз», «Врачи», включающая сведения о пациенте, его диагнозе, процедурах и датах проведения». В графическом представлении отобразим созданные таблицы (рис. 5.1).

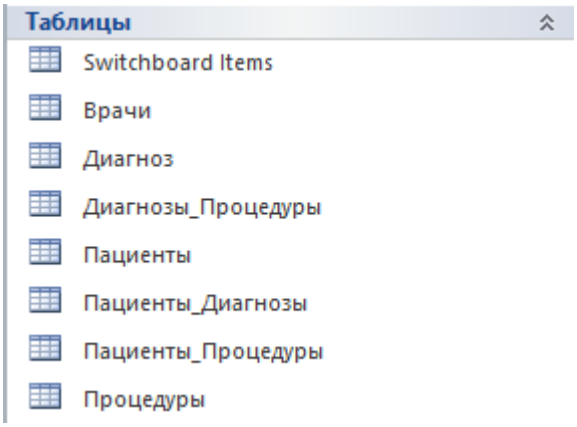


Рисунок 5.1 – *Таблицы «Пациенты», «Процедуры», «Диагноз» в MS Access*

Второе функциональное требование предполагает наличие таблицы «Диагноз_Процедуры», где каждому диагнозу соответствует определенный перечень процедур, необходимый для реализации ключевого бизнес-процесса (рис. 5.2).

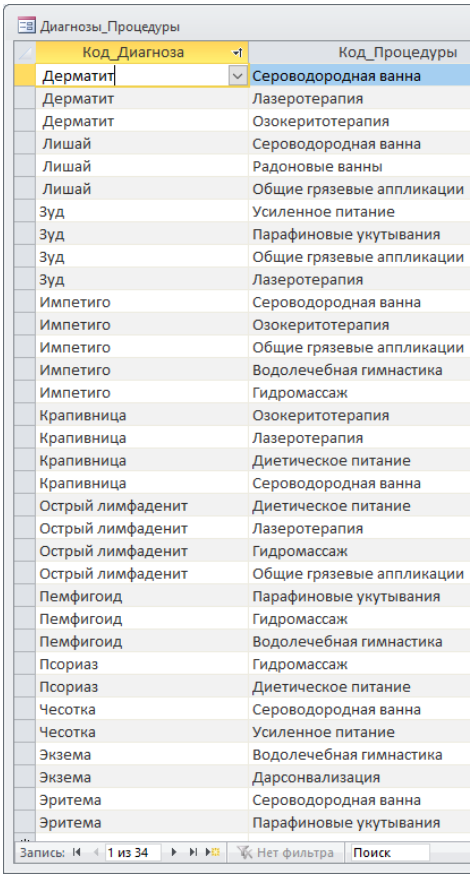


Рисунок 5.2 – *Таблица «Диагноз_Процедуры» в MS Access*

Третье функциональное требование обязует возможность исправления удаления и внесения новой информации о пациенте в таблицу данных «Пациенты», «Процедуры», «Диагноз». Ниже, в графическом представлении продемонстрируем возможность работать с данными в выше упомянутых таблицах (рис. 5.3, рис. 5.4, рис. 5.5).

Пациенты

Фамилия	Имя	Отчество	Дата_Рождения	Пол
Алешин	Алексей	Алексеевич	19.04.1987	Мужской
Алексеева	Александра	Владимировна	06.06.1966	Женский
Греночкин	Вячеслав	Вячеславович	12.12.1970	Мужской
Денисов	Игорь	Антонович	13.03.1977	Мужской
Иванович	Алина	Сергеевна	10.11.1990	Женский
Глушаков	Денис	Борисович	27.01.1987	Мужской
Карпин	Валерий	Георгиевич	02.02.1969	Мужской
Джиоева	Мария	Ивановна	03.04.1941	Женский
Громова	Александра	Павловна	09.09.1999	Женский
Пыркин	Андрей	Юрьевич	06.06.1966	Мужской
Иванов	Иван	Иванович	07.04.1977	Мужской
Васечкин	Василий	Петрович	12.04.1977	Мужской

Рисунок 5.3 – *Таблица «Пациенты» с возможностью работы с данными*

Процедуры

Название_Процедуры
Сероводородная ванна
Радоновые ванны
Общие грязевые аппликации
Лазеротерапия
Озокеритотерапия
Парафиновые укутывания
Гидромассаж
Водолечебная гимнастика
Дарсонвализация
Диетическое питание
Усиленное питание

Рисунок 5.4 – *Таблица «Процедуры» с возможностью работы с данными*

Диагноз

Название_диагноза
Импетиго
Острый лимфаденит
Пемфигоид
Дерматит
Зуд
Лишай
Крапивница
Эритема
Псориаз
Экзема
Чесотка

Рисунок 5.5 – *Таблица «Диагноз» с возможностью работы с данными*

Функция просмотра данных из БД «Пациенты», «Процедуры», «Диагноз» позволяет вывести на экран запрашиваемые данные благодаря

созданным отчетам. Представим программную реализацию данного требования и корректность ее работы на рисунке 5.6.

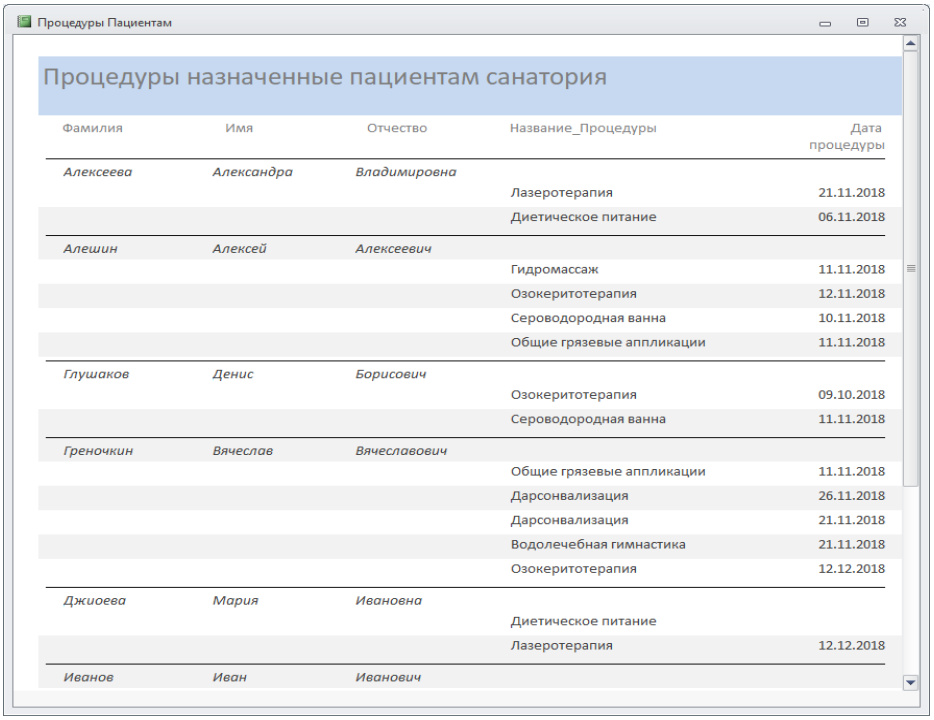


Рисунок 5.6 – *Отчет «Процедуры, назначенные пациентам»*

Возможность поиска записей по пациентам предоставляет найти записи в таблице по фамилии, дате рождения, полу. Изучим правильность работы программы на примере поиска пациента по фамилии Алешин (рис. 5.7). Результат поиска показан на рисунке 5.8.

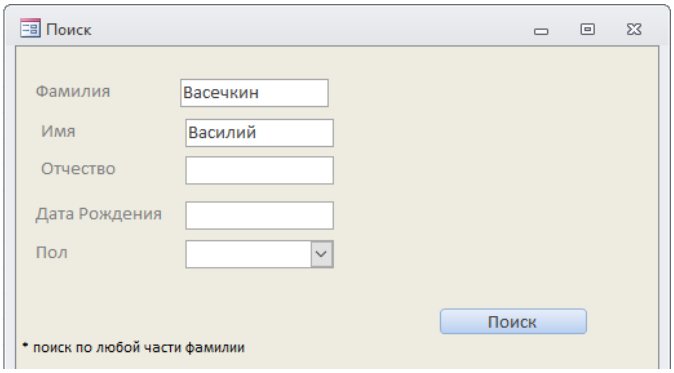


Рисунок 5.7 – *Окно поиска процедур и диагноза пациента Васечкина*

qtmp

Фамилия	Имя	Отчество	Дата_Рожд	Пол	Название_диагноза	Название_Процедуры	Дата_Начал	Дата_Окон
Васечкин	Василий	Петрович	15.05.1990	Мужской	Экзема	Дарсонвализация	01.05.2019	07.05.2019
Васечкин	Василий	Петрович	15.05.1990	Мужской	Экзема	Водолечебная гимнастика	02.05.2019	06.05.2019

Рисунок 5.8 – *Таблица с выданным запросом по поиску процедур и диагнозу на пациента Васечкина*

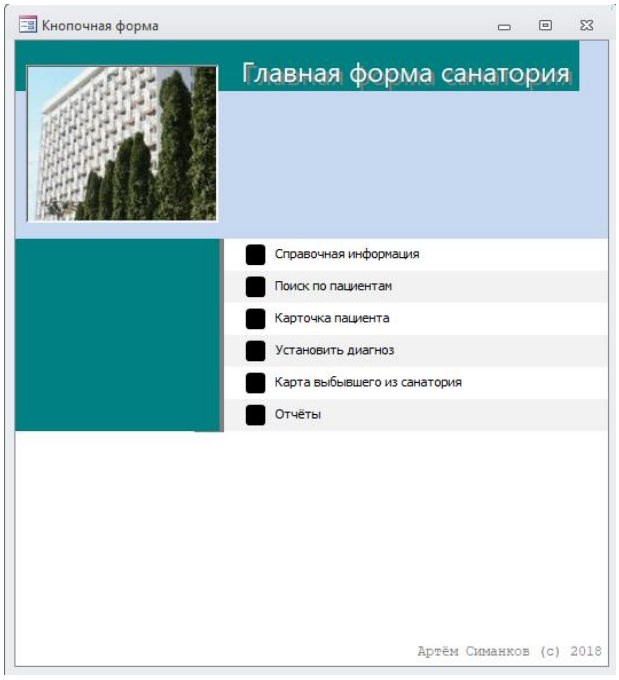


Рисунок 5.9 – *Главное меню приложения*

Функция доступности предполагает возможность любому пользователю легко взаимодействовать с системой (интерфейсом приложения). Если пользователю будет непонятно или трудно разбираться с приложением, он, скорее всего, просто откажется от этого. Следовательно, любо приложение должны быть понятными для пользователя на уровне интуиции (рис. 5.9).

Место хранения в различных условиях должно соответствовать требованиям, предъявляемым к носителям информации на которых будет содержаться данное программное изделие. Программа может храниться на жестком диске персонального компьютера, на Flash-носителе, на компакт-

дисках. Из вышесказанного можно сделать вывод, что функциональное тестирование показало отсутствие ошибок и полную работоспособность ПО. Результаты тестирования свидетельствуют о том, что все функциональные требования успешно реализованы.

5.2. *Интеграционное тестирование*

Интеграционное тестирование – тестирование, которое проводится с целью проверки правильного взаимодействия между несколькими элементами приложения. Данный вид тестирования производит глубокую проверку взаимодействия компонентов в ситуации, когда пользователю почти нечего наблюдать, т. к. все представляющие интерес и подвергаемые тестированию процессы проходят на уровнях более глубоких, чем пользовательский интерфейс. Главные цели интеграционного тестирования состоят в тестировании составляющих модулей программы на неконфликтность и соответствие требованиям. Следующая цель заключается в тестировании корректности взаимодействия нескольких модулей, объединенных в единое целое [14]. Достаточно проверить корректность поиска введенной ранее информации. Вызовем запрос на поиск пациента (рис.5.10).

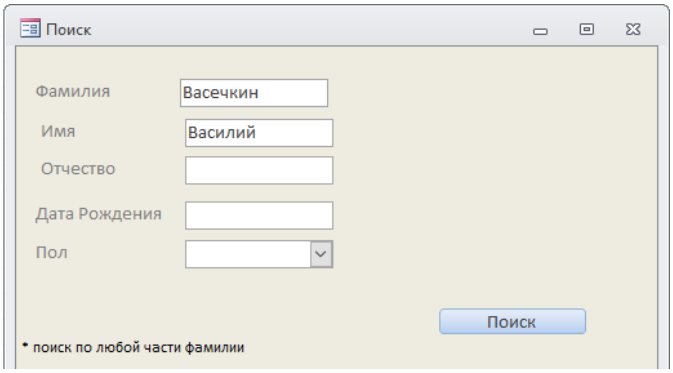
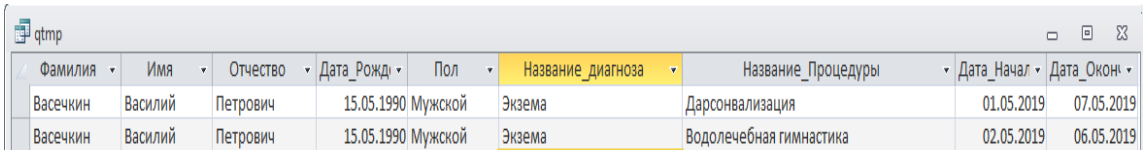


Рисунок 5.10 – *Запрос на поиск пациента*

Рисунок 5.11 содержит таблицу, демонстрирующую правильность запрашиваемой ранее информации.



qtmp

Фамилия	Имя	Отчество	Дата_Рожд	Пол	Название_диагноза	Название_Процедуры	Дата_Начал	Дата_Окон
Васечкин	Василий	Петрович	15.05.1990	Мужской	Экзема	Дарсонвализация	01.05.2019	07.05.2019
Васечкин	Василий	Петрович	15.05.1990	Мужской	Экзема	Водолечебная гимнастика	02.05.2019	06.05.2019

Рисунок 5.11 - *Таблица с выданным запросом по поиску процедур и диагнозу на пациента Васечкина*

5.3. Нагрузочное тестирование

Нагрузочное тестирование заключается в проверке возможности приложения сохранять требуемые параметры при воздействии предельных нагрузок и определенном превышении пределов [10]. Для данного тестирования взято следующее количество записей пациентов: 10, 50, 100. Определим время отклика при работе с разным количеством записей и занесем значения в табл. 5.2 в столбцы 3-5.

Далее, определим среднее арифметическое всех значений времени отклика для определенного количества записей и занесем в столбец 6. Среднее арифметическое значение рассчитывается по формуле:

$$\overline{t_{\text{арифм.}}} = \frac{\sum_i t_i}{N} \quad (5.1)$$

Среднее квадратичное отклонение для определенного количества записей рассчитывается по формуле:

$$\sigma = \sqrt{\frac{\Delta t_i^{\,2}}{N}}, \quad (5.2)$$

где N– число значений. Погрешность измерений рассчитывается по формуле:

$$\Delta t = \sqrt{(\frac{\sigma}{N} \cdot t_{\alpha(N-1)})^2 + \Delta \mathrm{t_p}^2}, \quad (5.3)$$

где $t_{\alpha(N-1)}$ – доверительный коэффициент Стьюдента, равный 0.95; Δt_p – абсолютная погрешность электронного секундомера равная 0,005, которым производилось измерение [15]. Время отклика определяется:

$$t_{\text{откл.}} = t_{\text{арифм.}} \pm \Delta t \,, \qquad (5.4)$$

Данные по нагрузочному тестированию приведены в таблице 5.2.

Таблица 5.2 – *Результаты нагрузочного тестирования*

Количество записей	Действие	t_1,с	t_2,с	t_3, с	Среднее время отклика, с	Среднее квадратичное отклонение, с	Погрешность измерений, с	Время отклика, с
10	Поиск	0,04	0,07	0,13	0,08	0,0458	0,0153	0,08±0,015
	Вывод на экран информации	0,12	0,17	0,23	0,17	0,0551	0,0182	0,17±0,018
50	Поиск	0,22	0,27	0,33	0,27	0,0551	0,0182	0,27±0,018
	Вывод на экран информации	0,38	0,41	0,49	0,43	0,0569	0,0187	0,43±0,019
100	Поиск	0,3	0,36	0,44	0,37	0,0702	0,0228	0,37±0,023
	Вывод на экран информации	0,42	0,49	0,45	0,45	0,0351	0,0122	0,45±0,012

Раздел 6. Расчет затрат на разработку и внедрение программного продукта

6.1. Расчет заработной платы исполнителей проекта

Разработка и внедрение программного продукта – длительный и сложный по времени процесс, включающий в себя несколько стадий и этапов. По расчетам на разработку и внедрение отводится 57 дней. Стадии, этапы и их продолжительность представлена в таблице 6.1.

Таблица 6.1 – *Стадии и этапы разработки и внедрения программного продукта*

№ стадии	Стадия	№ этапа	Этап	Продолжительность работ, дни	
1	Разработка технического задания			4	
2	Подготовка эскизного проекта	2.1	Постановка задачи	1	9
		2.2	Идентификация, анализ, приоритизация требований	4	
		2.3	Выбор методологии моделирования	1	
		2.4	Подбор модели разработки программного обеспечения	3	
3	Подготовка технического проекта	3.1	Формирование проекта разработки по выбранной модели разработки программного обеспечения	1	4
		3.2	Нормализация входных параметров и данных	1	
		3.3	Моделирование, структурирование данных и интерфейсов программы	2	
4	Подготовка рабочего проекта	4.1	Программирование программы	8	33
		4.2	Тестирование программы	3	
		4.3	Отладка программы	4	
		4.4	Демонстрация готового продукта	1	
		4.5	Адаптация программы	3	
		4.6	Разработка рабочей документации	14	

№ стадии	Стадия	№ этапа	Этап	Продолжительность работ, дни	
5	Внедрение	5.1	Подготовка персонала	5	7
		5.2	Пусконаладочные работы	1	
		5.3	Проведение предварительных и приемочных испытаний	1	

Произведем расчет затрат на оплату труда. Затраты на оплату труда состоят из основной заработной платы и дополнительной заработной платы. Основная зарплата начисляется исходя из отработанного времени и оклада. Расчет основной заработной платы производится по формуле (6.1).

$$ЗП_{осн.} = ЗП_{рук.} + ЗП_{раз.}, \quad (6.1)$$

Расчет заработной платы руководителя ($ЗП_{рук.}$) и разработчика ($ЗП_{раз.}$) производится делением месячного оклада на количество рабочих дней в месяц (22 дня) с последующие умножением на продолжительность работ [16]. Представим данные в таблице 6.2.

Таблица 6.2 – *Основная заработная плата исполнителей проекта*

№ стадии	Стадия	Должность	Месячный оклад	Продолжительность работ, дни	Сумма основной заработной платы, руб.
1	Разработка технического задания	Руководитель	260000	4	47272,7
2	Подготовка эскизного проекта	Руководитель	260000	9	106363,7
		Разработчик	70000	9	28636,4
3	Подготовка технического проекта	Руководитель	260000	4	47272,7
		Разработчик	70000	4	12727,3
4	Подготовка рабочего проекта	Руководитель	260000	33	390000,0
		Разработчик	70000	33	105000,0
5	Внедрение	Руководитель	260000	7	82727,3
		Разработчик	70000	7	22272,7

№ стадии	Стадия	Должность	Месячный оклад	Продолжительность работ, дни	Сумма основной заработной платы, руб.
Руководитель				57	673636,4
Разработчик				53	168636,4
Итого					842272,8

Подставив в формулу (6.1) числовые значения, вычислим основную заработную плату исполнителей проекта: $ЗП_{осн.} = 673636{,}4 + 168636{,}4 = 842272{,}8$.

Расчет дополнительной заработной платы осуществляется на основании 15 % от основной заработной платы по формуле (6.2).

$$ЗП_{доп.} = 0{,}15 \cdot ЗП_{осн.}, \quad (6.2)$$

Подставим в формулу (6.2) числовые значения: $ЗП_{доп.} = 0{,}15 \cdot 842272{,}8 = 126340{,}9$. Таким образом, рассчитаем фонд оплаты труда по формуле (6.3).

$$Ф_{зп.} = З_{осн.} + З_{доп.}, \quad (6.3)$$

т.е. $Ф_{зп.} = 842272{,}8 + 126340{,}9 = 968613{,}7$ руб.

6.2. Расчет затрат на электроэнергию, амортизацию оборудования и прочих затрат

Произведем расчет материальных затрат, исходя из того, что при разработке и внедрении программного продукта оборудование и специальное программное обеспечение не приобреталось. Таким образом, в расчет затрат включаем только расходы на электроэнергию. Тогда, расчет затрат на электроэнергию осуществляется по формуле (6.4).

$$З_{эл.} = Р \cdot Ц_{эл.} \cdot Т_{и}, \quad (6.4)$$

где Р – потребляемая мощность оборудования, кВт/ч; $Ц_{эл}$ – стоимость 1 кВт/ч, руб.; $Т_и$ – время использования оборудования при проведении работ, ч. [16].

При разработке, тестировании, отладке программного продукта использовался один персональный компьютер с потребляемой мощностью 500 Вт/ч, который работал 57 рабочих дней по 8 часов. Стоимость 1кВт энергии составляет 4,37 руб./кВт (на 2019 год) [17]. Подставив в формулу (6.4) числовые значения вычислим материальные затраты: $З_{эл.} = 0{,}5 * 4{,}37 * 57 * 8 = 996{,}4$ руб.

Далее произведем расчет затрат на амортизацию оборудования. Амортизационные отчисления учитываются в сметной стоимости научно-исследовательской и опытно-конструкторской работы и рассчитываются по формуле (6.5):

$$А_{нир.} = \frac{Ф_{перв.} \cdot Т_и \cdot Н_а}{Ф_{эф.}}, \quad (6.5)$$

где $Ф_{перв}$ – первоначальная стоимость оборудования, руб.; $Т_и$ – время использования оборудования при проведении работ, дн.; $Н_а$ – норма амортизации; $Ф_{эф}$ – годовой эффективный фонд времени работы оборудования, для односменной работы он составляет 256 дн. [16].

Норма амортизации $Н_а$ является величиной, обратной сроку службы оборудования, и считается по формуле (6.6):

$$Н_а = \frac{1}{Т_{пи}}, \quad (6.6)$$

где $Т_{пи}$ – срок службы оборудования, лет [16].

Срок службы компьютера – 2-3 года (на 2019 г.) [18]. Тогда норма амортизации рассчитывается следующим образом: $Н_а = \frac{1}{3} = 0{,}3$. Затраты на

амортизацию оборудования за 57 дней использования для компьютера стоимостью в 55000 руб. составят: $А_{нир} = \frac{55000 \cdot 57 \cdot 0,33}{256} = 4041,2$ руб.

Общие прямые затраты рассчитываются по формуле (6.7), как сумма затрат на заработную плату, затрат на электроэнергию и затрат на амортизацию оборудования:

$$З_{прям} = Ф_{зп} + З_{эл.} + А_{нир.} \quad (6.7)$$

Подставив в формулу (6.7) числовые значения и рассчитаем сумму прямых затрат: $З_{прям} = 968613,7 + 996,4 + 4041,2 = 973651,3$ руб. Расчет прочих расходов подразумевает расчет страховых взносов и дополнительных прочих расходов по формуле (6.8):

$$З_{пр.} = страх.\,взносы + величина\,доп.\,проч.\,расходов \quad (6.8)$$

Ставка страховых взносов в 2019 г. составляет 30% от величины фонда оплаты труда [19]. Таким образом, страховые взносы составят: 968613,7*0,3 =290584,1 руб.

Величина дополнительных прочих расходов рассчитывается как 10% от суммы прямых затрат. Дополнительные прочие затраты: 973651,3*0,1 = 97365,1 руб. В итоге, прочие затраты составят: $З_{пр}$ = 290584,1+97365,1 =387949,2 руб.

6.3. Расчет сметы затрат

Общие затраты на разработку и внедрение программного продукта – это сумма прямых затрат и прочих затрат. Расчет производится по формуле (6.9):

$$З = З_{прям} + З_{пр}, \quad (6.9)$$

Подставив в формулу (6.9) числовые значения, вычислим значение общих затрат на разработку и внедрение программного продукта: З = 973651,3 + 387949,2 = 1361601,0 руб.

В таблице 6.3 представлена смета затрат на разработку и внедрение программного продукта.

Таблица 6.3 – *Смета затрат*

Наименование статьи	Сумма затрат, руб.	Удельный вес, %
Затраты на заработную плату	968613,7	71,1
Затраты на электроэнергию	996,4	0,1
Затраты на амортизацию оборудования	4041,2	0,3
Прочие затраты	387949,2	28,5
Итого затраты на реализацию проекта	1361601,0	100,0

Таким образом, можно сделать следующий вывод. Основная часть затрат – 71,7%, необходимых на реализацию проекта, составляют затраты на заработную плату. Прочие затраты составляют 28,5% от общей суммы затрат по проекту. Затраты на электроэнергию и амортизацию оборудования составляют 0,1% и 0,3% от общей суммы затрат соответственно.

Заключение

Целью данной работы являлось создание приложения для автоматизации работы врача-курортолога в санатории. Стояла необходимость в создании механизмов, которые дадут возможность снизить трудозатраты, сократить временные расходы и повысить комфорт условий труда медицинского персонала учреждений. Для выполнения данной задачи требовалось исследовать и определить все пользовательские требования. Отталкиваясь от них, формировались функциональные требования, на основе которых проходила разработка программного продукта. После этого был составлен план разработки приложения согласно итерационной модели. В результате были получены четыре итерации, в ходе которых реализовывались новые функции, а также те, которые были разработаны в процессе прошлых итераций.

После окончания анализа требований мы приступили к проектированию и моделированию основных бизнес-процессов учреждения, AS-IS и TO-BE в нотации IDEF0 и DFD для того, чтобы представить разницу «до» и «после» внедрения разрабатываемой системы на каждом уровне организации. Поскольку создание приложения происходило в среде MS Access, требовалась классификация и нормализация данных и моделирование их архитектуры. Создание пользовательских интерфейсов позволило представить, как будет выглядеть приложение в конце. Целью создания интерфейса являлось обеспечение возможности обновления представленных данных по деятельности санатория без использования программирования и специального кодирования. Создание блок-схемы действия алгоритма программы оказалось удобным средством описания, демонстрирующее наглядный принцип работы ключевого бизнес-процесса с помощью геометрических фигур с линиями-связями.

Заключительным этапом созданного приложения являлось тестирование, в основу которого легли такие методологии тестирования: функциональное, интеграционное, нагрузочное. В ходе прохождения всех тестов при различном количестве записей в БД, было выявлено, что приложение отличается высокой производительностью.

Подведя итоги, можно сказать, разработанное программное средство отвечает всем требованиям функциональности, так как выполняет все возложенные на нее функции и требования надежности.

Список использованных литературных источников

1. Федорова Г.Н. Информационные системы: учебник для студ. учреждений сред. проф. образования // Издательский центр «Академия», 2013 – 208 с.
2. Барышникова М.Ю.Инженерный менеджмент и информационные технологии. Конспект лекций. М.: МГТУ им. Н. Э. Баумана, 2009. - 252 с.
3. Карпенко С.Н., Вершинина Е.В., Гонченко М.С. Обзор моделей жизненного цикла разработки программного обеспечения// Математические и программные технологии для современных компьютерных систем (Информационные технологии), 2017 – 69 с.
4. Идентификация требований и определения спецификаций. Конспект лекций. М.: Томский Политехнический Университет. URL: https://studfiles.net/preview/4241597/page:8/ (дата обращения: 01.11.2018)
5. Свободная энциклопедия Википедия, статья «Анализ требований».URL:https://ru.wikipedia.org/wiki/Анализ_требований (дата обращения: 02.11.2018)
6. Лешек А. Мацяшек Анализ требований и проектирование систем. // Издательский дом «Вильямс», 2005 – 432 с.
7. Fieldboom, статья «The Kano Model Explained: Analysis and Examples». URL: https://www.fieldboom.com/kano-model (дата обращения: 02.11.2018)
8. Клебанов Б.И. Проектирование автоматизированных систем обработки информации и управления. Методические указания к выполнению

курсового проекта М.: Уральский государственный технический университет, 2004. – 66 с.

9. Самуйлов К.Е., Серебренникова Н.В., Чукарин А.В., Яркина Н.В. Основы формальных методов описания бизнес-процессов: Учебное пособие. – М.: РУДН, 2008 – 130 с.
10. Морозов Д.А. Применение нотации DFD в моделировании внешних схем // Портал научно-практических публикаций [Электронный ресурс]. URL: http://portalnp.ru/2014/06/2015 (дата обращения: 05.11.2018)
11. Дейт К. Дж. Введение в системы баз данных, 8-е издание. // Издательский дом «Вильямс», 2010 – 1328 с.
12. Культин Н. Б. Цой Л. Б. Visual Basic для студентов и школьников // Издательство «БХВ – Петербург», 2010 – 401 с.
13. Афанасьева Т.В. Основы визуальной алгоритмизации: Учебное пособие для студентов. – М.: Ульяновский государственный технический университет, 2012 – 64 с.
14. Куликов С.С. Тестирование программного обеспечения. Базовый курс. – Минск: Четыре четверти, 2017. – 312 с.
15. Рабинович С.Г. Погрешности измерений. // Издательство «Энергия», 2008 – 262 с.
16. Методические указания по выполнению экономического (организационно-экономического) раздела выпускной квалификационной работы бакалавра (для студентов технических направлений подготовки)): учеб. пособие; Российский технологический университет (МИРЭА), 2019. - 45 с.
17. Энергосбытовая компания «Мосэнергосбыт», Тарифы на электрическую энергию для населения и приравненных к нему категорий потребителей на территории г. Москвы. URL:

https://www.mosenergosbyt.ru/website/faces/individuals/tariffs-n-payments/tariffs-msk/polnaya-versiya-tarifov/

18. Постановление Правительства РФ от 01.01.2002 N 1 (ред. от 28.04.2018) "О Классификации основных средств, включаемых в амортизационные группы". – [Электронный ресурс]. – Режим доступа: http://www.consultant.ru/document/cons_doc_LAW_34710/ (дата обращения 07.05.2019)
19. Постановление Правительства РФ от 28.11.2018 N 1426 «О предельной величине базы для исчисления страховых взносов на обязательное социальное страхование на случай временной нетрудоспособности и в связи с материнством и на обязательное пенсионное страхование с 1 января 2019 г». – [Электронный ресурс]. – Режим доступа: https://normativ.kontur.ru/document?moduleId=1&documentId=325340/ (дата обращения 07.05.2019).