**МИНИСТЕРСТВО ПРОМЫШЛЕННОСТИ И ТОРГОВЛИ**

**ТВЕРСКОЙ ОБЛАСТИ**

**Государственное бюджетное профессиональное образовательное учреждение**

**«Вышневолоцкий колледж»**

**ТЕХНИЧЕСКОЕ ЗАДАНИЕ**

На тему: **«Программа «Автоматизированная информационная система на железнодорожном вокзале»»**

По специальности: 09.02.07 «Информационные системы и программирование»

По ПМ.02 «Учебная практика»

Выполнил студент группы П-46-20

Васильев Андрей Евгеньевич

Руководитель проекта:

Яковлева Жанна Сергеевна

г. Вышний Волочек

2023 г.

СОДЕРЖАНИЕ

[Введение 3](#_Toc152361203)

[1 Анализ предметной области 4](#_Toc152361204)

[2 Техническое задание 5](#_Toc152361205)

[2.1 Основание для разработки 5](#_Toc152361206)

[2.2 Назначение разработки 5](#_Toc152361207)

[2.3 Требования к программе 5](#_Toc152361208)

[2.3.1 Требования к функциональным характеристикам 5](#_Toc152361209)

[2.3.2 Требования к надежности 6](#_Toc152361210)

[2.3.3 Требования к составу и параметрам технических средств 6](#_Toc152361211)

[2.3.4 Требования к информационной и программной совместимости 6](#_Toc152361212)

[2.4 Требования к программной документации 6](#_Toc152361213)

[2.5 Стадии и этапы разработки 7](#_Toc152361214)

[2.6 Порядок контроля и приемки 8](#_Toc152361215)

[3 Проектирование системы «Расписание РЖД» 9](#_Toc152361216)

[3.1 Общие сведения case-средства bpWin 9](#_Toc152361217)

[3.2 Описание функциональной модели 10](#_Toc152361218)

[4 Проектирование базы данных «Расписание движение РЖД» 16](#_Toc152361219)

[4.1 Описание case-средства erWin 16](#_Toc152361220)

[4.2 Логическое проектирование базы данных системы 18](#_Toc152361221)

[4.3 Разработка структуры связей 22](#_Toc152361222)

[4.4 Нормализация отношений базы данных 24](#_Toc152361223)

[5 Обоснование выбора средств разработки 26](#_Toc152361224)

[6 Описание программы 28](#_Toc152361225)

[6.1 Общие сведения 28](#_Toc152361226)

[6.2 Функциональное назначение 29](#_Toc152361227)

[6.3 Описание логической структуры 29](#_Toc152361228)

[6.4 Используемые технические средства 45](#_Toc152361229)

[6.5 Вызов и загрузка 45](#_Toc152361230)

[6.6 Входные данные 45](#_Toc152361231)

[6.7 Выходные данные 45](#_Toc152361232)

[7 Программа и методика испытаний 46](#_Toc152361233)

[7.1 Объект испытаний 46](#_Toc152361234)

[7.2 Цель испытаний 46](#_Toc152361235)

[7.3 Требования к программе 46](#_Toc152361236)

[7.4 Требования к программной документации 46](#_Toc152361237)

[7.5 Средства и порядок испытаний 47](#_Toc152361238)

[7.6 Методы испытаний 47](#_Toc152361239)

[8 Описание применения 49](#_Toc152361240)

[8.1 Назначение применения 49](#_Toc152361241)

[8.2 Условия применения 49](#_Toc152361242)

[8.3 Описание задачи 49](#_Toc152361243)

[8.4 Входные и выходные данные 50](#_Toc152361244)

[Список использованных источников 51](#_Toc152361245)

**Введение**

В настоящее время ЭВМ широко применяется во многих отраслях деятельности человека. Информационные технологии сегодня – это важнейшая составляющая эффективной управленческой деятельности. Не одна фирма не может обойтись в своей работе без применения компьютеров, которые с успехом заменяют рутинную работу, выполнявшуюся ранее вручную, повышая эффективность работы любой фирмы. Информационные технологии находят применение и в железнодорожном комплексе/3/.

Железнодорожный комплекс имеет особое стратегическое значение. Железные дороги - это один из самых могучих и совершенных видов транспорта. Они являются главнейшими источниками внутренней силы, производительности и богатства каждой страны.

Железную дорогу можно назвать промышленным предприятием. Основной задачей, продуктом ее производства является совершение перевозок. Железные дороги являются срочным, регулярным и постоянным средством транспорта на далёкие расстояния.

Стратегическое использование информационных технологий может помочь работе справочного бюро вокзала, следующим образом:

- предоставить услуги, отвечающие потребностям людей, быстро и эффективно отвечая на их запросы;

- упростить и ускорить работу диспетчеров по ведению учета расписания движения поездов и повысить эффективность их работы.

Целью данного курсового проекта является автоматизация составления и ведения расписания движения поездов. Автоматизация позволит производить ввод, редактирование и просмотр содержимого базы данных, а также отвечать на запросы пользователя и составлять отчеты по расписанию движения поездов.

В процессе выполнения данной курсовой работы должна быть разработана автоматизированная информационная подсистема, обслуживающая диспетчеров, позволяющая автоматизировать часть процессов, происходящих на железнодорожном вокзале.

**1 Анализ предметной области**

Одним из направлений деятельности железнодорожного вокзала является ведение расписания движения поездов, содержащего информацию о станциях назначения, маршруте следования поезда, дате и времени прибытия, стоимости проезда, и стоимости постельного белья. Поезда бывают разных категорий, в состав которых входят вагоны. Стоимость проезда зависит от категории вагонов, поездов и дальности поездки. Поезда ходят по расписанию. Однако составить расписание движения поездов на заданный период времени является трудоёмкой задачей. При составлении графика движения поезда трудно заранее учесть все факторы, влияющие на движение поездов. Отсюда возникает необходимость управления и контроля движением поездов.

Принятие решений по оперативному управлению поездами возлагается на диспетчеров и дежурных по станциям, которые осуществляют мониторинг движения и контролируют прибытие, следование и отправление всех поездов на станции.

Часто выполнить точный оперативный прогноз развития обстановки вручную, учитывая все нюансы движения поездов, невозможно, особенно в условиях ограниченности времени на анализ ситуации и принятие решения. С целью повышения оперативной работы диспетчерских служб на практике принимают автоматизированные системы, предназначенные для составления расписания движения поездов и отслеживания его выполнения.

Разрабатываемая автоматизированная система предназначена для учета информации о проходящих поездах через станцию Пенза-1, включая информацию по номерам поездов, пунктам назначения и периодичности движения поездов.

Применение данной системы позволит оперативно получать информацию о маршрутах следования поездов, идущих до необходимой станции назначения, а также о номере поезда, дне и времени отправления и прибытия на станцию назначения, стоимость проезда в вагонах различных категорий, стоимость постельного белья.

Использование данной системы позволит сократить временные и трудовые затраты диспетчеров на составление расписания движения поездов, а также получить информацию по возможным маршрутам до требуемой, станции назначения, наличии свободных мест, стоимости проезда, и, соответственно, существенно повысить качество обслуживания пассажиров.

**2 Техническое задание**

**2.1 Основание для разработки**

Разработка ведется на основе задания

**2.2 Назначение разработки**

Автоматизированная информационная система предназначена для автоматизации процесса составления и ведения расписания движения поездов.

**2.3 Требования к программе**

**2.3.1 Требования к функциональным характеристикам**

Программное обеспечение системы должно позволять вести расписание движения поездов, формировать справки о движении поездов, проходящих через железнодорожную станцию Пенза-1 и стоимости проезда.

В базе данных должна храниться информация о:

* поездах: номер поезда, название, категория, состав;
* электропоездах: номер, название;
* вагонах: номер, категория, количество мест;
* конечных и промежуточных станциях: название станции, зона,
* зонах: номер, цена;
* расписании: время прибытия, время отправления, время в пути, периодичность;
* стоимости проезда: цена проезда.

Программное обеcпечение должно обеспечивать ввод, изменение и удаление необходимых данных.

Приложение должно по запросам пользователя выдавать сведения о поездах, на которых можно доехать до выбранной станции (номер поезда, название, категория); по указанному поезду выдавать стоимость проезда в различных категориях вагона. Также должно предоставлять информацию по выбранному поезду (номеру, названию): номер, название поезда, перечень станций, время отправления, время прибытия, время в пути, периодичность). Аналогично с электропоездами.

Приложение должно формировать отчёт со списком поездов до станции назначения, отчёт о времени отправления, прибытии, длительности пути и периодичности поезда, отчет о стоимости проезда.

**2.3.2 Требования к надежности**

Приложение должно выдавать сообщения о возникающих ошибках при неверном задании исходных данных, поддерживать диалоговый режим в рамках предоставляемых пользователю возможностей.

**2.3.3 Требования к составу и параметрам технических средств**

Приложение должно обеспечивать бесперебойную работу пользователя. Приложение должно выдавать сообщения о возникающих ошибках при неверном задании исходных данных, предоставлять возможность исправления ошибок и продолжения работы, а также поддерживать диалоговый режим в рамках предоставляемых пользователю возможностей.

**2.3.4 Требования к информационной и программной совместимости**

База данных должна быть создана в среде SQL, программа должна быть разработана в среде – Native client 2022 Программное обеспечение должно работать в операционных системах Windows 7/8/10.

**2.4 Требования к программной документации**

Приложение должно сопровождаться комплектом программных документов, включающих:

* техническое задание;
* описание программы;
* программу и методику испытаний;
* описание применения.

**2.5 Стадии и этапы разработки**

Разработка автоматизированной системы должна осуществляться в соответствии со стадиями и этапами разработки, приведенными в таблице 1.

Таблица 1 – Стадии и этапы разработки

|  |  |
| --- | --- |
| Стадии | Этапы работ |
| 1 Формирование требований к АС | 1.1 Обследование объекта и обоснование необходимости создания АС |
| 1.2 Формирование требований пользователя к АС |
| 1.3 Оформление отчёта о выполненной работе и заявки на разработку АС (тактико-технического задания) |
| 2 Разработка концепции АС | 2.1 Изучение объекта |
| 2.2 Проведение необходимых научно-исследовательских работ |
| 2.3 Разработка вариантов концепции АС, удовлетворяющего требованиям пользователя |
| 2.4 Оформление отчёта о выполненной работе |
| 3 Техническое задание | 3.Разработка и утверждение технического задания на создание АС |
| 4 Эскизный проект | 4.1 Разработка предварительных проектных решений по системе и её частям |
| 4.2 Разработка документации на АС и её части |
| 5 Технический проект | 5.1 Разработка проектных решений по системе и её частям |
| 5.2 Разработка документации на АС и её части |
| Стадии | Этапы работ |
|  | 5.3 Разработка и оформление документации на поставку изделий для комплектования АС и (или) технических требований (технических заданий) на их разработку |
|  | 5.4 Разработка заданий на проектирование в смежных частях проекта объекта автоматизации |
| 6 Рабочая документация | 6.1 Разработка рабочей документации на систему и её части |
| 6.2 Разработка или адаптация программ |
| 7 Ввод в действие | 7.1 Подготовка объекта автоматизации к вводу АС в действие |
| 7.2 Подготовка персонала |
| 7.3 Комплектация АС поставляемыми изделиями (программными и техническими средствами, программно-техническими комплексами, информационными изделиями) |
| 7.4 Строительно-монтажные работы |
| 7.5 Пусконаладочные работы |
| 7.6 Проведение предварительных испытаний |
| 7.7 Проведение опытной эксплуатации |
| 7.8 Проведение приёмочных испытаний |
| 8 Сопровождение АС | 8.1 Выполнение работ в соответствии с гарантийными обязательствами |
| 8.2 Послегарантийное обслуживание |

# 2.6 Порядок контроля и приемки

В целях контроля за правильностью решения задачи и работы системы, должно быть произведено тестирование программы, в котором должны быть учтены все возможные варианты исходных данных, отражающие работу всех частей программы.

Для тестирования программы должны быть разработаны тестовые примеры, охватывающие все функциональные возможности программы.

Приемка системы должна осуществляться при условии с правильной работы программы, определяемой по результатам выполнения тестовых примеров.

# 3 Проектирование системы «Расписание РЖД»

# 3.1 Общие сведения case-средства bpWin

Разработка функциональной модели системы осуществлена с помощью CASE - средства BPWin.

При проектировании информационных систем применяют структурный подход (функционально-модульный). В его основе лежат принципы функциональной декомпозиции. Это означает, что система описывается в терминах иерархии функций системы и передачи информации между функциями.

Процесс моделирования системы начинается с определения контекста, т.е. наиболее абстрактного уровня описания системы в целом. В контекст входит определение субъекта моделирования, цели и точки зрения на модель.

Под субъектом понимается сама система, при этом необходимо точно установить, что входит в систему, а что лежит за ее пределами. На определение субъекта системы будет существенно влиять позиция, с которой рассматривается система, и цель моделирования – вопросы, на которые построенная модель должна дать ответ.

Существуют три методологии создания моделей процессов: IDEF0 (функциональная модель), IDEF3 (WorkFlow Diagram), DFD (DataFlow Diagram), каждая из которых решает свои специфические задачи.

Наиболее удобной методологией моделирования бизнес-процессов являетсяIDEF0. В IDEF0 система представляется как совокупность взаимодействующих работ или функций. Такая чисто функциональная ориентация является принципиальной - функции системы анализируются независимо от объектов, которыми они оперируют. Это позволяет более четко смоделировать логику и взаимодействие процессов организации. Под моделью в IDEF0 понимают описание системы (текстовое и графическое), которое должно дать ответ на некоторые заранее определенные вопросы.

Диаграммы потоков данных (Data flow diagramming,DFD) используются для описания документооборота и обработки информации. Подобно IDEF0, DFD представляет модельную систему как сеть связанных между собой работ. Их можно использовать как дополнение к модели IDEF0 для более наглядного отображения текущих операций документооборота в корпоративных системах обработки информации. DFD описывает:

а) функции обработки информации (работы);

б) документы (стрелки, arrow), объекты, сотрудников или отделы, которые участвуют в обработке информации;

в) внешние ссылки (external references), которые обеспечивают интерфейс с внешними объектами, находящимися за границами моделируемой системы;

г) таблицы для хранения документов (хранилище данных, data store).

Для описания логики взаимодействия информационных потоков более подходит IDEF3, называемая также workflow diagramming - методологией моделирования, использующая графическое описание информационных потоков, взаимоотношений между процессами обработки информации и объектов, являющихся частью этих процессов. Диаграммы Workflow могут быть использованы в моделировании бизнес-процессов для анализа завершенности процедур обработки информации. С их помощью можно описывать сценарии действий сотрудников организации, например, последовательность обработки заказа или события, которые необходимо обработать за конечное время. Каждый сценарий сопровождается описанием процесса и может быть использован для документирования каждой функции.

# 3.2 Описание функциональной модели

Функциональная модель предназначена для описания существующих бизнес-процессов на предприятии/4/. Первым шагом описания функциональной организации процесса было создание вершины древовидной структуры диаграмм, представленной контекстной диаграммой. Эта диаграмма является самым общим описанием системы и отображает ее взаимодействие с внешней средой. Контекстная диаграмма реализована в нотации IDEF0, так как данная нотация служит средством для отражения прежде всего процессов. Диаграммы в IDEF0 представляют процесс (операцию или совокупность операций или действий), имеющих вход (данные или объекты, потребляемые или изменяемые процессом), выход (результат выполнения процесса, продукт процесса), управляющее воздействие (стратегии, процедуры, регламенты процесса) и механизмы (ресурсы, необходимые для выполнения процесса). На рисунке А.1 приложения А, приведена диаграмма вершины древовидной структуры модели.

Входными данными активности являются:

* сведения о категориях поезда;
* сведения о поездах и электропоездах;
* сведения о категориях вагона;
* сведения о вагонах;
* сведения о станциях;
* сведения о зонах;
* критерий запроса.

Выходными данными активности являются:

* отчет со списком поездов до станции назначения;
* отчет о времени отправления и прибытия, длительности пути и периодичности поездов;
* отчет о стоимости проезда.

Учет расписания движения поездов ведется в соответствии с правилами учета движения поездов (стрелка управления) оператором (механизм).

Декомпозиция активности осуществлена на основе нотации DFD.

В результате декомпозиции выделены следующие бизнес-процессы (работы):

1) занесение категорий поезда;

2) занесение сведений о поездах и электропоездах;

3) занесение категорий вагонов;

4) занесение сведений о вагонах;

5) занесение сведений о станциях;

6) занесение сведений о зоне;

6) составление расписания;

7) определение стоимости проезда;

8) формирование отчётов.

Функциональная декомпозиция работы «Автоматизированная информационная система расписания движения » проведена на основе результатов анализа предметной области с помощью методологии DFD (рисунок А.2). В результате декомпозиции выделено 8 бизнес-процессов (работ).

Входными данными процесса «Занесение категорий поезда» являются:

* сведения о категориях поезда.

Выходными данными процесса «Занесение категорий поезда» являются:

* данные о категориях поезда, которые заносятсят в хранилище «Категории поезда».

Входными данными процесса «Занесение сведений о поездах и электропоездах» являются:

* сведения о поездах и электропоездах;
* информация о категориях поезда.

Выходными данными процесса «Занесение сведений о поездах и электропоездах» являются:

* данные о поездах, которые заносятся в хранилище «Поезда и электропоезда».

Входными данными процесса «Занесение категорий вагонов» являются:

* сведения о категориях вагонов.

Выходными данными процесса «Занесение категорий вагонов» являются:

* данные о категориях вагонов, которые заносятся в хранилище «Категории вагонов».

Входными данными процесса «Занесение сведений о вагонах» являются:

* сведения о вагонах;
* данные о категориях вагонов.

Выходными данными процесса «Занесение сведений о вагонах» являются:

* данные о вагонах, которые заносятся в хранилище «Вагоны».

Входными данными процесса «Занесение сведений о станциях» являются:

* сведения о станциях.

Выходными данными процесса «Занесение сведений о станциях» являются:

* данные о станциях, которые заносятся в хранилище «Станции».

Входными данными процесса «Занесение сведений о зоне» являются:

* сведения о зонах.

Выходными данными процесса «Занесение сведений о зоне» являются:

* данные о зонах, которые заносятся в хранилище «Зоны».

Входными данными процесса «Составление расписания» являются:

- информация о поездах и электропоездах;

* информация о станциях.

Выходными данными процесса «Составление расписания» являются:

* информация о расписании поездов, которая заносится в хранилище «Расписание».

Входными данными процесса «Формирование стоимости» являются:

- информация о вагонах;

- информация о поездах и электропоездах;

* информация о станциях.

Выходными данными процесса «Формирование стоимости» являются:

* информация о стоимости проезда, которая заносится в хранилище «Стоимость».

Входными данными процесса «Формирование запросов к системе и получение отчётов» являются:

* данные о категориях поезда;
* информация о поездах и электропоездах;
* данные о категориях вагонов;
* информация о вагонах;
* информация о станциях;
* информация о зонах;
* данные расписания;
* информация о стоимости;
* критерий запроса.

Выходными данными процесса «Формирование запросов к системе и получение отчётов» являются:

* отчет cо списком поездов до станции назначения;
* отчет о времени отправления и прибытия, длительности пути и периодичности поездов;
* отчет о периодичности движения поездов;
* отчет о стоимости проезда.

Функциональная декомпозиция работы «Формирование отчётов» проводится на основе методологии DFD (рисунок А.3). В результате декомпозиции выделены следующие процессы (работ):

1) получение списка поездов до станции назначения;

2) получение информации об отправлении и прибытии поезда;

3) получение информации о длительности пути поезда;

4) получение информации о периодичности движения поездов.

5) получение информации о стоимости проезда.

Входными данными процесса «Получение списка поездов до станции назначения» являются:

-информация о станциях;

* критерий запроса.

Выходными данными процесса «Получение списка поездов до станции назначения» являются:

* список поездов.

Входными данными процесса «Получение информации об отправлении и прибытии поезда» являются:

* информация о поездах и электропоездах;
* информация о станциях;
* информация о расписании;
* критерий запроса.

Выходными данными процесса «Получение информации об отправлении и прибытии поезда» являются:

* время отправления и прибытия поезда.

Входными данными процесса «Получение информации о длительности пути поезда» являются:

* информация о поездах и электропоездах;
* информация о станциях;
* критерий запроса.

Выходными данными процесса «Получение информации о длительности пути поезда» являются:

* время в пути поезда.

Входными данными процесса «Получение информации о стоимости проезда» являются:

* информация о поездах и электропоездах;
* информация о станциях;
* информация о зонах;
* данные о категориях вагонов;
* данные о категориях поезда;
* информация о вагонах;
* информация о стоимости;
* критерий запроса.

Выходными данными процесса «Получение информации о стоимости проезда» являются:

* стоимость проезда.

Входными данными процесса «Получение информации о периодичности движения поездов» являются:

* информация о поездах;
* критерий запроса.

Выходными данными процесса «Получение информации о периодичности поезда» являются:

* периодичность движения поездов.

В результате анализа построенной модели системы декомпозиция проведена до элементарного уровня и достаточного для рязработки программного обеспечения системы. Поэтому дальнейшая декомпозиция выделенных процессов не требуется.

# 4 Проектирование базы данных «Расписание движение РЖД»

# 4.1 Описание case-средства erWin

Семейство продуктов ERwin относится к мощным персональным CASE-продуктам, предназначенным для моделирования баз данных самого различного типа. Отличительной чертой продуктов ERwin является высокая степень обеспечения согласованного взаимодействия между средствами создания баз данных и средствами разработки приложений в технологии клиент-сервер/2/.

С помощью ERwin разработчик может, используя визуальные средства, описать логическую модель данных. На основе логической модели создается физическая модель для конкретной СУБД с использованием хранимых процедур и триггеров. Результатом работы по созданию физической модели может стать генерация структуры базы данных.

Возможна также обратная разработка - создание модели данных на основе структуры базы данных. Этот процесс наиболее распространен в процессе перехода с одной технологии на другую (с файл-сервер на клиент-сервер), а также при смене сервера БД. На основе модели данных предоставляется возможность создавать отчеты, которые позволяют существенно упростить процесс документирования технического проекта.

Методологическую основу ERwin составляет технология IDEF1X (моделирование данных для реляционных СУБД). Результатом построения является ER-диаграмма ("сущность-связь"). Графический подход к созданию моделей значительно упрощает процесс разработки.

При проектировании структуры новой базы данных определяют сущности (объекты, явления) предметной области, которые должны найти свое отражение в базе данных. Объект – это такая абстракция множества предметов реального мира, что все экземпляры этого объекта имеют одни и те же характеристики и подчиняются одним и тем же правилам поведения. Объекты обладают определенными свойствами – атрибутами. Атрибут – это абстракция одной характеристики объекта. Каждый атрибут имеет имя и может получать значения из некоторого множества допустимых значений. Как правило, каждому объекту в базе данных соответствует таблица, а его атрибутам – поля этой таблицы.

# 4.2 Логическое проектирование базы данных системы

Логическая модель базы данных системы учета расписания движения поездов, построена на основе анализа предметной области. В связи с этим было выделено 9 сущностей:

- сущность «Категория поезда» определяется следующими атрибутами: код категории поезда, название;

- сущность «Поезда» определяется следующими атрибутами: номер поезда, название, состав, периодичность, код категории поезда;

- сущность «Электропоезда» определяется следующими атрибутами: номер электропоезда, название;

- сущность «Категория вагона» определяется следующими атрибутами: код категории вагона, название, стоимость постельного белья;

- сущность «Вагоны» определяется следующими атрибутами: номер вагона, номер поезда, количество мест, код категории вагона;

- сущность «Станции» определяется следующими атрибутами: код станции, название;

- сущность «Зона» определяется следующими атрибутами: номер зоны, название;

- сущность «Расписание» определяется следующими атрибутами: код записи, номер поезда, код станции, время прибытия, время отправления, время в пути;

- сущность «Стоимость проезда» определяется следующими атрибутами: номер поезда, номер вагона, код станции, цена.

Однозначно идентифицируем каждый экземпляр сущности - выделим первичные ключи.

Сущность «Категория поезда» - первичный ключ «Код категории поезда».

Сущность «Поезда» - первичный ключ «Номер поезда».

Сущность «Электропоезда» - первичный ключ «Номер электропоезда».

Сущность «Категория вагона» - первичный ключ «Код категории вагона».

Сущность «Вагоны» - первичный ключ «Номер вагона».

Сущность «Станции» - первичный ключ «Код станции».

Сущность «Зона» - первичный ключ «Номер зоны»

Сущность «Расписание» - первичный ключ «Код записи».

Свяжем таблицы через внешние ключи.

На этапе логического проектирования на основе семантического анализа предметной области осуществляется разбиение БД на таблицы для СУБД. При этом каждой сущности предметной области ставится в соответствие таблица, атрибутам объекта соответствуют атрибуты таблицы, а идентификатору объекта соответствует ключ таблицы.

Типы данных атрибутов сущностей приведены ниже в таблицах.

Таблица 1 - Типы данных атрибутов сущности «Категория поезда»

|  |  |
| --- | --- |
| Категория поезда | |
| Атрибут | Тип данных |
| Код категории поезда | Числовой, целый |
| Название | Текстовый (18) |

Таблица 2 - Типы данных атрибутов сущности «Поезда»

|  |  |
| --- | --- |
| Поезда | |
| Атрибут | Тип данных |
| Номер поезда | Текстовый (3) |
| Название | Текстовый (18) |
| Состав | Текстовый (18) |
| Периодичность | Текстовый (18) |
| Код категории поезда | Числовой, целый |

Таблица 3 - Типы данных атрибутов сущности «Электропоезда»

|  |  |
| --- | --- |
| Категория поезда | |
| Атрибут | Тип данных |
| Номер электропоезда | Числовой, целый |
| Название | Текстовый (18) |

Таблица 4 - Типы данных атрибутов сущности «Категория вагона»

|  |  |
| --- | --- |
| Категория вагона | |
| Атрибут | Тип данных |
| Код категории вагона | Числовой, целый |
| Название | Текстовый (18) |
| Стоимость постельного белья | Числовой, вещественный |

Таблица 5 - Типы данных атрибутов сущности «Вагоны»

|  |  |
| --- | --- |
| Вагоны | |
| Атрибут | Тип данных |
| Номер вагона | Текстовый (2) |
| Номер поезда | Текстовый (3) |
| Количество мест | Числовой, целый |
| Код категории вагона | Числовой, целый |

Таблица 6 - Типы данных атрибутов сущности «Станции»

|  |  |
| --- | --- |
| Станции | |
| Атрибут | Тип данных |
| Код станции | Числовой, целый |
| Название | Текстовый(18) |

Таблица 7 - Типы данных атрибутов сущности «Зона»

|  |  |
| --- | --- |
| Зона | |
| Атрибут | Тип данных |
| Номер зоны | Числовой, целый |
| Название | Текстовый(18) |

Таблица 8 - Типы данных атрибутов сущности «Расписание»

|  |  |
| --- | --- |
| Расписание | |
| Атрибут | Тип данных |
| Код станции | Числовой, целый |
| Номер поезда | Текстовый (3) |
| Код расписания | Числовой, целый |
| Время прибытия | Время |
| Время отправления | Время |
| Время в пути | Текстовый(18) |

Таблица 9 - Типы данных атрибутов сущности «Стоимость проезда»

|  |  |
| --- | --- |
| Стоимость проезда | |
| Атрибут | Тип данных |
| Код станции | Числовой, целый |
| Номер поезда | Текстовый (3) |
| Номер вагона | Текстовый (2) |
| Цена | Числовой, вещественный |

# 4.3 Разработка структуры связей

Между объектами предметной области существуют связи, которые должны быть отражены в виде связей между объектами инфологической модели. Графически связь обозначается линией, соединяющей связываемые объекты. В каждом направлении связи можно выделить главный объект, от которого идет связь, и подчиненный.

Различают идентифицирующую связь и не идентифицирующую связь. При установлении не идентифицирующей связи дочерняя сущность остается независимой. Экземпляр сущности родителя может существовать безотносительно к какому-либо экземпляру дочерней сущности.

Сущности «Категория поезда» и «Поезда» связаны через внешний ключ по полю «Код категории поезда». Так как одна категория может быть у разных поездов, а один поезд может быть одной категории, то эта связь будет «один-ко- многим». Тип связи – не идентифицирующая.

Сущности «Категория вагона» и «Вагоны» связаны через внешний ключ по полю «Код категории вагона». Так как одна категория может быть у разных вагонов, а один вагон может быть одной категории, то эта связь будет «один-ко-многим». Тип связи – не идентифицирующая.

Сущности «Поезда» и «Вагоны» связаны через внешний ключ по полю «Номер поезда». Так как один поезд может состоять из нескольких вагонов, а один и тот же вагон может быть только в одном поезде, то эта связь будет «один-ко-многим». Тип связи – идентифицирующая.

Сущности «Расписание» и «Поезда» связаны через внешний ключ по полю «Номер поезда». Так как по одному расписанию могут ходить несколько поездов, а один поезд может идти по одному расписанию, то эта связь будет «один-ко-многим». Тип связи – идентифицирующая.

Сущности «Расписание» и «Электропоезда» связаны через внешний ключ по полю «Номер электропоезда». Так как по одному расписанию могут ходить несколько электропоездов, а один электропоезд может идти по одному расписанию, то эта связь будет «один-ко-многим». Тип связи – не идентифицирующая. Тип связи – идентифицирующая.

Сущности «Расписание» и «Станции» связаны через внешний ключ по полю «Код станции». Так как одно расписание может относиться к нескольким станциям, а одна станция может относиться к одному расписанию, то эта связь будет «один-ко-многим». Тип связи – идентифицирующая.

Сущности «Станции» и «Зона» связаны через внешний ключ по полю «Номер зоны». Так как одна станция может относиться к к одной ценовой зоне, а одна зона может включать несколько станций, то эта связь будет «один-ко-многим». Тип связи – не идентифицирующая.

Сущности «Стоимость проезда» и «Категория вагона» связаны через внешний ключ по полю «Код категории вагона». Так как одна стоимость проезда может быть в нескольких категориях вагонах, а в одной категории вагона может быть одна стоимость проезда, то эта связь будет «один-ко-многим». Тип связи – идентифицирующая.

Сущности «Расписание» и «Стоимость проезда» связаны через внешний ключ по полю «Код расписания». Так как к одному коду расписания может относиться несколько стоимостей проезда, а одна стоимость проезда может относиться к одному коду проезда, то эта связь будет «один-ко-многим». Тип связи – идентифицирующая.

# 4.4 Нормализация отношений базы данных

Процесс нормализации выполняется путем анализа отношений с учетом их первичного ключа и существующих функциональных зависимостей. Он включает ряд правил, которые используются для проверки отношений, чтобы вся база данных могла быть нормализована до желаемой степени нормализации. Если некоторое требование не удовлетворяется, то должна быть проведена декомпозиция отношения на отношения, каждое из которых удовлетворяет всем требованиям нормализации.

При работе с реляционными базами данных обязательным является удовлетворение только требованиям первой нормальной формы (1НФ). Все остальные формы нормализации могут использоваться по желанию проектировщиков. Однако, рекомендуется выполнять нормализацию как минимум до 3НФ, чтобы получить хорошее качество отношений (приемлемую избыточность отношений).

Атрибут, значения которого атомарны (неделимы) называется простым атрибутом. Сложный атрибут получается соединением нескольких атомарных атрибутов, которые могут быть определены на одном или разных доменах (его также называют вектор или агрегат данных).

Ненормализованное отношение характеризуется таблицей, содержащей в одном или нескольких столбцах, повторяющиеся группы данных - сложные атрибуты (массив или вектор значений).

Отношение, в котором на пересечении каждой строки и каждого столбца содержится атомарное (или единственное) значение, находится в 1НФ. При этом необходимо, чтобы отношение имело первичный ключ. Для преобразования ненормализованной таблицы в первую нормальную форму, следует найти в исходной таблице и устранить все повторяющиеся группы данных, используя декомпозицию этой таблицы.

Вторая нормальная форма применяется к отношениям с составными ключами, т.е. к таким отношениям, первичный ключ которых состоит из двух или больше атрибутов. Отношение с первичным ключом на основе единственного атрибута всегда находится в 2НФ.

Не ключевой атрибут функционально полно зависит от составного ключа если он функционально зависит от всего ключа в целом, но не находится в функциональной зависимости от какого-либо из входящих в него атрибутов.

Отношение находится в 3НФ, если оно представлено в 2НФ и не имеет не входящих в первичный ключ атрибутов, которые находились бы в транзитивной функциональной зависимости от этого первичного ключа.

Нормализация 2НФ-отношения с образованием 3НФ-отношения осуществляется путем устранения транзитивных зависимостей - транзитивно-зависимые атрибуты удаляются из отношения и помещаются в новое отношение вместе с их детерминантом.

Все атрибуты всех девяти сущностей в данной проектируемой системе атомарны и находятся в 1 НФ. Рассмотрим для подтверждения данного факта сущность «Вагоны». Атрибуты данной сущности «Номер вагона», «Номер поезда», «Количество мест», «Код категории вагона» являются простыми и неделимыми.

2НФ требует, чтобы не ключевые атрибуты отношений зависели от первичного ключа в целом, но не от его части. 2НФ применяется к отношениям с составными ключами, т.е. к таким отношениям, первичный ключ которых состоит из двух и более атрибутов.

Рассмотрим отношение «Расписание», первичный ключ которого состоит из атрибутов «Код записи», «Код станции» и «Номер поезда». Не ключевые атрибуты данного отношения «Время прибытия», «Время отправления» и «Время в пути» зависят от данного первичного ключа в целом, а не от его части, поэтому можно сделать вывод, что данное отношение находится во 2НФ.

Ни в одном из отношений не существует транзитивных зависимостей, т.е. не ключевые атрибуты не зависят функционально друг от друга, отношения находятся в 3НФ.

# 5 Обоснование выбора средств разработки

Разработка автоматизированной информационной системы «Расписание РЖД» велась в среде под управлением СУБД InterBase .

Выбор Microsoft SQL Server в качестве среды разработки обусловлен тем, что данная среда обеспечивает:

* высокопроизводительный компилятор в машинный код;
* объектно-ориентированную модель компонент;
* визуальное (а, следовательно, и скоростное) построение приложений из программных прототипов;
* масштабируемые средства для построения баз данных.

Компилятор, встроенный в Microsoft SQL Server, обеспечивает высокую производительность, необходимую для построения приложений в архитектуре “клиент-сервер”. Этот компилятор в настоящее время является самым быстрым в мире, его скорость компиляции составляет свыше 120 тысяч строк в минуту на компьютере 486DX33. Он предлагает легкость разработки и быстрое время проверки готового программного блока. Кроме того, Microsoft SQL Server обеспечивает быструю разработку без необходимости писать вставки на Си или ручного написания кода (хотя это возможно).

В процессе построения приложения разработчик выбирает из палитры компонент готовые компоненты, еще до компиляции он видит результаты своей работы - после подключения к источнику данных их можно видеть отображенными на форме, можно перемещаться по данным, представлять их в том или ином виде. В этом смысле проектирование в Microsoft SQL Server мало чем отличается от проектирования в интерпретирующей среде, однако после выполнения компиляции мы получаем код, который исполняется в 10-20 раз быстрее, чем то же самое, сделанное при помощи интерпретатора. Кроме того, компилятор компилятору рознь, в Microsoft SQL Server компиляция производится непосредственно в родной машинный код, в то время как существуют компиляторы, превращающие программу в так называемый p-код, который затем интерпретируется виртуальной p-машиной. Это не может не сказаться на фактическом быстродействии готового приложения.

Основной упор в Microsoft SQL Server делается на максимальном ре использовании кода. Это позволяет разработчикам строить приложения весьма быстро из заранее подготовленных объектов, а также дает им возможность создавать свои собственные объекты для среды Microsoft SQL Server. Никаких ограничений по типам объектов, которые могут создавать разработчики, не существует. Действительно, все в Microsoft SQL Server написано на нем же, поэтому разработчики имеют доступ к тем же объектам и инструментам, которые использовались для создания среды разработки. В результате нет никакой разницы между объектами, поставляемыми Borland или третьими фирмами, и объектами, которые вы можете создать.

Объекты БД в Microsoft SQL Server включают в себя полную мощь Borland Database Engine. В состав Microsoft SQL Server также включен Borland SQL Link, поэтому доступ к СУБД Oracle, Sybase, Informix и InterBase происходит с высокой эффективностью. Кроме того, Microsoft SQL Server включает в себя локальный сервер Interbase для того, чтобы можно было разработать расширяемые на любые внешние SQL-сервера приложения в офлайновом режиме. Разработчик в среде Microsoft SQL Server, проектирующий информационную систему для локальной машины (к примеру, небольшую систему учета медицинских карточек для одного компьютера), может использовать для хранения информации файлы формата .dbf (как в dBase или Clipper) или .db (Paradox). Если же он будет использовать локальный InterBase for Windows 4.0 (это локальный SQL-сервер, входящий в поставку), то его приложение без всяких изменений будет работать и в составе большой системы с архитектурой клиент-сервер.

Выбор InterBase в качестве СУБД основан на том, что во-первых, эта СУБД «родная» для Microsoft SQL Server, а во-вторых, это – мощная СУБД, позволяющая выполнять запросы к ней одновременно нескольким пользователям, что позволяет использовать ее в архитектуре клиент-сервер.

Firebird 2.0 (FirebirdSQL) — компактная, кроссплатформенная, свободная система управления базами данных (СУБД), работающая на GNU/Linux, Microsoft Windows и разнообразных Unix платформах.

В качестве преимуществ Firebird можно отметить многоверсионную архитектуру, обеспечивающую параллельную обработку оперативных и аналитических запросов (это возможно потому, что читающие пользователи не блокируют пишущих), компактность (дистрибутив 5Mb), высокую эффективность и мощную языковую поддержку для хранимых процедур и триггеров.

Таким образом, использование СУБД Firebird 2.0 достаточно для создания базы данных. Её функциональные возможности позволяют полностью спроектировать базу данных со множеством нюансов.

# 6 Описание программы

# 6.1 Общие сведения

Автоматизированная система называется «Расписание РЖД». Главный проектный файл называется Poezda.exe. При работе программа использует базу данных, файл которой называется Inform.fdb.

Программа работает в операционных системах Windows 7/8/10.

# 6.2 Функциональное назначение

Разработанная автоматизированная система предназначена для автоматизации работ, связанных с ведением расписания движения поездов. Система предоставляет возможность ввода, редактирования и просмотр содержимого базы данных, а также составления отчетов. Она позволяет существенно упростить и ускорить составление расписания движения поездов. Также система повышает эффективность работы, за счет того, что система легка в обращении, позволяет хранить большое количество сведений в одной базе, автоматизирует некоторые процессы ввода и расчета, включает запросы, создает отчеты и др.

Разработанная автоматизированная система должна давать возможность диспетчерам легко и просто вести расписание движение поездов.

# 6.3 Описание логической структуры

В базе данных необходимо хранить информацию о поездах, об их категориях, об электропоездах, о вагонах, об их категориях, о станциях, о зонах, о расписании, о стоимости.

С помощью инструмента IBExpert была создана база данных. При этом в качестве сервера базы данных использовался удаленный сервер с именем localhost.

Файл базы данных имеет имя INFOORM.FDB. При создании базы данных были заполнены поля: сервер — удалённый, имя сервера - localhost, файл БД - INFORM.fdb, клиентская библиотека — fbclient.dll, имя пользователя — SYSDBA, пароль — RZHDinf, диалект БД — 3. При этом использован сервер Firebird версии 2.0.

Были созданы 9 таблиц: «POEZDA», «KAT\_POEZD», «EPOEZDA», «VAGONY», «KAT\_VAGON», «STANCII», «ZONA», «RASPISANIE», «STOIMOST».

Таблицы «KAT\_POEZD» и «POEZDA» соединены между собой внешним ключом KAT\_POEZD\_ID. Для этого был написан обработчик:

ALTER TABLE POEZDA ADD CONSTRAINT FK\_POEZDA\_1 FOREIGN KEY (KAT\_POEZD\_ID) REFERENCES KAT\_POEZD (KAT\_POEZD\_ID)

Таким же образом были соединены и другие таблицы.

Таблицы «KAT\_VAGON» и «VAGONY» соединены между собой внешним ключом KAT\_VAGON\_ID.

Таблицы «POEZDA» и «VAGONY» соединены между собой внешним ключом POEZD\_NUM.

Таблицы «RASPISANIE» и «POEZDA» соединены между собой внешним ключом POEZD\_NUM.

Таблицы «RASPISANIE» и «EPOEZDA» соединены между собой внешним ключом EPOEZDA\_ID.

Таблицы «RASPISANIE» и «STANCII» соединены между собой внешним ключом STANCIA\_ID.

Таблицы «STANCII» и «ZONA» соединены между собой внешним ключом ZONA\_ID.

Таблицы «STOIMOST» и «KAT\_VAGON» соединены между собой внешним ключом KAT\_VAGON\_ID».

Таблицы «RASPISANIE» и «STOIMOST» соединены между собой внешним ключом RASPISANIE\_ID».

И при изменении или удалении поля внешнего ключа (например, «Код категории поезда») выдаётся ошибка:

violation of FOREIGN KEY constraint "".

violation of FOREIGN KEY constraint "FK\_POEZDA\_1" on table "POEZDA".

Foreign key references are present for the record.

Программа разработана в среде Delphi 7.0.

Проект сохранен в отдельном каталоге Kurs на локальном диске C:\ под именем Poezda.dpr. В указанном каталоге также сохранены модули всех созданных форм.

Были разработаны следующие формы:

а) вход в систему;

б) категории поездов;

в) категории вагонов;

г) поезда;

д) электропоезда;

ж) станции;

з) ценовые зоны;

и) расписание поездов дальнего следования;

к) расписание пригородных поездов;

м) запросы по поездам дальнего следования;

н) запросы по пригородным поездам;

о) перекрёстный запрос по поездам;

п) диаграмма;

р) справка;

c) модуль данных ( представлен на рисунке Г32).

Форма «Вход в систему» появляется при запуске системы, и позволяет начать работу с системой, либо изменить данные.

Форма «Категории поездов» содержит названия категорий поездов, позволяет их изменять, добавлять и удалять.

Форма «Категории вагонов» содержит названия категорий вагонов, стоимость постельного белья, позволяет их изменять, добавлять и удалять.

Форма «Поезда» содержит сведения о поездах, позволяет их изменять, добавлять и удалять, она содержит подчиненную форму «Поезд».

Форма «Электропоезда» содержит номера и названия электропоездов позволяет их изменять, добавлять и удалять.

Форма «Станции» содержит список станций и ценовые зоны, позволяет изменять, добавлять и удалять записи, она содержит подчиненную форму «Станция».

Форма «Ценовые зоны» содержит названия зон и цены, позволяет х изменять, добавлять и удалять данные.

Форма «Расписание поездов дальнего следования» позволяет формировать записи расписания движения поездов дальнего следования, изменять и удалять их. Она содержит подчиненную форму «Расписание».

Форма «Расписание пригородных поездов» позволяет формировать записи расписания движения поездов дальнего следования, изменять и удалять их. Она содержит подчиненную форму «Расписание».

Форма «Запросы по поездам дальнего следования» позволяет выводить список поездов до выбранной станции, просматривать стоимость билета выбранного поезда. Также выводит список доступных поездов в выбранный промежуток времени отправления и до указанной станции.

Форма «Запросы по пригородным поездам» позволяет выводить список электропоездов до выбранной станции, просматривать стоимость билета выбранного электропоезда. Также выводит список доступных электропоездов в выбранный промежуток времени отправления и до указанной станции.

Форма «Перекрёстный запрос по поездам» позволяет выводить сведения по выбранному поезду: номер, название, станция назначения, время отправления и прибытия, время в пути, периодичность.

Форма «Диаграмма» позволяет увидеть диаграмму соотношения поездов дальнего следования и пригородных поездов.

Форма «Справка» содержит справочные сведения о системе.

В ходе выполнения курсового проекта были разработаны следующие отчеты:

а) отчёт со списком поездов до станции назначения;

б) отчёт о времени отправления и прибытия, длительности пути и периодичности поездов;

в) отчет о стоимости проезда.

Исполняемый файл — POEZDA.exe.

Текст модулей разработанной программы приведен в приложении В. Экранные формы представлены в приложении Г.

Модули, процедуры и функции данной программы перечислены в таблице 12.

Таблица 12 – Структура программы

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Модули | Процедуры и функции | Назначение | |  | |
| Form\_Main \_Unit  (модуль главной формы) | procedure N3Click(Sender: TObject) | Обработчик события закрытия главной формы при нажатии на кнопку «Выход» главного меню | |  | |
| procedure TForm\_Main.N2Click(Sender: TObject) | Обработчик события создания формы «Категории поездов» | |  | |
| procedure TForm\_Main.N4Click(Sender: TObject) | Обработчик события создания формы «Категории вагонов» | |  | |
| procedure TForm\_Main.N5Click(Sender: TObject) | Обработчик события создания формы «Электропоезда» | |  | |
| procedure TForm\_Main.N1Click(Sender: TObject) | Обработчик события создания формы «Сростав Поездов» | |  | |
| procedure TForm\_Main.N8Click(Sender: TObject) | Обработчик события создания формы «Зона» | |  | |
| procedure TfmMain.N7Click(Sender: TObject); | Обработчик события создания формы «Станции» | |  | |
| procedure TfmMain.N10Click(Sender: TObject); | Обработчик события создания формы «Расписание поездов дальнего следования» | |  | |
| procedure TfmMain.N14Click(Sender: TObject); | Обработчик события создания формы «Расписание пригородных поездов» | |  | |
| procedure TfmMain.N15Click(Sender: TObject); | Обработчик события вызова формы «Запросы для поездов дальнего следования» | |  | |
| procedure TfmMain.N16Click(Sender: TObject); | Обработчик события вызова формы «Запросы для пригородных поездов» | |  | |
| procedure TfmMain.N20Click(Sender: TObject); | Обработчик события вызова формы «Перекрёстный запрос для поездов» | |  | |
| procedure TfmMain.N17Click(Sender: TObject); | Обработчик события вызова формы «График» |  | | |
| procedure TfmMain.N19Click(Sender: TObject); | Обработчик события вызова отчета со списком поездов до станции назначения Otchet1 |  | | |
| procedure TfmMain.N23Click(Sender: TObject); | Обработчик события вызова отчета о времени отправления и прибытия, длительности пути и периодичности поездов Otchet2 |  | | |
| procedure TfmMain.N22Click(Sender: TObject); | Обработчик события вызова формы «Справка» |  | | |
|  |  |  | | |
| Form\_ElPoezda\_Unit  (модуль данных об электропоездах) | procedure TfmElPoezda.FormClose(Sender: TObject; var Action: TCloseAction); | Обработчик события закрытия формы «Электропоезда», осуществляет закрытие сеанса связи с базой данных | |  | |
| DM\_Unit  (модуль данных) | procedure TdmVokzal.dst\_vagonyNewRecord(DataSet: TDataSet); | Обработчик события добавления новой записи в таблицу «Вагоны», при котором новому значению идентификатора присваивается текущий | | |  |
| procedure TdmVokzal.dst\_stoimost\_proezdaNewRecord(DataSet: TDataSet); | Обработчик события добавления новой записи в таблицу «Стоимость», при котором новым значениям идентификаторов присваиваются текущие | |  | |
| procedure TdmVokzal.dst\_stanciiFilterRecord(DataSet: TDataSet; | Обработчик события фильтрации по названию станции в запросах для поездов дальнего следования | |  | |
| procedure TdmVokzal.dst\_Stancii\_ElPoezdaFilterRecord(DataSet: TDataSet; | Обработчик события фильтрации по названию станции в запросах для пригородных поездов | |  | |
| procedure TdmVokzal.dst\_poezdaFilterRecord(DataSet: TDataSet; | Обработчик события фильтрации по названию поездов в перекрёстном запросе для поездов | |  | |
| Form\_Raspisanie\_Unit (модуль данных расписания поездов дальнего следования) | procedure TfmRaspisaniePoezd.buAddClick(Sender: TObject); | Обработчик события нажатия кнопки «Добавить». Осуществляет добавление записи данных расписания. | |  | |
| procedure TfmRaspisaniePoezd.buChangClick(Sender: TObject); | Обработчик события нажатия кнопки «Изменить» для редактирования данных расписания. | |  | |
| procedure TfmRaspisaniePoezd.buDelClick(Sender: TObject); | Обработчик события нажатия кнопки «Удалить». Осуществляет удаление записи расписания. | |  | |
| procedure TfmRaspisaniePoezd.buPriceClick(Sender: TObject); | Обработчик события нажатия кнопки «Оформить стоимость». Осуществляет оформление и расчёт цены билета. | |  | |
| procedure TfmRaspisaniePoezd.buPriceChangClick(Sender: TObject); | Обработчик события нажатия кнопки«Изменить стоимость» для редактирования стоимости проезда. | |  | |
| procedure TfmRaspisaniePoezd.FormClose(Sender: TObject; | Обработчик события закрытия формы | |  | |
| Form\_New\_Raspisanie\_Unit (модуль добавления  данных расписания поездов дальнего следования) | procedure TfmNewRaspis.DateTimePicker2Change(Sender: TObject); | Обработчик события изменения даты в поле DateTimePicker2. | |  | |
| procedure TfmNewRaspis.DateTimePicker4Change(Sender: TObject); | Обработчик события изменения даты в поле DateTimePicker4. | |  | |
| procedure TfmNewRaspis.FormCreate(Sender: TObject); | Обработчик события присваивания DBEdit1 полю БД дату DateTimePicker2 и DBEdit2 соответственно – DateTimePicker4. | |  | |
| Form\_Zona\_Unit  (модуль данных о зонах) | procedure TfmZona.FormClose(Sender: TObject; var Action: TCloseAction); | Обработчик события закрытия формы. Осуществляет закрытие сеанса связи с базой данных. | | |  |
| Form\_Stancii\_Unit  (модуль данных о станциях) | procedure TfmStancii.buAddClick(Sender: TObject); | Обработчик события нажатия кнопки «Добавить». Осуществляет добавление станций | | |  |
| procedure TfmStancii.buChangClick(Sender: TObject); | Обработчик события нажатия кнопки «Изменить». Осуществляет редактирование станций | | |  |
| procedure TfmStancii.buDelClick(Sender: TObject); | Обработчик события нажатия кнопки «Удалить». Осуществляет удаление станций | | |  |
| procedure TfmStancii.FormClose(Sender: TObject; var Action: TCloseAction); | Обработчик события закрытия формы. Осуществляет закрытие сеанса связи с базой данных. | | |  |
| Form\_Raspisanie\_ElPoezd\_Unit (модуль данных расписания пригородных поездов) | procedure TfmRaspisanieElPoezd.buAddClick(Sender: TObject); | Обработчик события нажатия кнопки «Добавить». Осуществляет добавление записи данных расписания. | | |  |
| procedure TfmRaspisanieElPoezd.buChangClick(Sender: TObject); | Обработчик события нажатия кнопки «Изменить» для редактирования данных расписания. | | |  |
| procedure TfmRaspisanieElPoezd.buDelClick(Sender: TObject); | Обработчик события нажатия кнопки «Удалить». Осуществляет удаление записи расписания. | | |  |
| procedure TfmRaspisanieElPoezd.FormClose(Sender: TObject | Обработчик события закрытия формы | | |  |
| Form\_Zapros2\_Unit  (модуль данных запроса для пригородных поездов) | procedure TfmZapros2.FormClose(Sender: TObject; var Action: TCloseAction); | Обработчик события закрытия формы. Осуществляет закрытие сеанса связи с базой данных. | | |  |
| procedure TfmZapros2.DBGrid1CellClick(Column: TColumn); | Обработчик события по клику выбранной строки таблицы | | |  |
| procedure TfmZapros2.DBGrid2CellClick(Column: TColumn); | Обработчик события по клику выбранной строки таблицы | | |  |
| procedure TfmZapros2.Time1Change(Sender: TObject); | Обработчик события изменения времени | | |  |
| procedure TfmZapros2.Time2Change(Sender: TObject); | Обработчик события изменения времени | | |  |
| procedure TfmZapros2.Edit1Change(Sender: TObject); | Обработчик фильтрации по названию станции | | |  |
| Form\_Graph\_Unit  (модуль построения диаграммы) | procedure TfmGraph.FormShow(Sender: TObject); | Обработчик события отображения формы. Осуществляет установление связи с БД и активирует запрос при запуске. | | |  |
| procedure TfmGraph.FormClose(Sender: TObject; var Action: TCloseAction); | Обработчик события закрытия формы. Осуществляет закрытие сеанса связи с базой данных. | | |  |
| Unit1  (модуль данных запроса для поездов дальнего следования) | procedure TfmZapros.FormClose(Sender: TObject; var Action: TCloseAction); | Обработчик события закрытия формы. Осуществляет закрытие сеанса связи с базой данных. | | |  |
| procedure TfmZapros.DBGrid1CellClick(Column: TColumn); | Обработчик события по клику выбранной строки таблицы | | |  |
| procedure TfmZapros.DBGrid2CellClick(Column: TColumn); | Обработчик события по клику выбранной строки таблицы | | |  |
| procedure TfmZapros.Time1Change(Sender: TObject); | Обработчик события изменения времени | | |  |
| procedure TfmZapros.Time2Change(Sender: TObject); | Обработчик события изменения времени | | |  |
| procedure TfmZapros.Edit1Change(Sender: TObject); | Обработчик фильтрации по названию станции | | |  |
| Form\_Poezda\_Unit  (модуль данных о поездах) | procedure TfmPoezda.FormClose(Sender: TObject; var Action: TCloseAction); | Обработчик события закрытия формы. Осуществляет закрытие сеанса связи с базой данных. | | |  |
| procedure TfmPoezda.buAddClick(Sender: TObject); | Обработчик события нажатия кнопки «Добавить» для добавления нового поезда | | |  |
| procedure TfmPoezda.buChangClick(Sender: TObject); | Обработчик события нажатия кнопки «Изменить» для изменения данных поезда | | |  |
| procedure TfmPoezda.BuDelClick(Sender: TObject); | Обработчик события нажатия кнопки «Удалить» для удаления данных | | |  |
| Form\_Kat\_Poezd\_Unit (модуль данныхо категориях поездов) | procedure TfmKat\_Poezd.FormClose(Sender: TObject; | Обработчик события закрытия формы. Осуществляет закрытие сеанса связи с базой данных. | | |  |
| Form\_Kat\_Vag\_Unit  (модуль данных о категориях вагонов) | procedure TfmKat\_Vag.FormClose(Sender: TObject; var Action: TCloseAction); | Обработчик события закрытия формы. Осуществляет закрытие сеанса связи с базой данных. | | |  |
| Form\_Perekr\_Unit (модуль данных перекрёстного запроса по поездам) | procedure TfmPerekr.Edit1Change(Sender: TObject); | Обработчик фильтрации по названию поезда | | |  |
| procedure TfmPerekr.DBGrid1CellClick(Column: TColumn); | Обработчик события по клику выбранной строки таблицы | | |  |
| procedure TfmPerekr.Button1Click(Sender: TObject); | Обработчик события нажатия на кнопку «Вывести отчёт», запрос на вывод отчёта по поездам | | |  |
| Form\_New\_Stoimost\_Unit (модуль данных о стоимости проезда) | procedure TfmStoimost.FormClose(Sender: TObject; var Action: TCloseAction); | Обработчик события закрытия формы. Осуществляет закрытие сеанса связи с базой данных. | | |  |
| procedure TfmStoimost.DBLookupComboBox1CloseUp(Sender: TObject); | Обработчик события выбора категории вагона из DBLookupComboBox1 | | |  |
| SpravUnit3  (модуль данныхсо со справкой) | - | - | | |  |
| Otchet1Unit  (модуль данных отчета) | procedure TOtchet1.FormClose(Sender: TObject; var Action: TCloseAction); | Обработчик события закрытия формы. Осуществляет закрытие сеанса связи с базой данных. | | |  |
| procedure TOtchet1.Button1Click(Sender: TObject); | Обработчик события нажатия на кнопку «Вывести». | | |  |
| Form\_New\_Poezda\_Unit (модуль добавления данных о поездах) | - | - | | |  |
| Form\_New\_Stancii\_Unit (модуль добавления данных о станциях) | - | - | | |  |
| Form\_New\_Vagon\_Unit (модуль добавления данных о вагонах) | - |  |  |  |  |

# 6.4 Используемые технические средства

Автоматизированная система разработана на ПК со следующими характеристиками:

* процессор Pentium 4 c тактовой частотой 2166 МГц;
* оперативная память - 2048 Мбайт;
* свободное дисковое пространство - 900 Мбайт;
* клавиатура;
* мышь;
* принтер.

# 6.5 Вызов и загрузка

В среде Microsoft SQL Server следует открыть проект с данной программой под названием Poezda. В папке с проектом, находится EXE файл под названием Poezda. Следует щелкнуть по нему два раза кнопкой мыши.

# 6.6 Входные данные

Входные данные системы:

а) по категориям поезда – код категории, название;

б) по поездам – номер поезда, название, состав, периодичность;

в) по категориям вагона – код категории, название, стоимость постельного белья;

г) по вагонам – номер вагона, количество мест;

д) по электропоездам – номер, название;

е) по зонам – номер зоны, название;

ж) по станциям – код станции, название;

з) по расписанию – код расписания, название поезда или электропоезда, название станции, время отправления, время прибытия, время в пути.

**6.7 Выходные данные**

Выходными данными должны быть ответы на запросы и следующие отчеты:

а) со списком поездов до станции назначения;

б) о времени отправления и прибытия, длительности пути и периодичности поездов;

в) о стоимости проезда.

**7 Программа и методика испытаний**

**7.1 Объект испытаний**

Объектом испытаний является программа – автоматизированная информационная система «Расписание движения поездов».

**7.2 Цель испытаний**

Цель испытания любой программы и в частности данной состоит в том, чтобы убедиться, что она удовлетворяет всем требованиям технического задания и действительно решает поставленную задачу.

**7.3 Требования к программе**

Автоматизированная информационная подсистема «Расписание движение поездов» должна обеспечивать выполнение следующих функций:

* учет сведений по поездам;
* учет сведений по вагонам;
* учет сведений по станциям;
* учет сведений по электропоездам;
* учет сведений по зонам;
* подсчет стоимости проезда на поезде/электропоезде до указанной станции/зоне;
* формирование отчетов.

# 7.4 Требования к программной документации

Программная документация к разрабатываемому проекту должна содержать:

* анализ предметной области;
* техническое задание;
* описание применения.

# 7.5 Средства и порядок испытаний

Для проведения испытаний необходимо наличие ПК с процессором Intel core i3 9100f и тактовой частотой 4,2 Ггц, оперативной памятью 8 ГБ, клавиатура, мышь, принтер. А также выход в интернет и интернет-браузер. Испытание системы должно проводиться на основе контрольно – отладочных примеров, результат которых будет выведен на печать с помощью кнопки Print Screen и вставлен в Microsoft Word для наглядной работы программы.

# 7.6 Методы испытаний

В процессе разработки программы была применена методология поэтапного тестирования, на каждом из этапов создания программы производилось её тестирование в пределах реализованных функциональных возможностей.

Тестирование программы предполагает проверку её работы на все возможных вариантах.

При тестировании системы входными данными являются:

Входными данными для системы являются:

а) данные о категориях поезда;

б) данные о поездах;

в) данные о категориях вагона;

г) данные о вагонах;

д) данные о электропоездах;

е) данные о зонах;

ж) данные о станциях;

Информационная система обеспечивает ввод, изменение и удаление этих данных.

Расписание отдельно ведётся для поездов дальнего следования и пригородных поездов. В расписании пригородных поездов обеспечивается добавление, изменение, удаление записи расписания. Аналогично для расписания пригородных поездов.

В пункте расписание для поездов дальнего следования система предоставляет возможность оформления и изменения стоимости проезда для выбранной записи расписания.

Перейдём к запросам, которые обеспечивает система. Выберем пункт меню запросы для поездов дальнего следования. Указываем необходимую нам станцию назначения (выбираем из списка или с помощью фильтра), система выдаёт доступные поезда. Далее выбираем интересующий поезд, и система показывает нам стоимость билетов в разных категориях его вагонов. Аналогично для электропоездов.

Система позволяет сформировать отчёт со списком поездов до станции назначения. Для этого в пункте меню отчёт выбираем список поездов до станции назначения, в открывшемся окне указываем станцию и нажимаем кнопку «Вывести».

В системе предусмотрен и перекрёстный запрос по поездам. В пункте меню запросы выбираем перекрёстный запрос. В открывшемся окне выбираем нужный нам поезд (из списка или с помощью фильтра), система выдаёт нам информацию о поезде, станции назначения, времени отправления и прибытия, времени в пути и периодичности поезда. Можем сформировать отчёт по поезду, нажав кнопку «Вывести отчёт».

Выбрав пункт меню график можем увидеть соотношение поездов дальнего следования и пригородных поездов в виде диаграммы.

При удалении любой записи в системе выдаётся сообщение о подтверждении удаления выбранной записи. Рассмотрим на примере расписание для поездов дальнего следования. В окне расписания выберем последнюю запись и нажимаем кнопку «Удалить». Появляется сообщение подтверждения удаления, нажимаем кнопку «ОК», и запись удаляется.

Таким образом, по результатам тестирования системы выявлено, что система функционирует корректно и обеспечивает все необходимые функции, заданные заказчиком.

**8 Описание применения**

**8.1 Назначение применения**

Автоматизированная информационная система предназначена для учета расписания движения поездов.

**8.2 Условия применения**

Для корректной работы системы необходимо наличие ПК с процессором Intel Core i3 9100f и тактовой частотой 4,2 Ггц, оперативной памятью 8 ГБ, а также необходимы клавиатура, мышь, принтер. Должен быть установлен интернет браузер.

**8.3 Описание задачи**

При запуске приложения перед пользователем появляется главное окно программы. Из него можно выбрать следующие пункты меню, кликнув по ним мышкой: управление поездами, станции, расписание, запросы, отчёт, график, помощь, выход.

Управление поездами содержит данные: категории поездов, категории вагонов, состав поездов, электропоезда, которые можно добавлять, изменять и удалять.

Станции содержит две вкладки: станции назначения и ценовые зоны, которые можно добавлять, изменять и удалять.

В расписании предусмотрено расписание для поездов дальнего следования и для пригородных поездов. В расписании для поездов дальнего следования обеспечивается добавление, изменение, удаление записей. Также есть возможность оформить и изменить стоимость. В расписании для пригородных поездов обеспечивается добавление, изменение, удаление записей. Оформление стоимости здесь не происходит, так как каждой станции соответствует своя ценовая зона.

Запросы предусмотрены для поездов дальнего следования и пригородных поездов, а также перекрёстный запрос по поездам. В запросах для поездов дальнего следования по указанной станции назначения (из списка или с помощью фильтра) выдаются доступные поезда. Далее после выбора интересующего поезда, показывается стоимость билетов в разных категориях его вагонов. Аналогично для электропоездов. В перекрёстном запросе по нужному поезду (из списка или с помощью фильтра), выдаётся информация о поезде, станции назначения, времени отправления и прибытия, времени в пути и периодичности поезда и по нажатию кнопки «Вывести отчёт» формируется отчёт о времени отправления и прибытия, времени в пути и периодичности поезда.

Отчёт позволяет по нажатию кнопки «Вывести» сформировать отчёт со списком поездов до указанной станции назначения.

График показывает соотношение поездов дальнего следования и пригородных поездов в виде диаграммы.

**8.4 Входные и выходные данные**

Входные данные системы:

а) по категориям поезда – код категории, название;

б) по поездам – номер поезда, название, состав, периодичность;

в) по категориям вагона – код категории, название, стоимость постельного белья;

г) по вагонам – номер вагона, количество мест;

д) по электропоездам – номер, название;

е) по зонам – номер зоны, название;

ж) по станциям – код станции, название;

з) по расписанию – код расписания, название поезда или электропоезда, название станции, время отправления, время прибытия, время в пути.

Выходными данными должны быть ответы на запросы и следующие отчеты:

а) со списком поездов до станции назначения;

б) о времени отправления и прибытия, длительности пути и периодичности поездов;

в) о стоимости проезда.

Заключение

В данной работе была разработана автоматизированная система «Расписание движения поездов».

Реализация системы проводилась в среде программирования Delphi 7.0, база данных реализована средствами утилиты IBExpert, СУБД Firebird 2.0.

При написании программы основное внимание было уделено удобству работы пользователя с программой и построению дружественного интерфейса.

Результаты тестирования показали, что программа работает верно во всех предполагаемых ситуациях.

**Список использованных источников**

Вендров А.М Проектирование программного обеспечения экономических информационных систем. – М.: Финансы и статистика, 2003. – 352с.

* www.interface.ru;
* www.wikipedia.ru;
* www.ms.by.ru;
* www.firebird-server.narod.ru.