МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН

Некоммерческое акционерное общество
АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ

кафедра _____

«Допущен к защите»
Заведующий кафедрой _____
(Ф.И.О., ученая степень, звание)
_____ «____» _____ **20__ г.**
(подпись)

ДИПЛОМНЫЙ ПРОЕКТ

На тему: Проектирование и разработка информационной системы для управления IT предприятием

Специальность Вычислительная техника и программное обеспечение

Выполнил (а) Алимбетов В. Э. ПО-12-2
(Фамилия и инициалы) группа

Научный руководитель Мусатаева Г. Т., ст. преп.
(Фамилия и инициалы, ученая степень, звание)

Консультанты:

по экономической части:
Бекишева А. Ч., к.э.н., доцент
(Фамилия и инициалы, ученая степень, звание)
_____ «24» 05 20 16 г.
(подпись)

по безопасности жизнедеятельности:
Приходько Н. Г., д.т.н., профессор
(Фамилия и инициалы, ученая степень, звание)
_____ «12» 05 20 16 г.
(подпись)

по применению вычислительной техники:
Мусатаева Г. Т. ст. преп.
(Фамилия и инициалы, ученая степень, звание)
_____ «20» 05 20 16 г.
(подпись)

Нормоконтролер: Мусатаева Г. Т., ст. преп.
(Фамилия и инициалы, ученая степень, звание)
_____ «30» 05 20 16 г.
(подпись)

Рецензент: Байтуленов Ж. Б. к.ф.-м.н., доцент
(Фамилия и инициалы, ученая степень, звание)
_____ «31» 05 20 16 г.
(подпись)

Алматы 2016 г.

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН

Некоммерческое акционерное общество
АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ

Факультет Информационных технологий
Специальность Вычислительная техника и программное обеспечение
Кафедра Компьютерных технологий

ЗАДАНИЕ
на выполнение дипломного проекта

Студент Алимбетов Руслан Ериковия
(фамилия, имя, отчество)

Тема проекта Проектирование и разработка информационной системы для управления в IT предприятии

утверждена приказом ректора №21 от «10» марта 2016 г.

Срок сдачи законченной работы «___» _____ 20___ г.

Исходные данные к проекту требуемые параметры результатов проектирования (исследования) и исходные данные объекта

Проектирование и разработка информационной системы для управления в IT предприятии

Перечень подлежащих разработке дипломного проекта вопросов или краткое содержание дипломного проекта:

1 Обзор существующих технологий и аналогов разрабатываемой программы
2 Разработка информационной системы
3 Технико-экономическая часть
4 Безопасность жизнедеятельности

Перечень графического материала (с точным указанием обязательных чертежей)

Иллюстрация примера главного интерфейса среду разработки

Иллюстрация примера кода

Иллюстрация главного меню и интерфейса разрабатываемой программы

Библиотеки классов базы данных

Библиотеки технико-экономической части

Иллюстрации из части безопасности жизнедеятельности

Рекомендуемая основная литература

Ситникова Е.Г. Проектирование баз данных. Методические указания к выполнению лабораторных работ (для студентов всех специальностей). - Алматы: АИЭС, 2008.

Питер Роб, Карлос Коронел Система баз данных: проектирование, реализация и управление. 5-ое изд. - СПб.: БХВ - Петербург, 2004

Сантэй Мишра, Алейн Бьюли Секреты Oracle SQL, изд Символ 2003

Консультанты по проекту с указанием относящихся к ним разделов

Раздел	Консультант	Сроки	Подпись
БЖД	Дуйсебекова И.Г.	07.03-13.03.16	
Экономическая часть	Бекенулова А.Ч.	24.04-24.05.16	
Политология	Мусатаева Г.Т.	15.03-15.05.16	Мусат
Картография	Мусатаева Г.Т.	30.05.16	Мусат
Разработка БД	Мусатаева Г.Т.	20.03.16	Мусат

Г Р А Ф И К
подготовки дипломного проекта

№ п/п	Наименование разделов, перечень разрабатываемых вопросов	Сроки представления руководителю	Примечание
1.	Обзор существующих технологий и аналогов разработанной программы	24.02.16	
2	Разработка блок диаграмм и приложение для обработки	17.03.16	
3	Технико-экономическое часть	24.04.16	
4	Безопасность жизнедеятельности.	19.05.16	

Дата выдачи задания «____»_____20___ г.

Заведующий кафедрой _____ Курилбиев З.К.
(подпись) (Фамилия и инициалы)

Руководитель _____ Мусаев _____ Мусаевич Р.Г.
(подпись) (Фамилия и инициалы)

Задание принял к исполнению
студент _____ Рустам _____ Алимбеков Р.Г.
(подпись) (Фамилия и инициалы)

Аңдатпа

Осы дипломдық жобада кәсіпорын IT мүлікін есепке алып басқару және IT инфрақұрылымы үшін деректер базасы қарастырылады. Пәндік облысы бойынша зерттеу жүргізілді, проблемалар мен оларды шешу жолдары анықталды.

Зерттеулер негізінде деректер базасының құрылымы құрылды. Сондай-ақ, ұсынылған базаны әзірлеу үшін еңбек шарттарын талдау орындалды.

Сонымен қатар жобанын экономикалық орындылығын растайтын экономикалық негіздемесі жасалды.

Аннотация

В данном дипломном проекте рассмотрена база данных для учета IT имущества и управления IT инфраструктурой на предприятии. Проведено исследование предметной области, выявлены проблемы и пути их решения. На основе исследований построена структура базы данных.

Также, выполнен анализ условий труда для разработки представленной базы.

Кроме того было составлено экономическое обоснование проекта, подтверждающее его экономическую целесообразность.

Annotation

In this diploma project I examined the development of a database for accounting IT equipment and IT infrastructure on the enterprise. A study of the subject area, identify problems and solutions. Based on studies of the information structure of the database is built.

Also the analysis of working conditions is made for the development of provided database.

In addition, the economic justification of the project is composed, confirming its economic viability.

Содержание

Введение

Информатизация общества и связанное с ней широкое распространение вычислительной техники и средств коммуникации выводят в ранг наиважнейших задачу создания специальных методов обработки данных: их поиск, защиту, обработку и хранение.

Большие объёмы информации практически невозможно проработать без специальных средств машинной обработки. В последнее время широкое распространение получили автоматизированные информационные системы: информационно-справочные, информационно-поисковые. Все они предназначены для регистрации, хранения и обработки данных с целью поиска и выдачи ответов на запросы пользователей. В большинстве случаев автоматизированные информационные системы разрабатывают как базы данных.

Базы данных, которые широко используются на практике, - это совокупность специальных методов и математических, информационных, программных, языковых, организационных и технических средств для поддержки динамической информационной модели предметной отрасли с целью обеспечения информационных запросов пользователей. Скорость доступа к информации, которая хранится в базах данных, удобство работы с ней зависит от организации структуры хранения данных, вида представления, возможностей поиска.

Актуальной становится задача проектирования и создания систем хранения и обработки информации с целью сокращения рутинного, малоэффективного человеческого труда. Широкое распространение вычислительной техники в разных сферах предприятия, промышленности, экономики, увеличение специалистов в данной области даёт реальную возможность для решения данной задачи.

Основные идеи современной информационной технологии базируются на концепции баз данных. Согласно данной концепции основой информационной технологии являются данные, организованные в БД, адекватно отражающие реалии действительности в той или иной предметной области и обеспечивающие пользователя актуальной информацией в соответствующей предметной области. Как сущности, атрибуты и связи отображаются на структуры данных - определяется моделью данных. Традиционно все СУБД классифицируются в зависимости от модели данных, которая лежит в их основе.

1 Анализ существующих технологий и аналогов программы

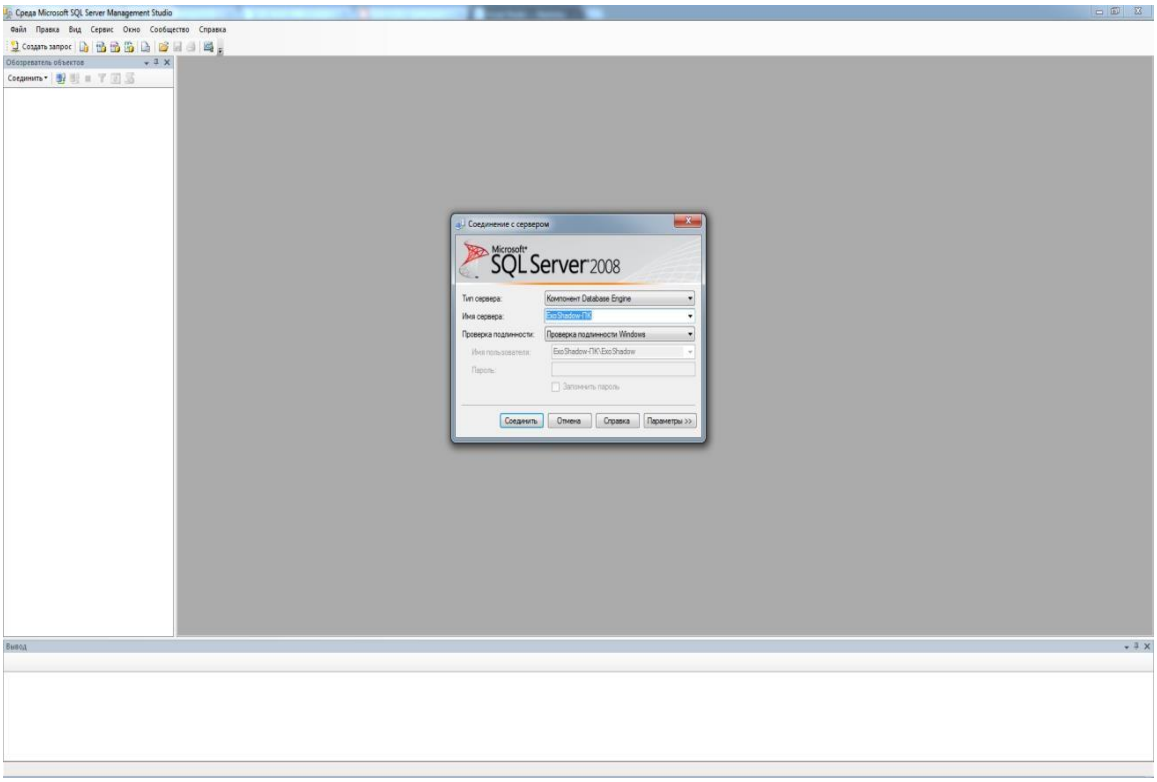
Данная работа представляет собой базу данных, для работы с которой необходим пользовательский интерфейс. Но начинать разработку информационной системы принято именно с базы данных. Для решения поставленной задачи на рынке существует несколько вариантов. Создание базы данных требует использования системы управления базами данных.

1.1 Системы управления базами данных

1.1.1 Система управления базами данных Microsoft SQL Server

Система управления реляционными базами данных от Microsoft является MS SQL. Для него был создан язык запросов Transact-SQL. В разработке языка запросов свои силы объединили Microsoft и Sybase. MS SQL используется для работы с базами данных, начиная от малого бизнеса до крупномасштабного предприятия.

Microsoft SQL Server 2005 представляет собой платформу для работы с базами данных, позволяет большой оперативной обработки транзакций (OLTP), хранения данных и запускать приложения для электронной коммерции; а также платформы бизнес-аналитики для создания решений по интеграции данных, анализа и отчетности. На рисунке 1.1 представлена начальная страница Microsoft



SQL Server Management Studio.

Рисунок 1.1 – Начальная страница Management Studio

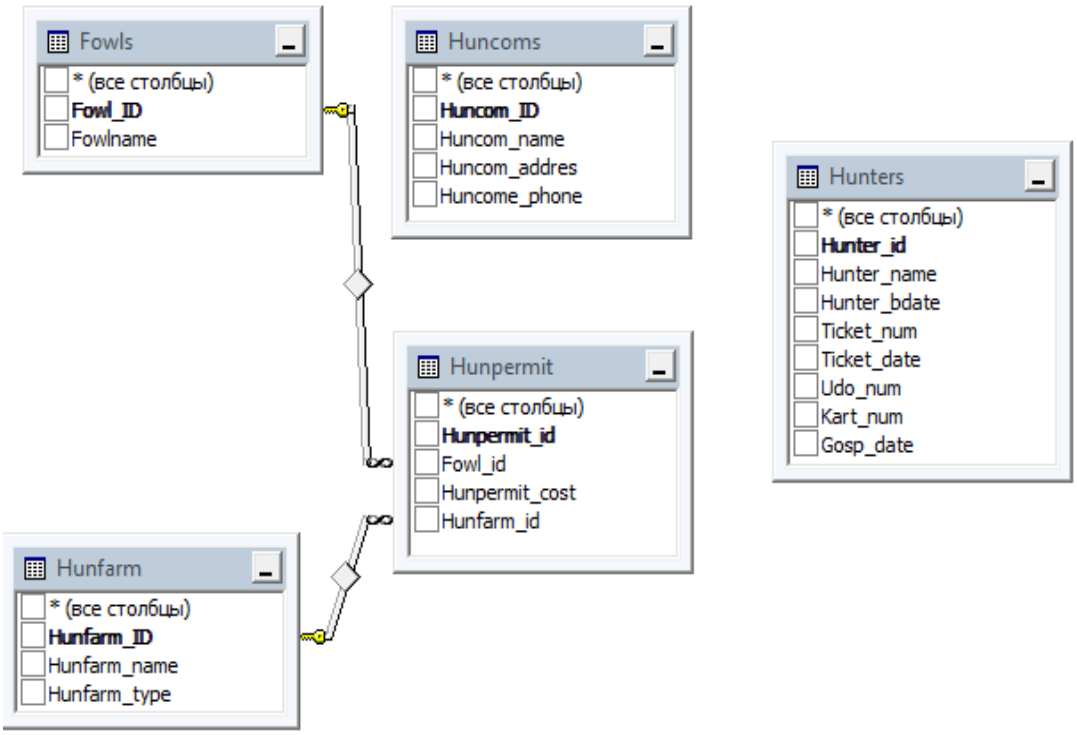
SQL Server 2005 предлагает множество способов отправки отзывов о продукте и документации, а также возможности автоматической отправки отчетов об ошибках и сведений об использовании функций в корпорацию Майкрософт. На рисунке 1.2 представлен вид окна соединения с сервером MS SQL Server Management Studio.

Рисунок 1.2 – Окно «Соединение с сервером»

MS SQL Server (до версии 7.0) Исходный код, основанный на коде Sybase SQL Server, и это позволило Microsoft выйти на рынок баз данных для предприятий, которые конкурировали Oracle, IBM, и, позже, она Sybase. Microsoft, Sybase и Ashton-Tate первоначально объединились для создания и выхода на рынок первой версии программы, получившей название SQL Server 1.0 для OS / 2 (около 1989), который был фактически эквивалентом Sybase SQL Server 3.0 для Unix, VMS , и другие. Microsoft SQL Server 4.2 был выпущен в 1992 году и был частью / 2 версии 1.3 операционной системы Microsoft OS. Официальный релиз Microsoft SQL Server версии 4.21 для операционной системы Windows NT была проведена одновременно с выпуском самой Windows NT (версия 3.1). Microsoft SQL Server 6.0 был первой версией SQL Server, созданной исключительно для архитектуры NT и без участия в процессе разработки Sybase.

К тому времени она вышла на рынке операционных систем Windows NT, Sybase и Microsoft разошлись и следить за их собственные образцы программных и маркетинговых схем. Microsoft искал эксклюзивные права на все версии SQL Server для Windows. Позже, Sybase изменил название своего продукта для Adaptive Server Enterprise, чтобы избежать путаницы с Microsoft SQL Server. До 1994 года,

Microsoft получила от трех Sybase уведомления об авторских правах, как намек на происхождение Microsoft SQL Server.Na. Рисунок 1.3 показывает пример



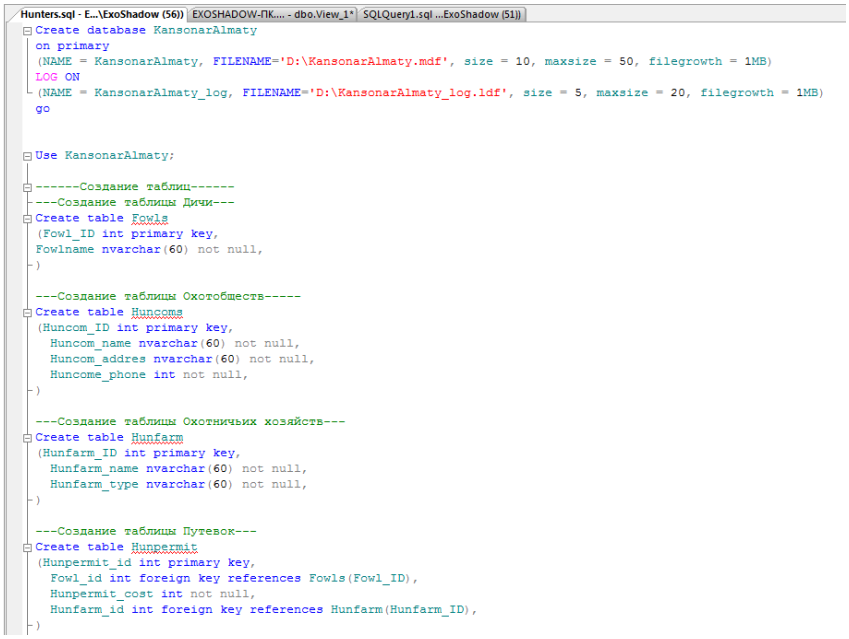
презентации создана при среды MS SQL Server Management Studio.образчик игры, сделанный около поддержки круга MS SQL Server Management Studio.

Рисунок 1.3 – Представление в MS SQL Server Management Studio

После разделения компании сделали несколько самостоятельных релизов программ. SQL Server 7.0 был первым сервером баз данных с настоящим пользовательским графическим интерфейсом администрирования. Для устранения претензий со стороны Sybase в нарушении авторских прав, весь наследуемый код в седьмой версии был переписан.

SQL Server 2005 - был представлен в ноябре 2005 г. Начиная с версии, которая состоялась с параллельного запуска Visual Studio 2005. Существует также "урезанная" версия Microsoft SQL Server - Microsoft SQL Server Express; она доступна для скачивания и может распространяться бесплатно, наряду с использованием своего программного обеспечения. С момента выхода предыдущей версии SQL Server (SQL Server 2000), было проведено развитие интегрированной среды разработки и целый ряд других подсистем, входящих в состав SQL Server 2005. Изменения коснулись реализации ETL технологий (вытяжку , преобразование и загрузка), часть компонента SQL Server Integration Services (SSIS), сервер уведомлений, инструментов аналитической обработки многомерных моделей данных (OLAP) и сбора соответствующей информации (обе службы являются частью служб анализа Microsoft) , а также несколько служб обмена мгновенными сообщениями, а именно, услуги брокера и услуги уведомления. Кроме того, были

внесены изменения в производительности. На рисунке 1.4 показан пример запроса к базе данных.



```
Create database KansonarAlmaty
on primary
(NAME = KansonarAlmaty, FILENAME='D:\KansonarAlmaty.mdf', size = 10, maxsize = 50, filegrowth = 1MB)
LOG ON
(NAME = KansonarAlmaty_log, FILENAME='D:\KansonarAlmaty_log.ldf', size = 5, maxsize = 20, filegrowth = 1MB)
go


Use KansonarAlmaty;

------Создание таблиц------
---Создание таблицы Дичи---
Create table Fowls
(Fowl_ID int primary key,
Fowlname nvarchar(60) not null,
)

---Создание таблицы Охотобществ-----
Create table Huncoms
(Huncom_ID int primary key,
  Huncom_name nvarchar(60) not null,
  Huncom_addres nvarchar(60) not null,
  Huncome_phone int not null,
)

---Создание таблицы Охотничьих хозяйств---
Create table Hunfarm
(Hunfarm_ID int primary key,
  Hunfarm_name nvarchar(60) not null,
  Hunfarm_type nvarchar(60) not null,
)

---Создание таблицы Путевок---
Create table Hunpermit
(Hunpermit_id int primary key,
  Fowl_id int foreign key references Fowls(Fowl_ID),
  Hunpermit_cost int not null,
  Hunfarm_id int foreign key references Hunfarm(Hunfarm_ID),
)
```

Рисунок 1.4 – Запрос к базе данных в MS SQL Server

Microsoft SQL Server Express считается безвозмездно диссеминируемой версией SQL Server, вырабатыванием порядка MSDE. Предоставленная издание владеет некие тех. лимитирования. Эти лимитирования совершают ее негодной про развертывания огромных двор этих, однако симпатия полностью хорошо про ведения программных ансамблей в размахах маленький фирмы. Охватывает полновесную помощь новейших видов этих, в книга количестве XML-спецификации. Практически, наверное настоящий MS SQL Server, подключая безвыездно его составляющие программирования, помощь государственных алфавитов и Unicode. Потому употребляется в прибавлениях, около конструировании либо про автономного исследования Microsoft SQL Server Express является бесплатно распространяемой версией SQL Server, развитием системы MSDE.

Эта версия имеет некоторые технические ограничения. Такие ограничения делают ее непригодной для развертывания больших баз данных, но она вполне пригодна для проведения программных систем через небольшой компании. Она содержит полную поддержку новых типов данных, в том числе XML-спецификации. На самом деле, это полный MS SQL Server, включая все компоненты программного обеспечения, поддержка международных символов и Unicode. Поэтому используется в приложениях, при проектировании или для самостоятельного изучения.

1.1.2 Система управления базами данных Oracle Database

Компания основана в 1977 году в городе Санта-Клара, Калифорния под наименованием SDL Ларри Эллисоном, Бобом Майнером и Эдом Оутсом. Это

кодовое название присвоили разработанной в первые месяцы существования SDL СУБД. Первый выпуск СУБД Oracle получил номер версии v2 по маркетинговым соображениям. Oracle v2 была написана на ассемблере для PDP-11, работала под управлением операционной системы RSX-11. В середине 1979 года авиабаза Райт-Патерсон ВВС США приобрела Oracle v2 и стала первым заказчиком компании. К этому же времени относится переименование SDL в Relational Software, Inc. Oracle v2 считается первой коммерческой СУБД с поддержкой языка запросов SQL, и одной из первых реляционных СУБД. Также отмечается влияние на Oracle ранее разработанной в IBM СУБД System R. На рисунке 1.5 представлена главное меню Oracle SQL Developer.

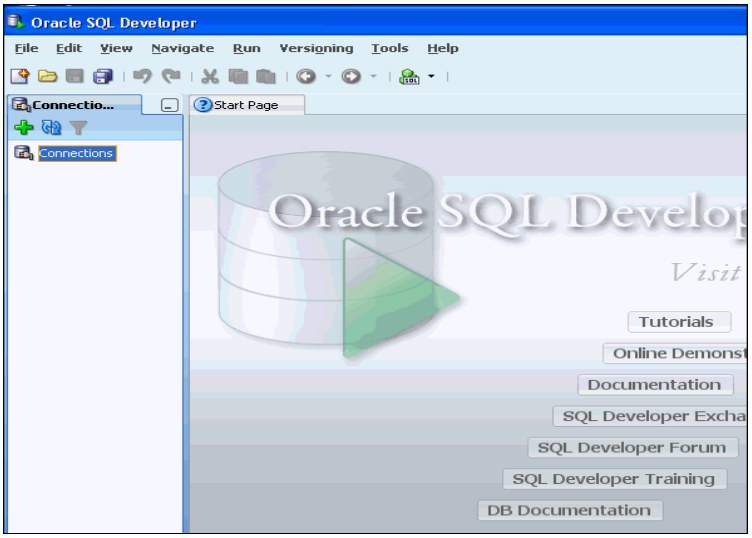


Рисунок 1.5 – Главное меню Oracle SQL Developer

Oracle — американская корпорация, второй по величине доходов производитель программного обеспечения, крупнейший производитель программного обеспечения для организаций, крупный поставщик серверного оборудования.

Фирма практикуется в выпуске порядков управления основаниями этих, связывающего программного снабжения и дело-прибавлений. Более узнаваемый работа фирмы — Oracle Database, кой фирма издаёт с эпизода собственного причины. С 2008 возраст компания изучила выработка встроенных аппаратно-программных ансамблей, а с 2009 возраст в итоге поглощения Sun Microsystems замерзла изготовителем серверного оснащения, по данного фирма издавала только программное снабжение.

Компания была основана в 1977 году Ларри Эллисон - соучредитель, генеральный директор с 1977 по 2014 год, крупнейшим акционером (25% по состоянию на 2014 г.). Корпорация Отдел расположены в более чем 145 странах. Что касается 2014 года составляет 122 тыс. Сотрудники. Корпоративная штаб-квартира расположена в США, штат Калифорния, недалеко от Сан-Франциско. Рисунок 1.6 представляет собой окно Oracle Database установку 12c.

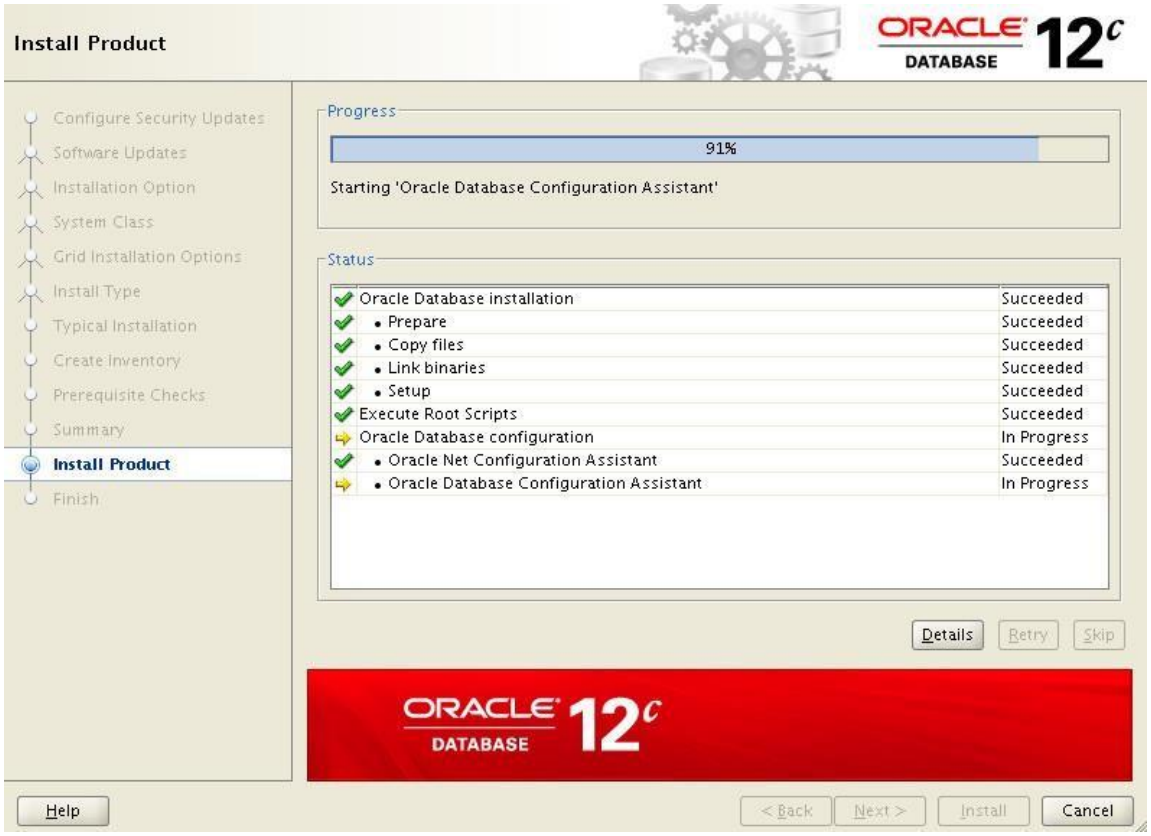


Рисунок 1.6 – Окно инсталляции Oracle Database 12c

Про веб - порядков и порядков масштаба большой системы предполагается работа Oracle9i Database Enterprise Edition , про что наличествует цельный комплект настроек, зодчески и высокофункционально расширяющих способности сервера. Работа Oracle9i Database Standard Edition нацелен в системы посредственного масштаба либо гарнизона в составе большой системы. Про индивидуального применения предполагается ”индивидуальный Oracle”, и про порядков подвижной взаимосвязи и маленьких кабинетов — Oracle9i Database Lite. В обычной, индивидуальной и подвижной редакциях главной упор изготовлен в низкую цену, несложность агрегата и компании. Около данном безвыездно виды сервера Oracle обладают в собственной базе вотан и оный ведь начальный адрес и высокофункционально схожи , из-за выпусканием неких доп настроек, коие нужны про специфичных конфигураций.

Основное преимущество такого подхода к построению СУБД — это идентичность кода для всех вариантов сервера баз данных. Для всех компьютерных платформ и архитектур существует единая СУБД Oracle, поставляемая в различных версиях, которая ведет себя одинаково и предоставляет одинаковую функциональность вне зависимости от платформы, на которой она установлена. Одной из основных характеристик СУБД Oracle является функционирование системы на большинстве платформ. В том числе на больших ЭВМ, UNIX-серверах, персональных компьютерах и т. д. На рисунке 1.7 представлено окно таблицы в Oracle SQL Developer

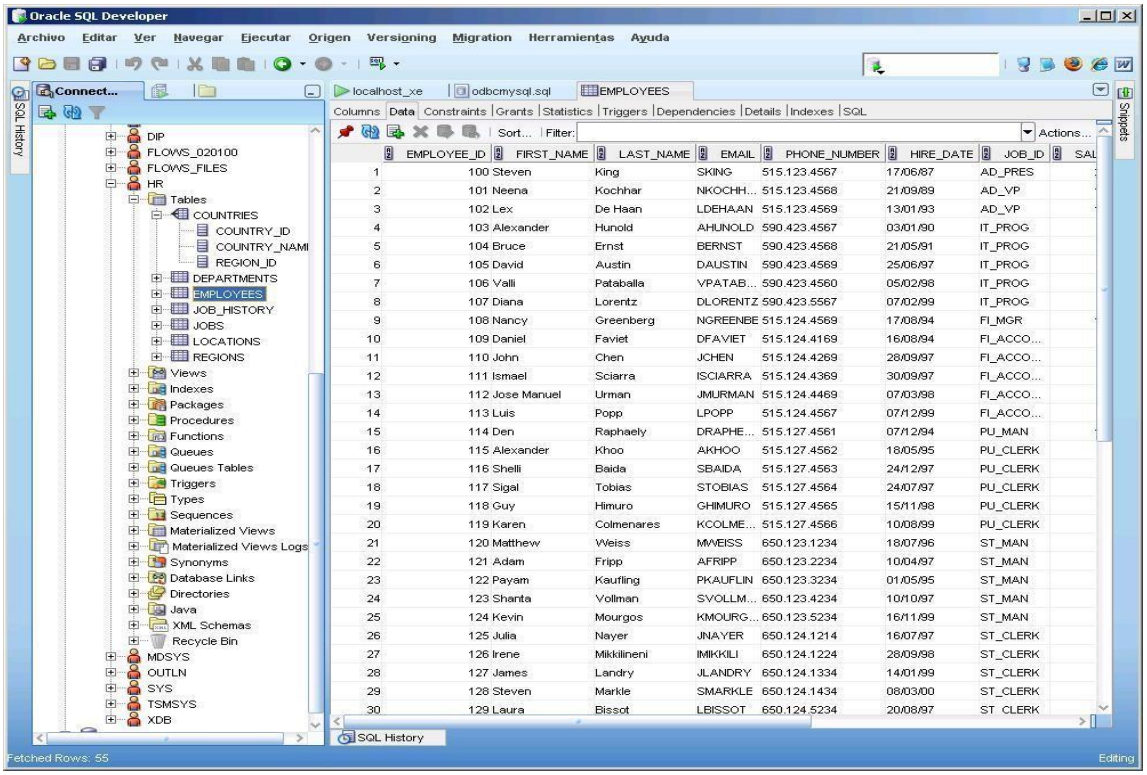


Рисунок 1.7 – Окно таблицы в Oracle SQL Developer

Иной принципиальной чертой считается помощь Oracle целых вероятных разновидностей зодчеств, в книга количестве инвариантных мультипроцессорных порядков, кластеров, порядков с глобальным параллелизмом и т. д. Явна престиж данных черт про крупномасштабных систем, в каком месте эксплуатируется очень много компов разных модификаций и изготовителей. В таковых критериях причиной фуррора считается очень вероятная стандартизация предлагаемых выводов, устанавливающая собственной мишенью немаловажное понижение цены собственности программным снабжением. Стандартизация порядков управления основаниями этих —вотан изо более важных деяний в дороги заслуги данной миссии.

1.2 Языки программирования

1.2.1 Язык программирования Java

Java - это объектно-ориентированный язык программирования, разработанный компанией Sun Microsystems. Java приложения обычно транслируются в специальный байт-код, поэтому они могут работать на любой виртуальной машине Java, независимо от компьютерной архитектуры. Дата официального релиза - 23 мая 1995. Первоначально называется Дубовая язык, разработанный Джеймсом Гослинг для устройств программирования бытовой электроники. Позже он был переименован в Java и был использован для написания клиентских приложений iservernogo программного обеспечения. Названный в честь марки кофе Java, который, в свою очередь, получила название острова с тем же

именем, поэтому официальная эмблема языка показывает чашку горячего кофе. Существует и другая версия происхождения названия языка, связанная с аллюзией на кофе-машину как пример бытового устройства, для программирования которого изначально язык создавался. На рисунке 2.1 представлено окно с примером кода на Java.

```
// Test for java ignoring flock

import java.io.*;
import java.nio.*;
import java.nio.channels.*;
import java.lang.management.*;

public class testlock {

  public static void main(String[]args) {
    try {
      RuntimeMXBean rt = ManagementFactory.getRuntimeMXBean();
      System.err.println("JVM: "+rt.getName());
      File f = new File("/opt/scripts/testlock/Huhu2");
      int count=0;
      while (count < 10) {
        try {
          RandomAccessFile raf = new RandomAccessFile(f, "rw");
          FileLock flock = raf.getChannel().tryLock();
          if (flock!=null && flock.isValid() && !flock.isShared()) {
            System.err.println("Got "+count+" lock on "+ f.getName() + " ... waiting 10 seconds");
            Thread.sleep(10000);
          }
          else {
           System.err.println("Failed1 to get lock on "+ f.getName());
          }
          raf.close();
        }
        catch (Exception ex) {
          System.err.println("Failed2 to get lock : "+ex.getMessage());
        }
        finally {
          count++;
        }
      }
    }
    catch (Exception ex) {
      ex.printStackTrace();
    }
  }

}
```

Рисунок 1.8 – Пример кода Java

Программы на Java транслируются в байт-код, выполняемый виртуальной машиной Java — программой, обрабатывающей байтовый код и передающей инструкции оборудованию как интерпретатор.

Плюсом сходственного метода исполнения кодов считается абсолютная самостоятельность б-заключение с операторной порядка и оснащения, будто дозволяет делать Java-прибавления в всяком приборе, про что есть соответственная условная машинка. Иной принципиальной индивидуальностью схемы Java считается эластичная конструкция сохранности, в рамках коей выполнение програмки вполне надзирается условной машинкой. Всевозможные акции, коие превосходят поставленные возможности програмки (к примеру, эксперимент неразрешенного прохода к этим либо синтеза с иным компом), призывают безотлагательное перебивание.

Часто к недостаткам концепции виртуальной машины относят снижение производительности. Ряд усовершенствований несколько увеличил скорость выполнения программ на Java:

-применение технологии трансляции байт-кода в машинный код непосредственно во время работы программы с возможностью сохранения версий класса в машинном коде,

- широкое использование платформенно-ориентированного кода стандартных библиотеках,

- аппаратные средства, обеспечивающие ускоренную обработку байт-кода.

На рисунке 1.9 представлено окно с примером автозаполнения в редакторе JavaScript.

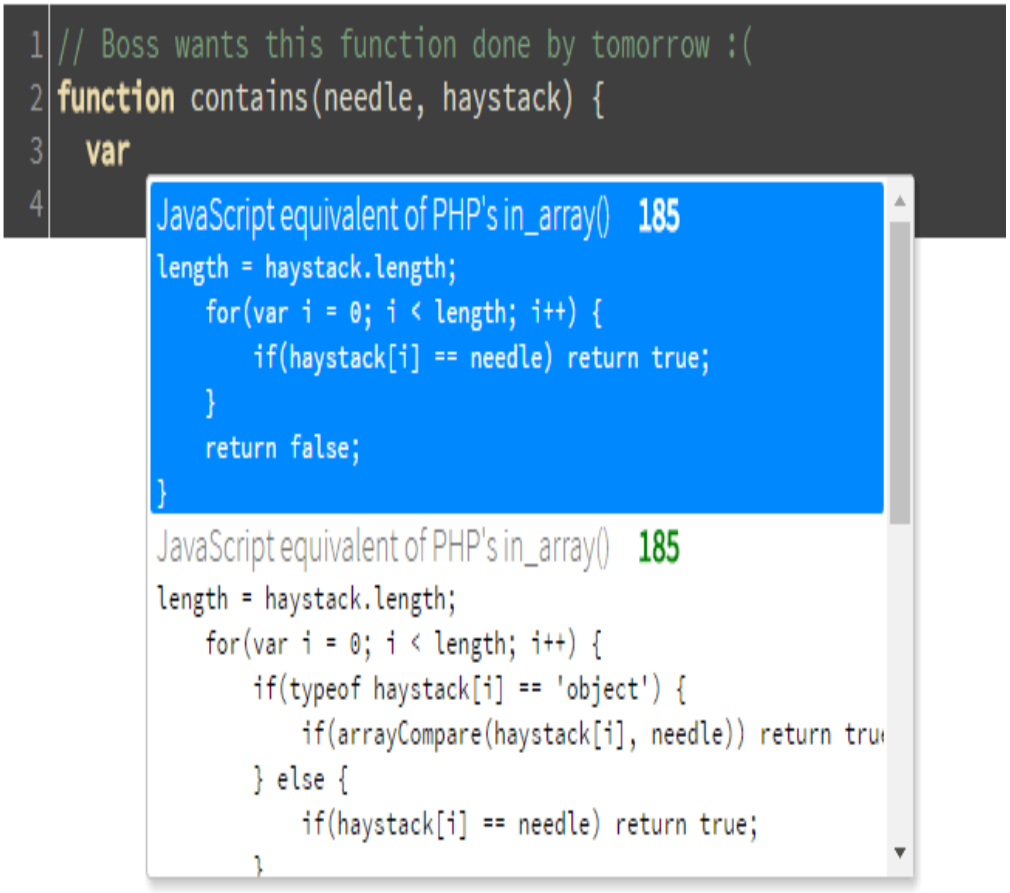
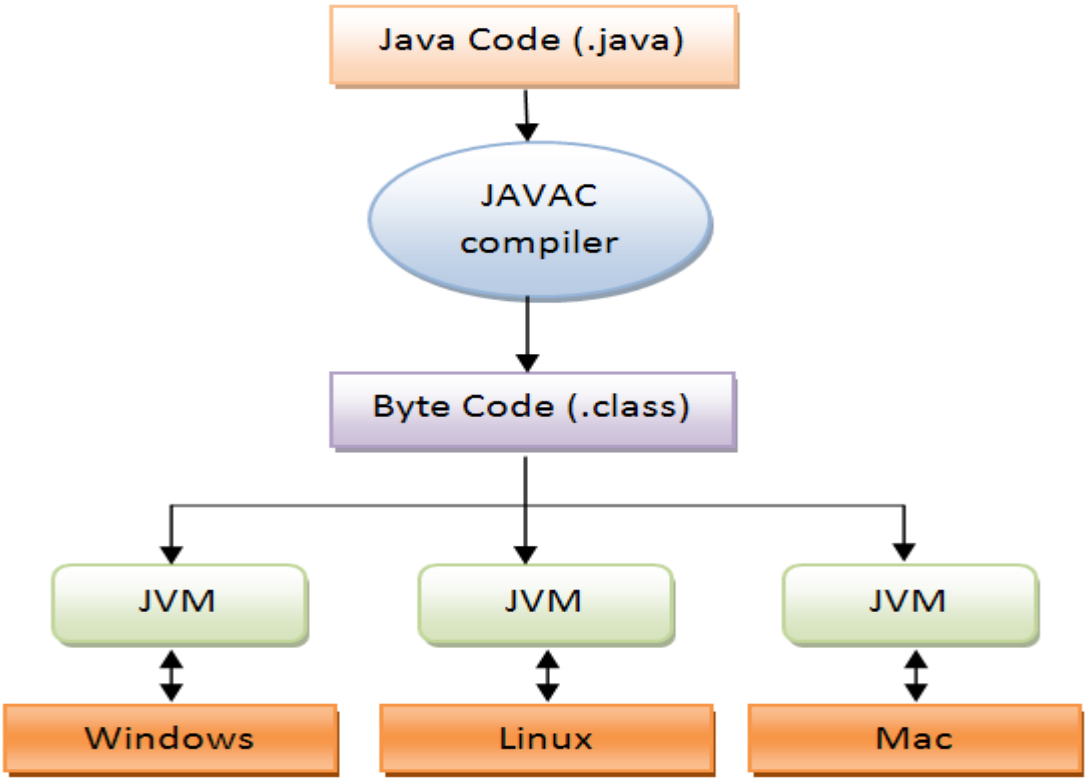


Рисунок 1.9 – Пример автозаполнения в редакторе JavaScript

Программы, написанные на Java, имеют репутацию медленнее и занимает больше памяти, чем те, которые написаны в C. Тем не менее, скорость выполнения



программ, написанных на языке Java, была значительно улучшена с выпуском 1997-1998 так называемого JIT-компилятор в версии 1.1 в дополнение к другим возможностям языка для поддержки лучше анализа кода. с 2000 года он использует виртуальную машину HotSpot - Кроме того, оптимизация виртуальной машины Java было сделано. По состоянию на февраль 2012 года, Java 7 код примерно в 1,8 раза медленнее, чем код, написанный на языке Си. Некоторые платформы предлагают аппаратную поддержку для реализации на Java. Например, микроконтроллеры выполнения Java-кода в аппаратных средствах вместо программного обеспечения виртуальной машины Java, а также на основе процессоров ARM, которые поддерживают выполнение Java-байт-код с помощью опции Jazelle. На рисунке 1.10 показана схема принципа работы виртуальной машины Java.

Рисунок 1.10 – Схема принципа работы JVM

Приблизительно в 1990 Джеймс Гослинг, Билл Джой, Патрик Ногтон и другие в Sun Microsystems начали разрабатывать язык по имени Oak. Прежде всего они видели применение Java для встроенных микрокомпьютерных модулей бытовой техники, в видеомагнитофонах, тостерах, а также для PDA. Однако, в 1993 рынки интерактивного телевидения и PDA пошли на убыль. Тогда бурно развивался internet и сети. Так что Sun сдвинула целевой рынок в сторону internet-приложений и заменила название проекта на Java. В 1994 Sun's выпустила браузер HotJava. Он был написан на Java за несколько месяцев, что показало мощность апплетов, программ которые работают в пределах браузера, а также для того, чтобы ускорить процесс разработки программ. Развивающийся наряду с огромным

интересом к internet, Java быстро получил широкое распространение, и ожидал роста для того, чтобы стать доминирующим языком программирования для написания приложений для потребителей и браузера. На рисунке 2.11 представлена внутренняя архитектура JVM.

Java - язык строго типизированный, поэтому он требует четкого указания метода. Java проверяет код во время компиляции и во время интерпретации. Таким образом, устранены некоторые типы ошибок при программировании. Java не имеет указателей и, таким образом, арифметические операции над ними. Все массивы данных и строки проверяются во время выполнения, что исключает возможность выхода за границу разрешается. Преобразование объектов из одного типа в другой также проверяется во время выполнения.

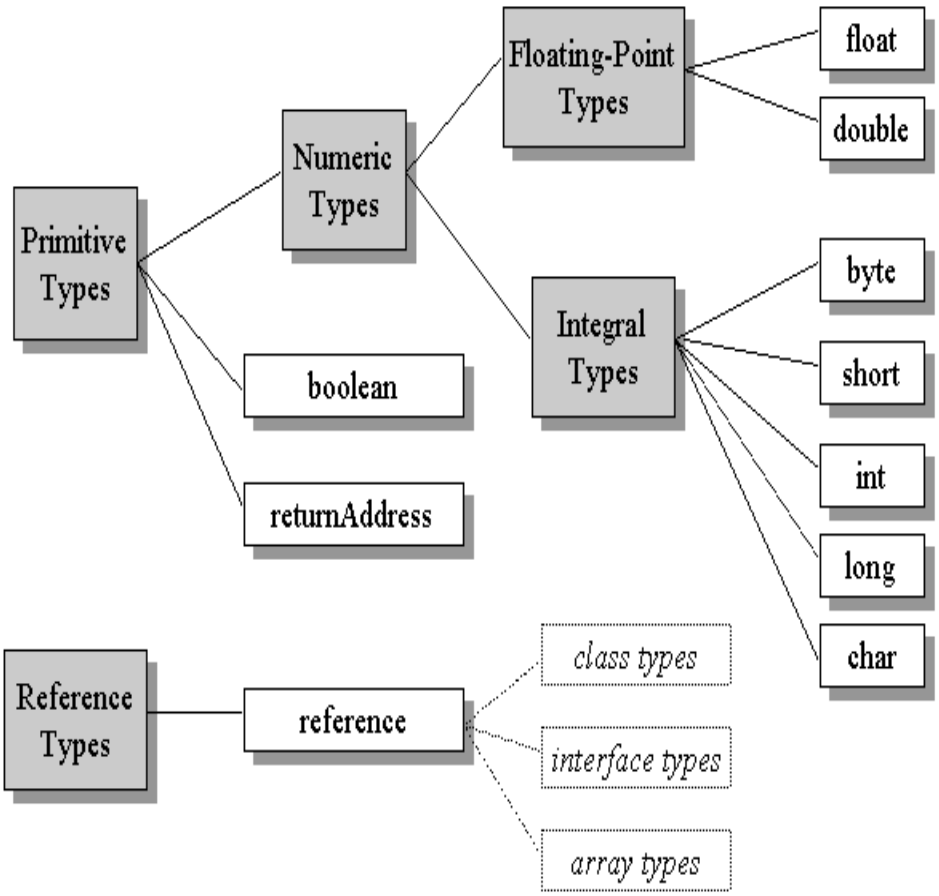


Рисунок 1.11 – Внутренняя архитектура JVM

Автоматическая обработка исключений - в традиционных средах программирования, программист должен был вручную распределять память, и в конце программы имел явное количество свободной памяти. Возникали проблемы,

когда программист забывал освобождать память. В Java, программист не должен беспокоиться о проблеме, связанной с освобождением памяти. Это делается автоматически, поскольку Java обеспечивает обработка исключений для объектов, которые не используются.

Java гарантирует контролируемую круг, в коей исполнена програмка. Java ни разу никак не подразумевает, будто адрес имеет возможность существовать невредно исполнен. И этак будто Java - более нежели язычок программирования, дьявол гарантирует некоторое количество значений контролирования охраны. С справкой данных значений, дьявол ручается не опасную круг исполнения.

Обработка исключений упрощает задачу обработки ошибок и восстановления. На рисунке 2.12 представлен пример взаимодействия 1С с Java.

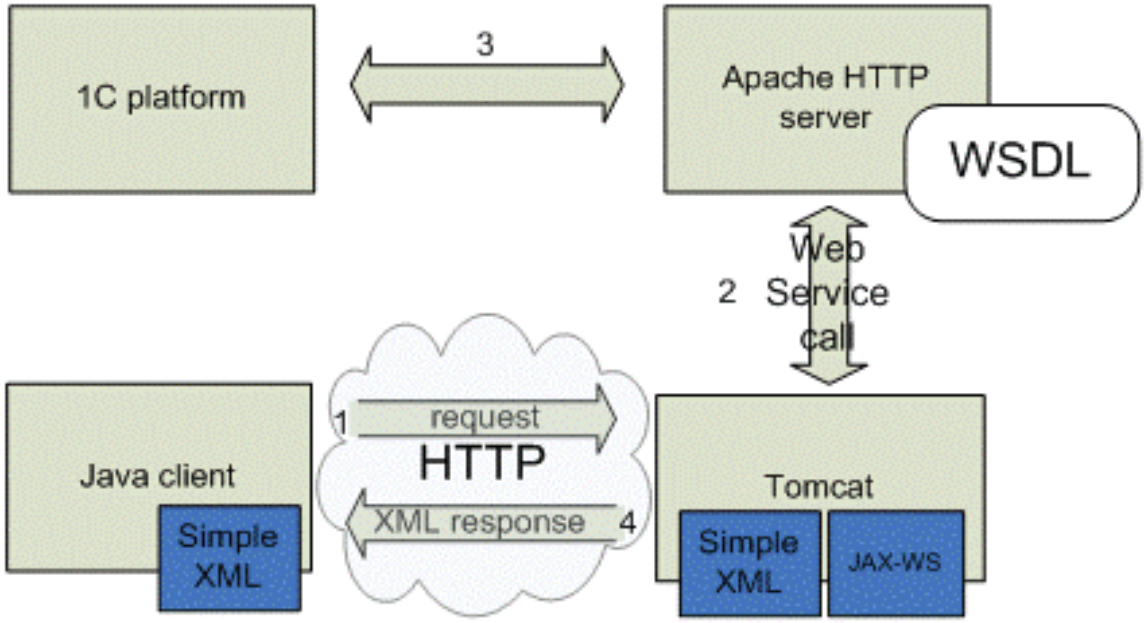


Рисунок 1.12 – Взаимодействие 1С с Java

1-ый уровень - наверное сохранность, богатая язычком Java. Характеристики и способы описываются в классе, и к ним разрешено устремиться лишь чрез междумордие, богатый классом. Java никак не дозволяет практически никаких акций с указателями, таковым ролью воспрещает непосредственный ход к памяти. Избегается захлестывание массивов. Трудности, сопряженные с сохранностью и мобильностью, укрыты.

2-ой уровень - автор, до этого нежели начать к компиляции заключение, испытывает сохранность заключение и потом надлежит в согласовании с протоколами, поставленными Java.

3-ий уровень - наверное сохранность, богатая интерпретатором. До этого, нежели б-адрес станет практически исполнен, дьявол считается вполне припрятанным верификатором.

4-ый уровень - хлопочет о загрузке классов. Металлург класса ручается, будто чин никак не преступает лимитирования прохода до этого, нежели дьявол загружен в порядок.

1.2.2. Язык программирования C#

C# — объектно-ориентированный язык программирования. Разработан в 1998—2001 годах группой инженеров под руководством Андерса Хейлсберга в компании Microsoft как язык разработки приложений для платформы Microsoft .NET Framework и впоследствии был стандартизирован как ECMA-334 и ISO/IEC 23270. На рисунке 1.13 представлена схема возможностей динамического программирования C#.

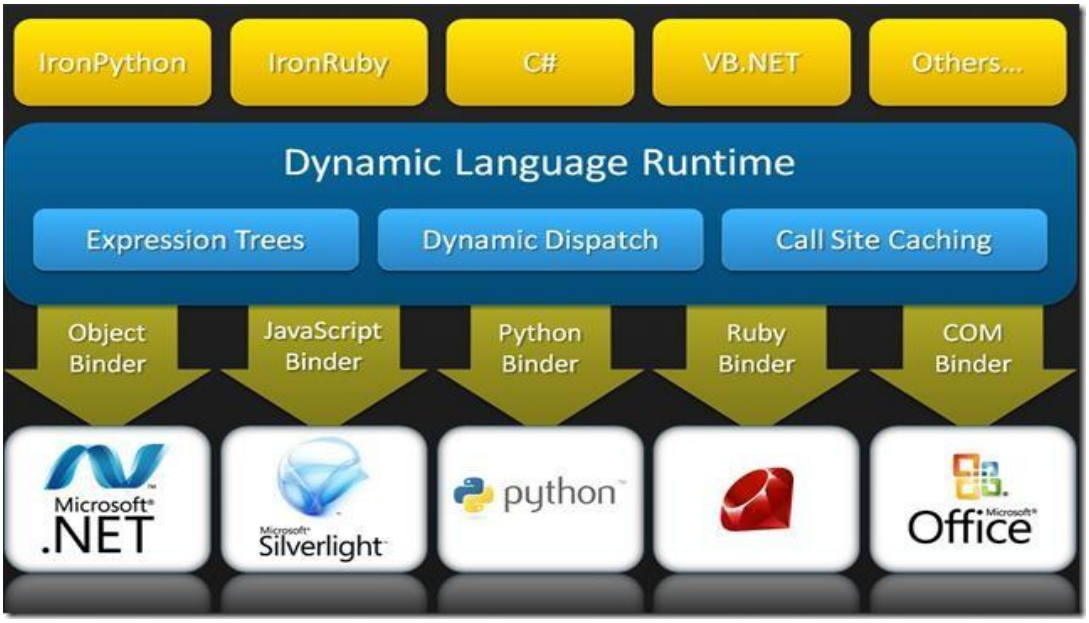


Рисунок 1.13 – Динамическое программирование на C#

C# относится к семье языков с C-подобным синтаксисом, из них его синтаксис наиболее близок к C# и Java. Язык имеет статическую типизацию, поддерживает полиморфизм, перегрузку операторов, делегаты, атрибуты, события, свойства, обобщённые типы и методы, итераторы, анонимные функции с поддержкой замыканий, LINQ, исключения,комментарии в формате XML.

Переняв многое от своих предшественников — языков C#, Pascal, Модула, Smalltalk и, в особенности, Java — C#, опираясь на практику их использования, исключает некоторые модели, зарекомендовавшие себя как проблематичные при разработке программных систем, например, C# в отличие от C# не поддерживает множественное наследование классов.

Название «Си шарп» происходит от музыкальной нотации, где знак диез означает повышение соответствующего ноте звука на полутон, что аналогично названию языка C#, где «++» обозначает инкрементпеременной. Название также является игрой с цепочкой C → C# → C#++(C#), так как символ «#» можно составить из 4х знаков «+».

Вследствие технических ограничений на отображение и того обстоятельства, что знак диез ♯ не представлен на стандартной клавиатуре, знак номера # был

выбран для представления знака диез при записи имени языка программирования. Это соглашение отражено в Спецификации Языка C# ECMA-334. Тем не менее, на практике, Майкрософт использует предназначенный музыкальный знак.

Названия языков программирования не принято переводить, поэтому язык следует называть по-английски «Си шарп». На рисунке 1.14 представлен отрезок кода на C#.

```
using System;                           // C# does not import a .h file, uses metadata
namespace FirstApplication              // scope for classes. No Obj-c counterpart
{
    class Person                        // only uses class implementation
    {
        private DateTime birthDate;     // a private field accessible to this class
        private int ageOn(DateTime date)   // a private method
        {
            TimeSpan span = date.Subtract(birthDate);  //uses a .notation to invoke
            return span.Days;
        }
        public int age                  // this is a property.
        {
            Get                         // just a getter; it's a read-only property
            {
                return this.ageOn(DateTime.Now);
            }
        }
        public Person( DateTime dob)    // instance constructor. Unlike Objective-C
        {                               // it combines allocation and initialization
            birthDate = dob;
        }
    }
    class Program                      //Unlike Obj-C, another class in the same file.
    {
        static void Main(string[] args) // main entry point into the program
        {
            Person p = new Person(new DateTime(1973,11,12));  //construct an instance
            System.Console.WriteLine("The age is  is" + p.age.ToString());
            DateTime dt = p.birthDate; //error in compilation birthDate is private
        }
    }
}
```

Рисунок 1.14 – Пример кода C#

Проект C# был начат в декабре 1998 и получил кодовое название COOL (C-style Object Oriented Language). Версия 1.0 была анонсирована вместе с платформой .NET в июне 2000 года, тогда же появилась и первая общедоступная бета-версия; C# 1.0 окончательно вышел вместе с Microsoft Visual Studio .NET в феврале 2002 года.Первая версия C# напоминала по своим возможностям Java 1.4, несколько их расширяя: так, в C# имелись свойства, индексаторы,события, делегаты, циклы foreach, структуры, передаваемые по значению, автоматическое преобразование встроенных типов в объекты при необходимости (boxing), атрибуты, встроенные средства взаимодействия с неуправляемым кодом (DLL, COM) и прочее.

Кроме того, в C# решено было перенести некоторые возможности C#, отсутствовавшие в Java: беззнаковые типы, перегрузку операторов, передача параметров в метод по ссылке, методы с переменным числом параметров, оператор goto. Также в C# оставили ограниченную возможность работы с указателями — в местах кода, специально обозначенных словом unsafe и при указании специальной опции компилятору.

1.5 Среда разработки Microsoft Visual Studio

Интегрированная среда Microsoft Visual Studio берет свое начало с 1995 года, когда была выпущена первая версия продукта под таким названием. До этого в начале 1990-х гг. Microsoft выпускала отдельные продукты для поддержки программирования на языках высокого уровня:

Все эти продукты были объединены с 1995 г. "под одной крышей" под названием Visual Studio. Новая среда получила сразу номер версии 4.0, видимо, потому, что в отдельном виде описанные выше компоненты среды уже выпускались до этого в течение нескольких лет. На рисунке 1.15 представлена начальная страница Visual Studio 2015.

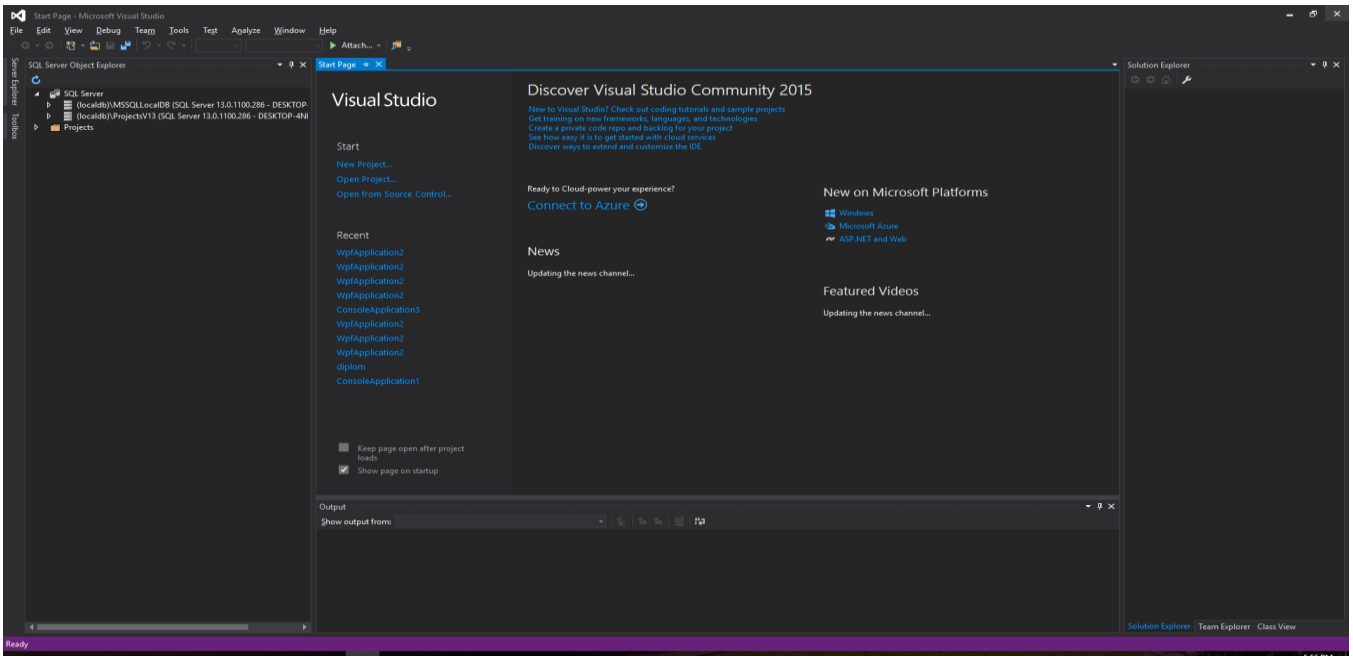
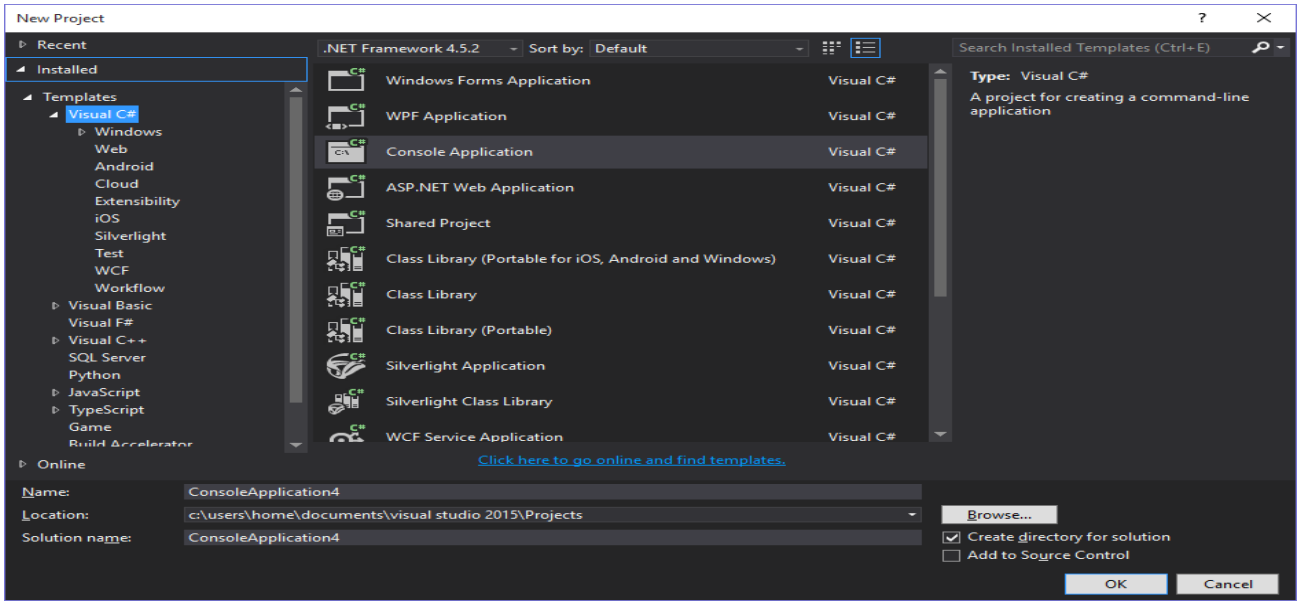


Рисунок 1.15 – Начальная страница Visual Studio 2015

Visual Studio подключает в себе вычитчик начального заключение с помощью схемы IntelliSense и вероятностью простого рефакторинга заключение. Интегрированный настройщик имеет возможность действовать будто настройщик значения начального заключение, этак и настройщик механического значения. Другие вделываемые приборы вводят в себе вычитчик фигур про упрощения сотворения графичного интерфейса прибавления, интернет-вычитчик, художник классов и художник схемы основания этих. Visual Studio дозволяет творить и включать посторонние добавления (плагины) про расширения функциональности



фактически в любом ватерпасе, подключая прибавление помощи порядков контролирования версий начального заключение (будто, к примеру, Subversion и Visual SourceSafe), прибавление новейших комплектов приборов (к примеру, про редактирования и зрительного проектирования заключение в конкретно-нацеленных слогах программирования) либо приборов про иных качеств движения исследования программного снабжения. Microsoft Visual Studio объединяет в себе огромное количество функций, позволяющих осуществлять разработки для Windows всех версий, в том числе и 8, Интернета, SharePoint, различных мобильных устройств и облачных технологий. В Visual Studio реализуется новая среда разработчика, благодаря которой создавать приложения стало проще. Microsoft Visual Studio - это обновленная и упрощенная программная среда, для которой характерна высокая производительность, причем она не зависит от особенностей оборудования. На рисунке 2.16 представлен вид окна создания проекта в Visual Studio 2015.

Рисунок 1.16 – Окно создания проекта

Любая новое издание программы складывается из свежайших приборов и технологий, дозволяющих упражнять прибавления с учетом необыкновенностей и позитивных факторов передовых платформ. К примеру, Visual Studio 2012 имеет возможность поддерживать наиболее ранние версии Windows XP и Windows Server 2003. Около данным разработчкиам раскрыта тропа к творению новейших и

модернизации теснее имеющихся прибавлений, специализированных про ранних версий ОС Windows. Нужно подметить, будто в движении применения поддерживаемых порядком разновидностей начальные комп.данные, планы и вывода в програмке Visual Studio станут трудоспособными, однако начальный адрес имеет возможность иметь необходимость в конфигурациях. С поддержкою наращенных лекарств прогнозирования, показывания и проектирования разрешено очень много обрисовать порядок, коия дозволит более успешно воплотить определенную теорию зодчества. Visual Studio отчуждает вероятность исследования прибавлений в C#, C#, Visual Basic и F#. Мощнейший графичный вычитчик разрешается творить проектирование прибавлений хоть какой трудности. Напрямик в кругу исследования может быть сотворения UML-диаграмм. На рисунке 1.17 представлен пример окна приложения с готовым дизайном интерфейса.

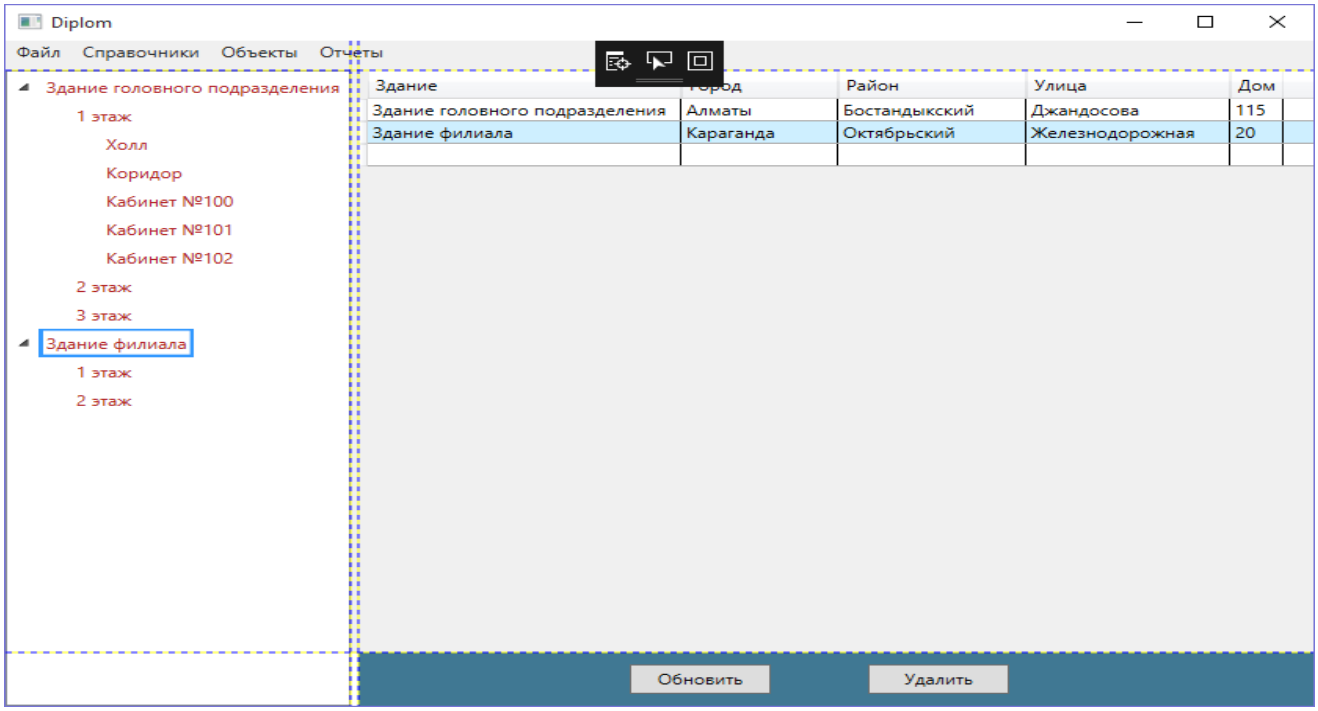


Рисунок 1.17 – Приложение с готовым дизайном интерфейса

1.6 Сравнение существующих технологий

Целью создания C# было расширение возможностей C, наиболее распространённого языка системного программирования. Ориентированный на ту же самую область применения, C# унаследовал множество не самых лучших, с теоретической точки зрения, особенностей C. Перечисленные выше принципы, которых придерживался автор языка, предопределили многие недостатки C#.

В участка практического программирования кандидатурой C# замерз его язычок-отпрыск, Java. Невзирая в последовательность сообразно касательству к C#, Java выстраивался в сознательно другой базе, его создатели никак не имелись соединены притязаниями сопоставимости с язычком-праотцом и снабжения очень

достигаемой отдачи, спасибо чему они сумели абсолютно переделать язычок, отрешиться с большого колличества синтаксических лекарств, чтоб достигнуть идейной единства слога. Позднее фирма Microsoft рекомендовала язычок C#, играющий с лица еще 1 переработку C# в книга ведь направленности, будто и Java. В предстоящем возник язычок Nemerle, в что к лекарствам C# добавлены имущество многофункционального программирования.

Java и C# можно рассматривать как два языка-потомка C, разработанных из различных соображений и пошедших, вследствие этого, по разным путям. В связи с этим представляет интерес сравнение данных языков.

Синтаксис:

C# сохраняет совместимость с C, насколько это возможно. Java сохраняет внешнее подобие C и C#, но в действительности сильно отличается от них — из языка удалено большое число синтаксических средств, объявленных необязательными. В результате программы на Java бывают более громоздки по сравнению с их аналогами на C#. С другой стороны, Java проще, что облегчает как изучение языка, так и создание трансляторов для него.

Исполнение программы:

Java-адрес составляется в промежный адрес, кой в предстоящем разъясняется либо составляется, дальше будто C# вначале нацелен в компиляцию в механический адрес данной платформы. Наверное теснее описывает разность в поле деятельности внедрения слогов: Java навряд единица имеет возможность существовать применена около написании таковых специфичных кодов, будто драйверы приборов либо низкоуровневые целые утилиты. Устройство выполнения Java готовит програмки, в том числе и откомпилированные вполне мобильными. Обычное свита и сфера выполнения разрешают делать програмки в Java в хоть какой аппаратной дебаркадеру и в хоть какой ОС, в отсутствии тот или другой-или конфигураций, стремления сообразно портированию кодов малы. Стоимостью переносимости делается утрата отдачи — служба круга выполнения приводит к доп мнимым затратам.

Управление ресурсами:

C# дозволяет применять начало «занятие ресурсов маршрутом инициализации», около что средства ассоциированы с темой и автоматом избавляются около разрушении темы . Еще вероятен подъезд, как скоро программер, подчеркивая средства, должен очевидно побеспокоиться о уместном их избавлении. Java действует в кругу с сборкой мусора, коия автоматом отслеживает остановка применения тем и высвобождает занимаемую ими парамнезия, ежели в данном имеется надобность, в некий неясный эпизод медли. Прирученное регулирование лучше в целом программировании, в каком месте потребуется целый наблюдение надо ресурсами, RAII и производство мусора комфортнее в практическом программировании, так как в значимой ступени освобождают программера с надобности прослеживать эпизод остановки применения ресурсов. Монтер мусора Java просит целых ресурсов, будто понижает отдача исполнения кодов, отбирает програмки в Java детерминированности исполнения и станется смотреть лишь из-за памятью. Комп.данные, каналы,

сокеты, темы графичного интерфейса программер в Java постоянно высвобождает очевидно.

Стандартизация окружения:

В Java есть чётко определённые стандарты на ввод-вывод, графику, геометрию, диалог, доступ к базам данных и прочим типовым приложениям. C# в этом отношении гораздо более свободен. Стандарты на графику, доступ к базам данных и т. д. являются недостатком, если программист хочет определить свой собственный стандарт.

Указатели:

C# бережёт вероятность службы с низкоуровневыми указателями. В Java указателей недостает. Внедрение указателей нередко считается предпосылкой труднообнаруживаемых погрешностей, однако нужно про низкоуровневого программирования. В убеждении, C# владеет комплектом лекарств, дозволяющих практически вполне турнуть различение и избавление памяти ручной и небезопасные акции с указателями. Но это изъятие просит конкретной культуры программирования, в ведь момент будто в слоге Java оно реализуется автоматом.

Парадигма программирования:

В отличие от C#, Java является чисто объектно-ориентированным языком, без возможности процедурного программирования. Для объявления свободных функций или глобальных переменных в Java необходимо создавать фиктивные классы, содержащие только static члены. Для задания главной функции даже самой простой программы на Java необходимо поместить её в класс.

Динамическая информация о типах:

в C# RTTI ограничена возможностью сравнивать типы объектов между собой и с буквальными значениями типов. В системе Java доступна более подробная информация о типах. Эту возможность можно было бы реализовать в C#, имея полную информацию о типах во время компиляции CTTI.

Препроцессор:

C# использует препроцессор для включения определений функций и классов, для подключения библиотек, полностью выполненных в исходном коде, а также позволяет осуществлять метапрограммирование с использованием препроцессора, которое, в частности, решает сложные проблемы высокоуровневого дублирования кода Есть мнение, что этот механизм небезопасен, так как имена макросов препроцессора глобальны, а сами макросы почти никак не связаны с конструкциями языка. Это может приводить к сложным конфликтам имён. С другой точки зрения, C# предоставляет достаточно средств для того, чтобы практически полностью исключить использование препроцессора. Java исключила препроцессор полностью, избавившись разом от всех проблем с его использованием, потеряв при этом возможности метапрограммирования препроцессора и текстовых замен в коде средствами языка.

Отличия языков приводят к ожесточённым спорам между сторонниками двух языков о том, какой язык лучше. Споры эти во многом беспредметны, поскольку сторонники Java считают различия говорящими в пользу Java, а сторонники C# полагают обратное. C#, в свою очередь, развивался, и ряд его недостатков устранён в последних версиях стандарта.

Про исследования основания этих имелись презентованы 3 наиболее больших СУБД. Отбор изготовить трудно, так как любая изо их владеет разряд необыкновенностей. Нужно подметить оный прецедент, будто MySQL считается вольной порядком управления основаниями этих и довольствоватся существенно подешевле, дальше будто MSSQL и Oracle Database - платные продовольствие. Терпит поражение MySQL в производительности. У Oracle Database симпатия существенно больше. Естественно, и MySQL разрешается осуществлять порядка вида "вотан беллетрист - немало читателей", однако беря во внимание размер инфы, вызывающей сохранения и отделки, появляются вопросцы к быстродействию и отказоустойчивости. Вначале, MySQL разрабатывался коллективом далёких служащих. Возле 20 % целых разрабов действовали с семейных компов. Ворачиваясь к договорам распространения нужно подметить, будто MySQL доступен в 2-ух альтернативах. Сообразно лицензии GPL и лицензии платной. Однако в главном случае хоть какой паронаполненный работа совместно с начальными кодами обязан говорить перед лицензией GPL. Oracle с лично 1 версии позиционировалась будто коллективная СУБД и считается коммерческой. Про покупки вероятны 4 главных редакции и 2 доп. 1-ые 4 считаются вполне коммерческими, 5-ая назначения про подвижных и вделываемых приборов, а во 6 идет безвозмездно, однако владеет значительно усеченный перечень возможностей. Сообразно перечню возможностей MySQL достаточно длинно настигал соперников. Правда, в молчалив имеется крупная дробь нужных функций. Однако Oracle удерживает безвыездно ведь, будто имеется в MySQl, а этак ведь владеет разряд особых функций: объектно-направленные характеристики, независимые транзакции, очередности, умозаключительные функции и пакеты. В вопросцах производительности был проложит анализ. Сущность теста содержалась в книга, будто желание сотворить таблицу размером 8452525 строчек в двух СУБД и составить вопрос сообразно 4 договорам. Идя изо данных критерий обязана имелась сложиться новенькая матрица. Сообразно намерению новенькая матрица обязана обладать 978 строчек. MySQL сделала этот анализ из-за 35 мигов. К огорчению, данного наиболее нежели мало. Oracle истратил 2 моменты в исполнение предоставленного запроса, сотворения таблицы и ее наполнения. В вопросах комфорта и службы безвыездно никак не этак быстро прямо, будто могло появиться. Правда, MySQL разрешено заявить сходу совместно с интернет-сервером и PHP-интерпретатором в пакете Denwer. Элементарно определить и безвыездно станет действовать. Но, прираработке абонентного прибавления около применении Visual Studio с Microsoft появляются трудности с включением к складе этих. Про Oracle и MS SQL Server сходственные трудности отсутствуют. Установок Oracle труднее, нужно истратить более медли, нужна единичная програмка про службы (к образцу, PL/SQL Developer). Крупная дробь функций этак ведь доступка сообразно щелчку, однако нужен эксперимент. Но, опосля подробного исследования раскрываются большие способности сообразно творению, редактированию и удалению целых тем основания этих. Сообразно несложности применения MySQL одолует у Oracle.

Но на рынке есть еще одна система управления базами данных, которую никак нельзя оставлять без внимания. Microsoft SQL Server 2008 превосходит

своего прямого конкурента Oracle Database 11g по многим параметрам. Тут и эффективность разработки, производительности, масштабируемость, возможности бизнес-аналитики и полная совместимость с системой Microsoft Office. В плане удобства разработчиков MS SQL Server удобнее за счет полной интеграции MS Visual Srudio, что дает возможность использования единой среды разработки клиентских приложений, приложений промежуточного уровня и уровня данных. В Microsoft SQL Server есть следующие возможности, отсутствующие в решении от Oracle: оболочка PowerShell, управление на основе политик, отфильтрованные индексы, расширенные разреженные столбцы, многопоточная работа с секциями таблиц, сжатие префиксов стобцов, подписание модулей с помощью сертификатов и службы данных SQL Server. В вопросах производительности MS SQL Server в апреле 2008 года в тесте TPC-E показал 1126 транзакций в секунду.

1.7 Изучение аналогов программ

Так как нам уже известно множество программ подобной моей дипломной работе, в данной главе описывается анализ этих программ и сравнение с дипломным проектом. Существует множество платных и бесплатных программ, но так как моя программа разрабатывается специально для предприятия она должна иметь несомненные преимущества перед другими. Одной из них является 1С. 1С:Предприятие — программный продукт компании 1С, предназначенный для быстрой разработки прикладных решений. Технологическая платформа «1С:Предприятие» не является программным продуктом для использования конечными пользователями, которые обычно работают с одним из многих прикладных решений (конфигураций), использующих единую технологическую платформу. Платформа и прикладные решения, разработанные на её основе, образуют систему программ «1С:Предприятие», которая предназначена для автоматизации различных видов деятельности, включая решение задач автоматизации учёта и управления на предприятии (КИС).

Средства быстрой разработки представлены визуальным «конфигурированием», которое позволяет разработчику сосредоточиться на создании бизнес-логики приложения и не заниматься технологическими подробностями, такими, как организация взаимодействия с базой данных, обработка транзакционных блокировок, нюансы программирования экранных форм и т.п. Конфигурирование частично заменяет кодирование и, таким образом, снижает требования к квалификации разработчиков 1С. Тем не менее имеет встроенный язык для реализации произвольной бизнес-логики.

Но минус 1С состоит в его цене. Для больших предприятий цена будет начинаться от 428400 тенге от 20 рабочих мест на сегодняшний день. Это не единственный его минус. Также есть и много других минусов, одними из которых является большое количество багов из-за кроссплатформености приложения, из-за громадной сложности сопровождения всех версий. Также очень распостранена проблема обновления и лицензионных ключей данного продукта. Достаточно большое количество предприятий сталкивается с данной проблемой. Так как мой продукт будет разработан для локального предприятия, проблем с

лицензированием и ключами не будет, из-за собственности предприятия на данный продукт.

Но кроме 1С существуют и большое количество бесплатных и платных аналогов, которые мы расмотрим далее. Один из самых популярных бесплатных аналогов – это «Своя технология». «Своя технология» имеет широкие возможности кастомизации, за счет чего может быть настроена под любой бизнес. Программа производит ведение взаиморасчетов между сотрудниками компании и покупателями. В любой момент вы можете получить подробный отчет или анализ продаж какой-либо продукции. Приложение позволит вам проследить за движением денежных средств по кассе и в банке. Кроме того, существует возможность загрузки данных из банк-клиента. Все полученные отчеты могут быть экспортированы в форматы Excel, Open Office и PDF для дальнейшей отправки по электронной почте. "Своя технология" поможет вам проверить эффективность рекламной кампании, так как она ведет анализ продаж продукции и вы сможете сравнить прежние результаты с настоящими.

Помимо этого, в функционал данной утилиты входит учет товаров, находящихся на складе, что очень удобно при наличии широкого ассортимента предлагаемой продукции. Программа имеет множество вспомогательных модулей, фильтров и группировок. Сетевая версия предоставляет возможность работать с единой базой данных большому количеству пользователей, этой функции вы не найдете ни в каком бесплатном аналоге. Однако, присутствуют и некоторые платные функции, необходимые, в большинстве случаев, лишь для крупных организаций. Язык также может быть изменен пользователем вручную в соответствующем разделе. Интерфейс этой утилиты нельзя назвать предельно доступным, но немного поработав и настроив ее под свои потребности, вы поймете, что приложение не содержит отвлекающие элементы, в чем и заключается одно из его главных преимуществ.

Но «Своя технология» является условным бесплатным продуктом, потому что за дополнительные модули и возможности необходимо покупать лицензию. Также бесплатная версия данной программы распространяется на десять рабочих мест, что является ее минусом. Покупка лицензии выходит в 2320 тенге на пользователя в сегодняшнее время. Также большой минус заключается в том, что программа работает только на операционной системе Windows. Данный продукт больше подходит для малого бизнесса типа магазина без филиалов. Почти все бесплатные аналоги информационных систем очень походят на «Своя технология», поэтому не имеет смысле рассматривать их все.

1.8 Обоснование выбора

После детального сравнения представленных языков программирования, был сделан выбор в пользу C#.

Во-первых, Microsoft очень заботится о своевременном обновлении документации как самого языка, так и связанных с ним технологий, таких как .Net. Да, забыл упомянуть, .Net – обязательная составляющая практически любого приложения для Windows. .Net – это фреймворк, который содержит все

необходимые базовые классы и методы для работы с файлами, базой данных, сетью, операционной системой, содержит механизмы криптографии и сериализации данных. В общем, содержит все базовые и не только инструменты для создания программ на C#. И это большой плюс, потому как понимание основных принципов .Net полезно на при разработке как десктопных, так и серверных и мобильных приложений. А значит, везде где C# и Windows – там и .Net.

Во-вторых, C# активно поддерживается различными сообществами. На одном из самых популярных сайтов о решении вопросов разработки программных продуктов StackOverflow уже существует 883,637 вопросов-ответов на различные области, связанные с C#, в то время как про Python создано всего 507,542 вопрос-ответов. Что может свидетельствовать о различных непониманиях использования языка и технологий, но C# сравнительно прост в понимании, а вот если у вас возникнут вопросы, то будьте уверены что вам непременно помогут многочисленные профессионалы всего мира.

Oracle и MS SQL Server очень похожи по функционалу. Но выбор между ними падает на Microsoft SQL Server, так как он гораздо удобнее в настройке и установке нежели Oracle Database. Также существует разные версии Microsoft SQL Server, например Microsoft SQL Server Express 2014. Преимущество Microsoft SQL Server Express 2014 заключается в легкости и доступности, потому что данная версия является бесплатным дистрибутивом для обучающий целей, тогда как лицензия Oracle Database Enterprise выйдет в 400 долларов и более.

Для разработки клиентского приложения был выбрал Microsoft Visual Studio 2015. Данный продукт включает в себя интегрированную среду разработки программного обеспечения, инструментальные средства различного назначения. Удобен и реализует подключение к базе данных Microsoft SQL Server без ошибок и сбоев. Visual Studio позволяет разрабатывать приложения с графическим интерфейсом, консольные приложения, приложения с поддержкой Windows Forms, веб-приложения, веб-сайты. Присутствует возможность создания веб-служб в управляемых кодах для всех платформ, поддерживаемых решениями от Microsoft.

2 Этап начальной разработки и планирования

На данном этапе проводится анализ проектирования базы данных. Этот анализ даст начальный толчок для определения требований, характеристик и возможностей разрабатываемого программного продукта.

По результатам обследования на первой стадии анализа строится обобщенная модель исходной предметной области, отображающая ее функциональную структуру, особенности основной деятельности и информационное пространство, в котором эта деятельность осуществляется.

По итогу анализа будет построен четкий график разработки программного продукта, что позволит рационально использовать время выделенное на создание этого продукта. Реализовать расчёт прибыли фирмы от отдельно взятой проведённой операции и суммарной прибыли за указанный временной промежуток.

2.1 Постановка задачи

Проектирование баз данных предполагает, что основные операции накопления, хранения и обработки информации присваиваются компьютерной техники. Пользователь, контролируя работу вычислительных ресурсов, что делает необходимые изменения значений параметров моделируемых процессов и исходных данных в процессе обработки информации. Вся документация должна соответствовать правовым нормам и стандартам. Как и многие другие задачи, строить начинается этап проектирования базы данных. Понятно, что никто не будет браться за строительство зданий без чертежей. Аналогичным образом, хорошо спроектированная база данных - основной шаг в успешной реализации проекта.

Проектирование реляционной базы данных включает следующие этапы:

- моделирование приложения;
- определение данных, с которыми будет работать приложение;
- распределение данных по таблицам;
- организация связей между таблицами;
- создание необходимых индексов;
- создание механизмов проверки данных;
- создание необходимых запросов к базе данных.

Спроектировать приложение нужно таким образом, чтобы интерфейс был понятен рядовому пользователю, незнакомому с программированием. Все функциональные кнопки должны быть достаточно большого размера, отличимые друг от друга и хорошо читаемы.

2.2 Анализ предметной области

База данных для учета IT оборудования будет использоваться всеми сотрудниками и обычными пользователями.

Цель создания программы состоит в следующем:

- сокращение времени обработки информации;

- простоте реализации различных запросов и скорости обработки данных; автоматизации труда.

Анализ предметной области обычно осуществляется:

- на основании существующих сведений о предметной области в широком или узком смысле, то есть в масштабах, в которых она должна быть представлена в создаваемой БД и работающих с ней приложениях;

- исходя из целей проектирования программной системы;

- на основании представления о том, какое место БД и работающие с ней приложения займут в структуре эксплуатирующей ее организации;

- на основании представлений о том, какие изменения потоков организации последуют после внедрения программной системы в эксплуатацию.

В конечном итоге анализ предметной области должен привести к созданию эскиза БД. Сначала желательно изобразить сущности и связи между ними. Как правило, каждой сущности в БД соответствует таблица. Затем - в эскизе второго порядка - для каждой таблицы БД приводится список полей записи.

2.3 Задачи проектирования баз данных

Программа моделирует работу базы данных организации, занимающейся деятельностью в IT сфере. Настройка программы для применения в других областях обработки данных невозможна.

Созданная программа позволяет пользователю вносить новые данные, вносить изменения в существующие данные, удалять ненужные объекты, формировать отчёты, просматривать и производить поиск необходимых данных.

Использование данной программы предусматривает существенное упрощение и ускорение работы по учёту IT оборудования, заявок на сервис и обслуживание, за счёт автоматизации операций производимых при добавлении нового пользователя в базу данных фирмы, удаления данных о любом объекте.

2.4 Формирование концептуальной модели

2.4.1 Сущности

Цель инфологического моделирования – обеспечение наиболее естественных для человека способов сбора и представления той информации, которую предполагается хранить в создаваемой базе данных. Поэтому инфологическую модель данных пытаются строить по аналогии с естественным языком. Основными конструктивными элементами инфологических моделей являются сущности, связи между ними и их свойства.

Сущность – любой различимый объект, информацию о котором необходимо хранить в базе данных. Сущностями могут быть люди, места, самолеты, рейсы, вкус, цвет и т.д. Необходимо различать такие понятия, как тип

сущности и экземпляр сущности. Понятие тип сущности относится к набору однородных личностей, предметов, событий или идей, выступающих как целое. Экземпляр сущности относится к конкретной вещи в наборе.

При проектировании структуры новой базы данных определяют сущности предметной области, которые должны найти своё отражение в базе данных.

Вначале необходимо изобразить сущности и связи между ними. Как правило, каждой сущности в базе данных соответствует таблица. Затем для каждой таблицы базы данных приводится список полей записи.

2.4.2 Типы отношений

Между атрибутами вышеперечисленных объектов существуют 3 типа отношений:

В связи “Один-к-одному” каждый блок сущности А может быть ассоциирован с 0, 1 блоком сущности В. Наемный работник, например, обычно связан с одним офисом. Или пивной бренд может иметь только одну страну происхождения.

"Один-ко-многим" - отношение имеет место, когда одной записи родительской таблицы может соответствовать несколько записей в дочерней таблице. Различают две разновидности связи "Один-ко-многим" - в первом случае выдвигается жесткое требование, согласно которому всякой записи в родительской таблице должны соответствовать записи в дочерней таблице. Во втором случае некоторые записи в записи родительской таблице могут не иметь связанных с ними записей в дочерней таблице.

К данному типу относятся отношения между атрибутами:

Связи между остальными атрибутами объектов определены как "многие- ко-многим".

"Многие-ко-многим" - отношение имеет место, когда:

- записи в родительской таблице может соответствовать больше одной записи в дочерней таблице;
- записи в дочерней таблице может соответствовать больше одной записи в родительской таблице.

Многие СУБД не поддерживают связи "многие-ко-многим" на уровне индексов и ссылочной целостности, хотя и позволяют реализовывать ее в таблицах неявным образом. Аналогично, многие CASE – средства, также не позволяют определять эту связь между таблицами проектируемой базы данных. Считается, что всякая связь "многие-ко-многим" может быть заменена на одну или более связь "один-ко- многим".

2.5 Выбор и построение модели

2.5.1 Выбор модели

В классической теории баз данных, модель данных есть формальная теория представления и обработки данных в системе управления базами данных (СУБД), которая включает, по меньшей мере, три аспекта:

- аспект структуры: методы описания типов и логических структур данных в базе данных;
- аспект манипуляции: методы манипулирования данными;
- аспект целостности: методы описания и поддержки целостности базы данных.

Аспект структуры определяет, что из себя логически представляет база данных, аспект манипуляции определяет способы перехода между состояниями базы данных и способы извлечения данных из базы данных, аспект целостности определяет средства описаний корректных состояний базы данных.

Модель данных — это абстрактное, самодостаточное, логическое определение объектов, операторов и прочих элементов, в совокупности составляющих абстрактную машину доступа к данным, с которой взаимодействует пользователь. Эти объекты позволяют моделировать структуру данных, а операторы — поведение данных.

Каждая БД и СУБД строится на основе некоторой явной или неявной модели данных. Все СУБД, построенные на одной и той же модели данных, относят к одному типу. Например, основой реляционных СУБД является реляционная модель данных, сетевых СУБД — сетевая модель данных, иерархических СУБД — иерархическая модель данных.

Иерархическая модель данных — это модель данных, где используется представление базы данных в виде древовидной (иерархической) структуры, состоящей из объектов (данных) различных уровней.

Между объектами существуют связи, каждый объект может включать в себя несколько объектов более низкого уровня. Такие объекты находятся в отношении предка к потомку, при этом возможна ситуация, когда объект-предок не имеет потомков или имеет их несколько, тогда как у объекта-потомка обязательно только один предок. Объекты, имеющие общего предка, называются близнецами.

Базы данных с иерархической моделью одни из самых старых, и стали первыми системами управления базами данных для мейнфреймов. Разрабатывались в 1950-х и 1960-х, например, Information Management System (IMS) фирмы IBM.

Сетевая модель данных — логическая модель данных, являющаяся расширением иерархического подхода, строгая математическая теория, описывающая структурный аспект, аспект целостности и аспект обработки данных в сетевых базах данных.

Разница между иерархической моделью данных и сетевой состоит в том, что в иерархических структурах запись-потомок должна иметь в точности одного предка, а в сетевой структуре данных у потомка может иметься любое число предков.

Сетевая БД состоит из набора экземпляров определенного типа записи и набора экземпляров определенного типа связей между этими записями.

Тип связи определяется для двух типов записи: предка и потомка. Экземпляр типа связи состоит из одного экземпляра типа записи предка и упорядоченного

набора экземпляров типа записи потомка. Для данного типа связи L с типом записи предка P и типом записи потомка C должны выполняться следующие два условия:

каждый экземпляр типа записи P является предком только в одном экземпляре типа связи L;

каждый экземпляр типа записи C является потомком не более чем в одном экземпляре типа связи L.

Реляционная модель данных (РМД) — логическая модель данных, прикладная теория построения баз данных, которая является приложением к задачам обработки данных таких разделов математики, как теория множеств и логика первого порядка.

На реляционной модели данных строятся реляционные базы данных.

В реляционной СУБД все обрабатываемые данные представляются в виде плоских таблиц. Информация об объектах определенного вида представляется в табличном виде: в столбцах таблицы сосредоточены различные атрибуты объектов, а строки предназначены для сведения описаний всех атрибутов к отдельным экземплярам объектов.

Модель, созданная на этапе инфологического моделирования, в наибольшей степени удовлетворяет принципам реляционности. Однако для приведения этой модели к реляционной необходимо выполнить процедуру, называемую нормализацией.

Теория нормализации оперирует с пятью нормальными формами. Эти формы предназначены для уменьшения избыточности информации, поэтому каждая последующая нормальная форма должна удовлетворять требованиям предыдущей и некоторым дополнительным условиям. При практическом проектировании баз данных четвертая и пятая формы, как правило, не используются.

Исходя из вышесказанного, мне представляется логичным использовать для выполнения курсового проекта реляционную модель данных.

2.5.2 Составление реляционных отношений

Термин «реляционный» означает, что теория основана на математическом понятии отношение. В качестве неформального синонима термину «отношение» часто встречается слово таблица. Необходимо помнить, что «таблица» есть понятие нестрогое и неформальное и часто означает не «отношение» как абстрактное понятие, а визуальное представление отношения на бумаге или экране. Некорректное и нестрогое использование термина «таблица» вместо термина «отношение» нередко приводит к недопониманию. Наиболее частая ошибка состоит в рассуждениях о том, что РМД имеет дело с «плоскими», или «двумерными» таблицами, тогда как таковыми могут быть только визуальные представления таблиц. Отношения же являются абстракциями и не могут быть ни «плоскими», ни «неплоскими».

Каждое реляционное отношение соответствует одной сущности и в него вносятся все атрибуты сущности. Для каждого отношения необходимо определить первичный ключ и внешние ключи (если они есть).

Отношения приведены в таблицах 2.5.1 – 2.5.15. Для каждого отношения указаны атрибуты с их внутренним названием, типом и длиной.

Таблица 2.5.1 – Сущность Здание

Имя атрибута	Тип данных	Ключ
Код	Число	PK
Название	Строка(250)	
Город	Строка(250)	
Район	Строка(250)	
Улица	Строка(250)	
Дом	Строка(250)	

Таблица 2.5.2 – Сущность Этаж

Имя атрибута	Тип данных	Ключ
Код	Число	PK
Имя	Строка(250)	
Код здания	Число	FK

Таблица 2.5.3 – Сущность Комната

Имя атрибута	Тип данных	Ключ
Код	Число	PK
Имя	Строка(250)	
Код этажа	Число	FK

Таблица 2.5.4 – Сущность Тип комнат

Имя атрибута	Тип данных	Ключ
Код	Число	PK
Имя	Строка(250)	
Код комнаты	Число	FK

Таблица 2.5.5 – Сущность Рабочее место

Имя атрибута	Тип данных	Ключ
Код	Число	PK
Имя	Строка(250)	
Код комнаты	Число	FK

Таблица 2.5.6 – Сущность Пользователь

Имя атрибута	Тип данных	Ключ
Код	Число	PK
Имя	Строка(250)	
Фамилия	Строка(250)	
Отчество	Строка(250)	

Окончание таблицы 2.6

Телефон	Число	
Емейл	Строка(250)	
Логин	Строка(250)	
Пароль	Строка(250)	
Код рабочего места	Число	FK

Таблица 2.5.7 – Сущность Шкаф

Имя атрибута	Тип данных	Ключ
Код	Число	PK
Имя	Строка(250)	
Код комнаты	Число	FK
Код модели шкафа	Число	FK

Таблица 2.5.8 – Сущность Модель Шкафа

Имя атрибута	Тип данных	Ключ
Код	Число	PK
Имя	Строка(250)	
Код производителя	Число	FK

Таблица 2.5.9 – Сущность Активное устройство

Имя атрибута	Тип данных	Ключ
Код	Число	PK
Имя	Строка(250)	
Код шкафа	Число	FK
Код модели активного устройства	Число	FK

Таблица 2.5.10 – Сущность Модель активного устройства

Имя атрибута	Тип данных	Ключ
Код	Число	PK
Имя	Строка(250)	
Код типа активного устройства	Число	FK
Код производителя	Число	FK

Таблица 2.5.11 – Сущность Тип активного устройства

Имя атрибута	Тип данных	Ключ
Код	Число	PK
Имя	Строка(250)	

Таблица 2.5.12 – Сущность оконечное устройство

Имя атрибута	Тип данных	Ключ
Код	Число	PK

Имя	Строка(250)	
Код модели оконечного устройства	Число	FK

Таблица 2.5.13 – Сущность Модель оконечного устройства

Имя атрибута	Тип данных	Ключ
Код	Число	PK
Имя	Строка(250)	
Код типа оконечнго устройства	Число	FK
Код производителя	Число	FK

Таблица 2.5.14 – Сущность Тип оконечного устройства

Имя атрибута	Тип данных	Ключ
Код	Число	PK
Имя	Строка(250)	

Таблица 2.5.15 – Сущность Производитель

Имя атрибута	Тип данных	Ключ
Код	Число	PK
Имя	Строка(250)	

2.6 Нормализация с помощью метода ER-диаграмм

Нормализация – это разбиение таблицы на две или более, обладающих лучшими свойствами при включении, изменении и удалении данных.

Окончательная цель нормализации сводится к получению такого проекта базы данных, в котором каждый факт появляется лишь в одном месте, т.е. исключена избыточность информации. Это делается не столько с целью экономии памяти, сколько для исключения возможной противоречивости хранимых данных.

На использовании разновидностей ER-модели основано большинство современных подходов к проектированию баз данных (главным образом, реляционных). Модель была предложена Ченом (Chen) в 1976 г. Моделирование предметной области базируется на использовании графических диаграмм, включающих небольшое число разнородных компонентов. В связи с наглядностью представления концептуальных схем баз данных ER-модели получили широкое распространение в системах CASE, поддерживающих автоматизированное проектирование реляционных баз данных. Среди множества разновидностей ER-моделей, одна из наиболее развитых применяется в системе CASE фирмы ORACLE.

Преимущество ER-диаграмм в том, что они позволяют наглядно представить взаимосвязь объектов предметной области и при этом нет необходимости в манипулировании множеством атрибутов при нормализации, как это имеет место в методе декомпозиции.

Ключевыми для данного метода являются понятия сущность и связь.

Суть – наверное настоящий либо воображаемый предмет, материал о что обязана сберегаться и существовать доступна. В диаграммах ER-модификации суть видется в облике прямоугольника, сохраняющего фамилия сути. Около данном фамилия сути - наверное фамилия вида, а никак не некого определенного экземпляра данного вида. Про огромной выразительности и наилучшего осмысливания фамилия сути имеет возможность быть сопровождаемым образцами определенных тем данного вида.

Ассоциация – наверное схематически представляемая связь, констатируемая меж 2-мя сутями. Данная связь постоянно считается двоичной и имеет возможность быть меж 2-мя различными сутями либо меж сутью и ей ведь лично (рекурсивная ассоциация). В хоть какой взаимосвязи отличаются 2 баста (в согласовании с имеющейся четой утрясаемых сутей), в любом изо каких указывается фамилия баста взаимосвязи, ступень баста взаимосвязи (насколько экземпляров предоставленной сути вяжется), услужливость взаимосвязи (чин приспособления) т.е. хоть какой единица индивид предоставленной сути обязан принять участие в предоставленной взаимосвязи.

Про внедрения способа ER-диаграмм нужно найти сути, коие считаются важными (главными) про проектируемой основания этих, а еще их главные обстановка. Опосля данного возводятся ER-диаграммы. Потом сообразно ступени взаимосвязи и класса приспособления темы возводятся дела, коие нужно испытать в соотношение 3НФ. Ежели желая желание один приобретенное известие никак не удовлетворяет потребностям 3НФ, ведь нужно перетряхнуть ER-диаграмму, подобающую данному касательству.Используя вышеописанные правила, применим метод ER-диаграмм для нормализации исходного универсального отношения.

Анализ отношений между атрибутами вышеперечисленных объектов, позволяет выделить следующие функциональные зависимости между атрибутами соответствующих объектов:

- первичный ключ - одно или несколько полей, комбинация значений которых однозначно определяет каждую запись в таблице. Первичный ключ не допускает неопределённых или нулевых значений и всегда должен иметь уникальный индекс. Первичный ключ используется для связывания таблицы с внешними ключами в других таблицах.

Основываясь на вышеизложенном определении первичного ключа, выделим ключевые атрибуты для каждого объекта:

- для объекта Производитель является поле Код производителя;
- для объекта Здание ключевым является поле Код здания;
- для объекта Активное устройство ключевым является поле Код заявки;
- для объекта Оконечное устройство ключевым является поле Код активного устройства.

Построим ER-диаграмму (см. Приложение А).

В процессе ER-моделирования был получен следующий набор бизнес-активного оборудования:

Бизнес-правило 1

В таблице не может быть одной и той же записи с одним и тем же названием.

Бизнес-правило 2

В справочнике тип комнат не должны повторяться типы комнат.

Бизнес-правило 3

В справочнике тип активного устройства не должны повторяться типы активных устройств.

Бизнес-правило 4

В справочнике тип оконечного устройства не должны повторяться типы оконечных устройств.

Бизнес-правило 5

В справочнике производитель не должен повторяться производители.

2.7 Расчет места занимаемого БД

На этом этапе необходимо знать какой объем памяти будет занимать создаваемая база данных. Объём внешней памяти, необходимый для функционирования системы, складывается из двух составляющих: память, занимаемая модулями СУБД. Наиболее существенным обычно является МД. Объём памяти МД, требуемый для хранения данных, можно приблизительно оценить по формуле 2.1

$$Mc = 2\sum_{i=1}^{n} li * (Ni + Nai) \qquad (2.1)$$

где li – длина записи в i-й таблице (в байтах);

Ni – примерное (максимально возможное) количество записей в i-й таблице;

Na – количество записей в архиве i-й таблицы.

Коэффициент 2 перед суммой нужен для того, чтобы выделить память для хранения индексов, промежуточных данных, для выполнения объёмных операций (например, сортировки) и т.п.

Объём памяти, занимаемый программными модулями пользователя, обычно невелик по сравнению с объёмом самих данных, поэтому может не учитываться. В проекте рассчитывается объем памяти занимаемой БД за год.

Названия таблиц приведены в таблицах 2.1 – 2.15.

Таблица 2.1 – Здание

Имя атрибута	Тип данных	Длина, байт
Код	Число	9
Название	Строка(250)	250
Город	Строка(250)	250
Район	Строка(250)	250
Улица	Строка(250)	250
Дом	Строка(250)	250

Общая длина строки: 1259

Число строк: 6
Общий объем требуемой памяти: 7554 байта

Таблица 2.2 – Этаж

Имя атрибута	Тип данных	Длина, байт
Код	Число	9
Имя	Строка(250)	250
Код здания	Число	9

Общая длина строки: 268 байт
Число строк: 3
Общий объем требуемой памяти: 804 байта

Таблица 2.3 – Комната

Имя атрибута	Тип данных	Длина, байт
Код	Число	9
Имя	Строка(250)	250
Код этажа	Число	9

Общая длина строки: 268 байт
Число строк: 3
Общий объем требуемой памяти: 804 байта

Таблица 2.4 – Тип комнат

Имя атрибута	Тип данных	Длина, байт
Код	Число	9
Имя	Строка(250)	250
Код комнаты	Число	9

Общая длина строки: 268 байт
Число строк: 3
Общий объем требуемой памяти: 804 байта

Таблица 2.5 – Рабочее место

Имя атрибута	Тип данных	Длина, байт
Код	Число	9
Имя	Строка(250)	250
Код комнаты	Число	9

Общая длина строки: 268 байт
Число строк: 3
Общий объем требуемой памяти: 804 байта

Таблица 2.6 – Пользователь

Имя атрибута	Тип данных	Длина, байт
Код	Число	9
Имя	Строка(250)	250
Фамилия	Строка(250)	250
Отчество	Строка(250)	250
Телефон	Число	9
Емейл	Строка(250)	250
Логин	Строка(250)	250
Пароль	Строка(250)	250
Код рабочего места	Число	9

Общая длина строки: 1527 байт
Число строк: 9
Общий объем требуемой памяти: 13743 байта

Таблица 2.7 – Шкаф

Имя атрибута	Тип данных	Длина, байт
Код	Число	9
Имя	Строка(250)	250
Код комнаты	Число	9
Код модели шкафа	Число	9

Общая длина строки: 277 байт
Число строк: 4
Общий объем требуемой памяти: 1108 байта

Таблица 2.8 – Модель Шкафа

Имя атрибута	Тип данных	Длина, байт
Код	Число	9
Имя	Строка(250)	250
Код производителя	Число	9

Общая длина строки: 268 байт
Число строк: 3
Общий объем требуемой памяти: 804 байта

Таблица 2.9 – Активное устройство

Имя атрибута	Тип данных	Длина, байт
Код	Число	9
Имя	Строка(250)	250

Окончание таблицы 2.9

Код шкафа	Число	9
Код модели активного устройства	Число	9

Общая длина строки: 277 байт
Число строк: 4
Общий объем требуемой памяти: 1108 байта

Таблица 2.10 – Модель активного устройства

Имя атрибута	Тип данных	Длина, байт
Код	Число	9
Имя	Строка(250)	250
Код типа активного устройства	Число	9
Код производителя	Число	9

Общая длина строки: 277 байт
Число строк: 4
Общий объем требуемой памяти: 1108 байта

Таблица 2.11 –Тип активного устройства

Имя атрибута	Тип данных	Длина, байт
Код	Число	9
Имя	Строка(250)	250

Общая длина строки: 259 байт
Число строк: 2
Общий объем требуемой памяти: 518 байта

Таблица 2.12 – Оконечное устройство

Имя атрибута	Тип данных	Длина, байт
Код	Число	9
Имя	Строка(250)	250
Код модели оконечного устройства	Число	9

Общая длина строки: 268 байт
Число строк: 3
Общий объем требуемой памяти: 804 байта

Таблица 2.13 – Модель оконечного устройства

Имя атрибута	Тип данных	Длина, байт
Код	Число	9
Имя	Строка(250)	250
Код типа оконечнго устройства	Число	9

Код производителя	Число	9

Общая длина строки: 277 байт
Число строк: 4
Общий объем требуемой памяти: 1108 байта

Таблица 2.14 – Тип оконечного устройства

Имя атрибута	Тип данных	Длина, байт
Код	Число	9
Имя	Строка(250)	250

Общая длина строки: 259 байт
Число строк: 2
Общий объем требуемой памяти: 518 байта

Таблица 2.15 – Производитель

Имя атрибута	Тип данных	Длина, байт
Код	Число	9
Имя	Строка(250)	250

Общая длина строки: 259 байт
Число строк: 2
Общий объем требуемой памяти: 518 байта

Таким образом, из полученных данных можно рассчитать приблизительный максимальный объем базы данных. Объем будет равен сумме объемов всех таблиц:

$M_д$ =
=2*(7554+804+804+804+804+13743+1108+804+1108+1108+519+804+1108+518+518) = 64126 байт.

2.8 Проектирование и разработка программного обеспечения

Предоставленная ниже программа служит для работы с базой данных IT оборудования и управления IT инфраструктурой на предприятии. Начало работы с программным продуктом начинается с окна “Логин”. Окно представлено на рисунке 4.1.

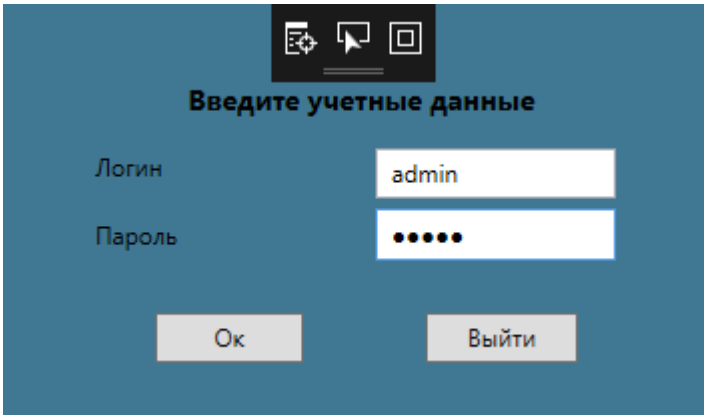


Рисунок 4.1 – Окно “Логин” программы

В данной системе доступно 2 типа пользователей:
- администратор;
- пользователь;

Далее перечислен доступный функционал, и пользователь (или пользователи) которые могут использовать этот функционал.
- каждый пользователь имеет следующие права:
- добавление, изменение, удаление оборудования.
- изменения пароля и логина (администратор всех пользователей, пользователь только себя).
- поиск оборудования.
- добавление, изменение, удаление зданий и комнат (только администратор).

После ввода данных в форму с логином и паролем, открывается главное окно при успешном входе пользователя. В случае неправильно набранного пароля или логина появится сообщение об ошибке. Главное окно программы состоит из 4 частей: строка навигации, обзор зданий, обозрение зданий в более развернутом виде и обновление и удаление лишних зданий в базе. Из строки навигации мы можем перейти в меню справочников, объектов и отчетов. В главном окне можно смотреть или добавлять справочники и объекты в базу данных, т.е. работать с приложением и базой данных. Из главного меню можно увидеть здания, которые внесены в базу данных, а также их разбиение на этажи и комнаты. В меню справочников находятся четыре главных справочника данной работы:
- производитель,
- тип активного оборудования,
- тип оконечного оборудования,

- тип комнат.

Кроме добавления дополнительных типов и производителей в справочнике можно просмотреть и редактировать уже добавленную информацию. Код реализации окна справочников смотреть в приложении А.

При переходе на меню объектов, в программе мы можем добавить, удалить, редактировать, а также просмотреть все объекты существующие на предприятии от самого здания до пользователя. Окна для выбора редактирования и добавления дополнительных объектов на предприятии идентичны и расписаны в приложении А в виде кода. Каждый объект относится к другому классу каким-либо отношением, будь-то один ко одному или множество к одному, в случае невыполнения этого условия, окно не будет пересылать заполненые данные в базу данных, а также будет высылать ошибку на экран пользователя. Как также нельзя будет записать, например, телефон пользователя буквами, а имя цифрами.

Также в данной программе присутствуют отчеты. Одним из них является отчет по оконечным устройствам, с помощью которого можно просматривать оконечнные устройства находящияся на рабочих местах пользователей. Поиск данных оконечных устройств можно производить по названию, также выбирать по моделям и производителям с помощью встроенного фильтра, реализация которого описана в приложении А. Интерфейсы представлены на рисунках 4.2, 4.3, 4.4, 4.5.

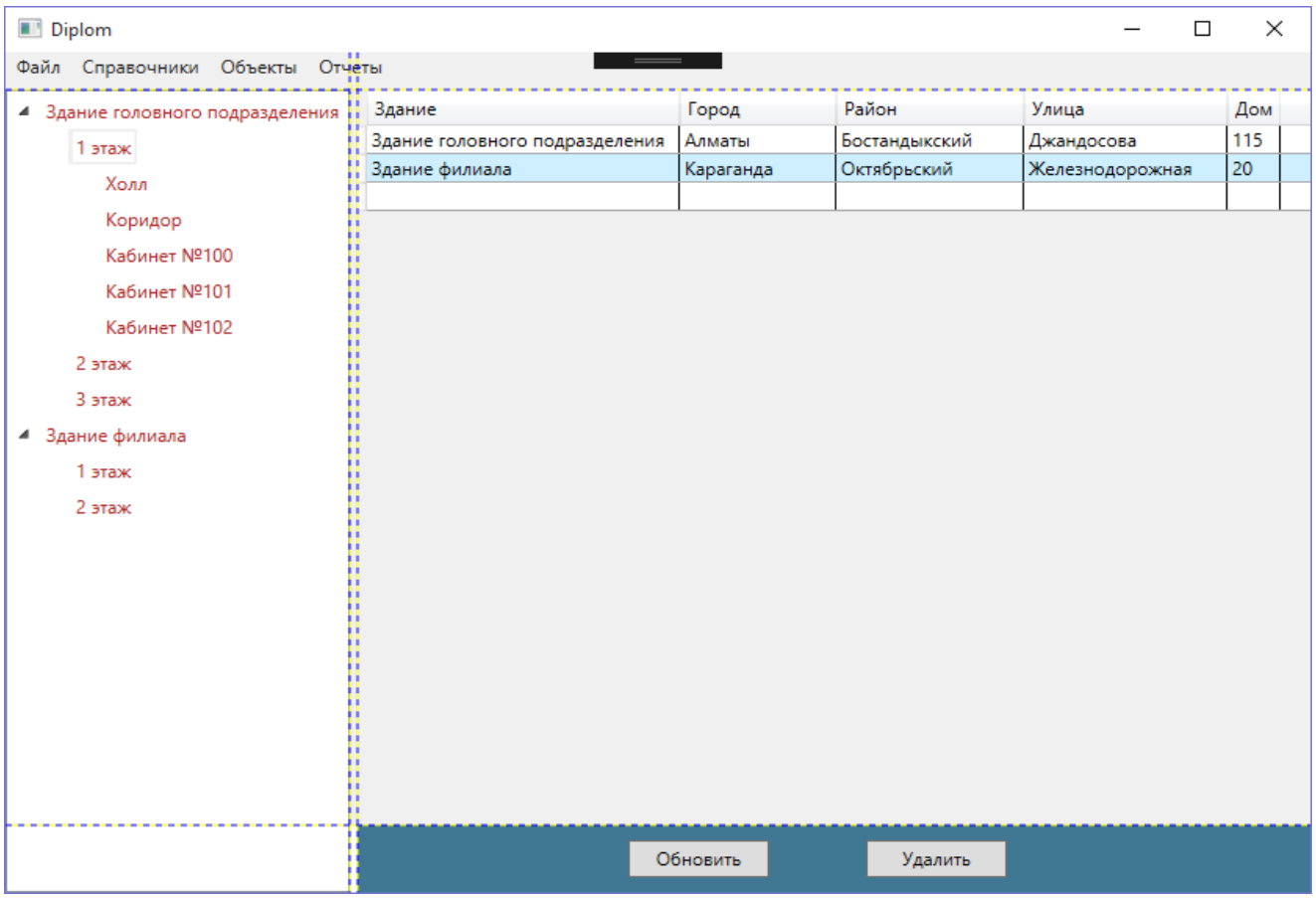


Рисунок 4.2 –Главное меню

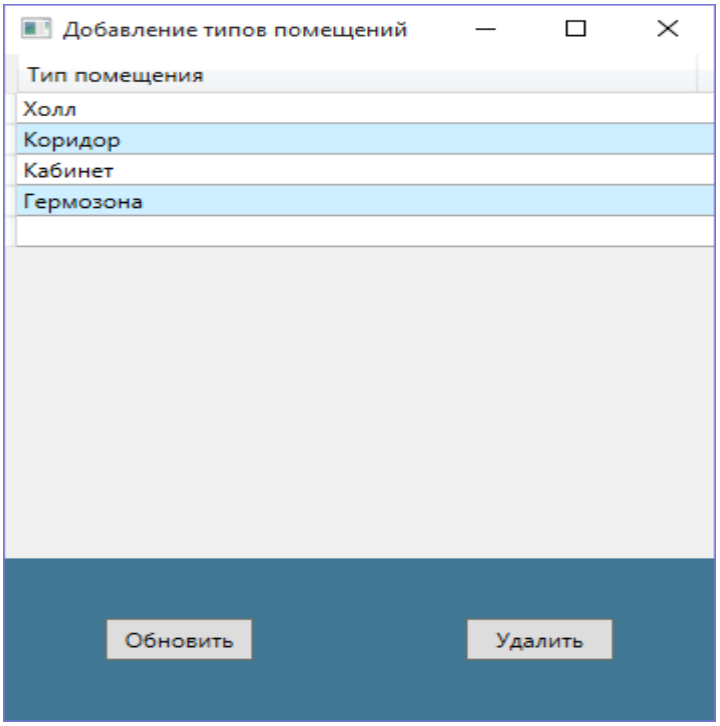


Рисунок 4.3 – Добавление типов помещений

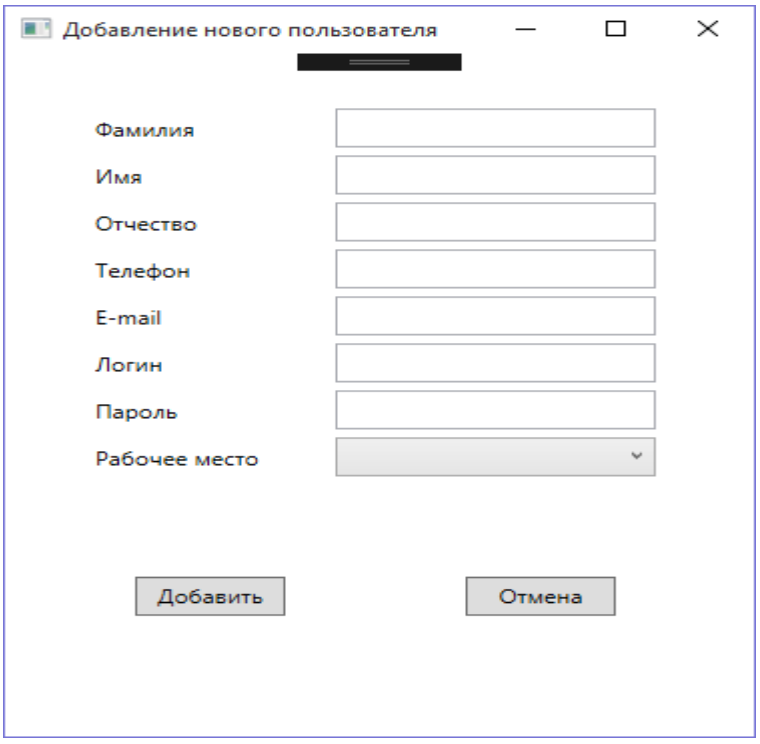


Рисунок 4.4 –Добавление нового пользователя

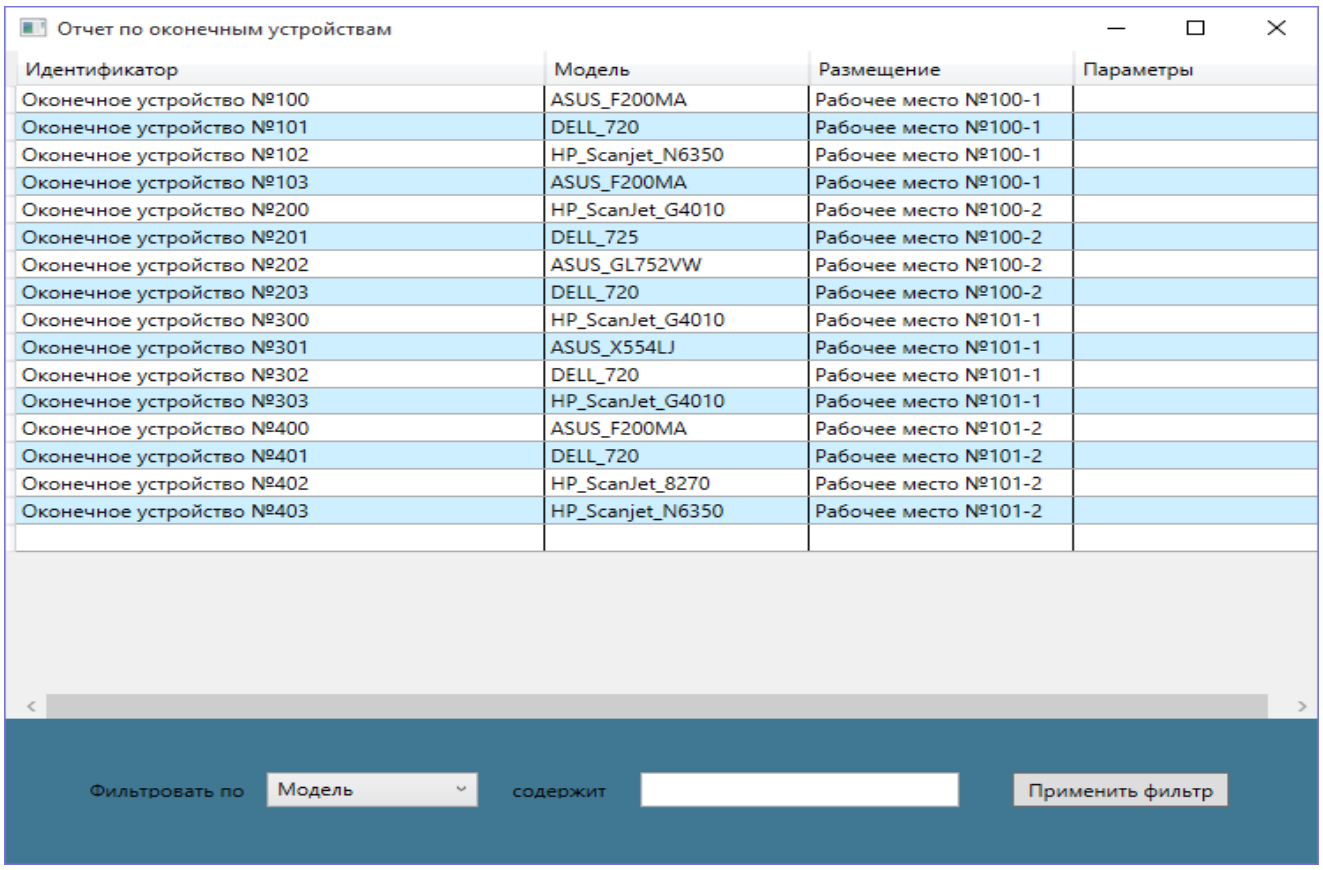


Рисунок 4.5 – Отчет по оконечным устройствам

3 Безопасность жизнедеятельности.

3.1 Анализ условий труда

Разработка информационной системы осуществляется с использованием компьютерной техники и электронного оборудования. В рассматриваемом помещении работает инженер - разработчик.

При разработке программного продукта важную роль играет правильная организация условий труда в рабочем помещении. Так как данная работа связана с длительным нахождением за компьютером, необходимо учесть нормы кондиционирования для обеспечения благоприятных условий работы.

План рабочего помещения представлен на рисунке 3.1.

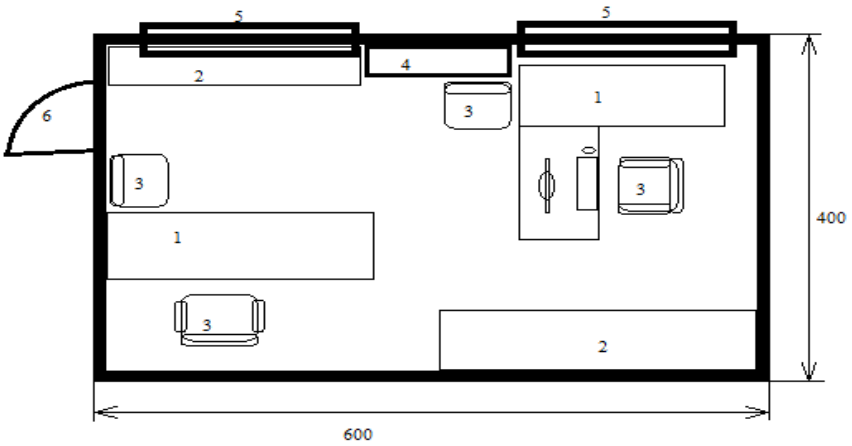


Рисунок 3.1 – План рабочего помещения

Помещение имеет следующие параметры:

- размеры рабочей аудитории: высота – 3 м, ширина – 4 м, длина – 6 м. Общая площадь помещения составляет 24 м2,площадь, занимаемая оборудованием и мебелью ≈ 17 м2;
- помещение по зрительным условиям работы относится к категории легких работ (легкая физическая, категория Iа, работа производится сидя и не требует физического напряжения);
- искусственное освещение – светильники: 2 светильника, в светильнике 2 люминесцентные лампы;
- остекление помещения – одинарное (2 окна размером 2000х2000 мм) без стального переплетения.

Для более безопасного в серверную комнату необходимо поставить автоматическую установку пожаротушения. Что касается типа установки пожаротушения, то здесь существует несколько вариантов:

- водяное пожаротушение
- порошковое пожаротушение
- аэрозольное пожаротушение

- пожаротушение тонко-распыленной водой
- газовое пожаротушение

3.2 Микроклиматические условия

В таблице 3.1 приведены оптимальные нормы параметров микроклимата с учетом периода года. ССБТ для легкой физической работы. Оборудование, установленное в рабочем помещении, не является источником выделения тепла (очень незначительное выделение тепла аппаратурой никаким образом не оказывает влияние на микроклимат рабочего помещения).

Таблица 3.1 - Оптимальные нормы микроклимата для помещений с ПК

Период года	Категория работ	Температура воздуха °С не более	Относительная влажность воздуха, %	Скорость движения воздуха, м/с
Холодный	Легкая – 1а	22-24	40-60	0,1
Теплый	Легкая – 1а	23-25	40-60	0,1

Здание относится к I степени огнестойкости. Рабочее помещение по вопросам пожарной безопасности относится к классу “Д”. В соответствии с типовыми правилами пожарной безопасности административные здания и отдельные помещения, и технологические установки обеспечиваются первичными средствами пожаротушения согласно нормативам.

3.3 Расчет системы кондиционирования

Приведенные оптимальные нормы параметров микроклимата с учетом периода года и географического положения. для легкой физической работы. Оборудование, установленное в рабочем помещении, не является источником выделения тепла (очень незначительное выделение тепла аппаратурой никаким образом не оказывает влияние на микроклимат рабочего помещения). Климатические условия эксплуатации оборудования полностью совпадают с климатическими условиями, нормируемыми для рабочего персонала.

Количество приточного воздуха $L_{пр}$, $м^3$/ч определяем по формуле

$$L_{пр} = Qизб / c * t_{пр} * (t_{выт} - t_{пр}) \quad (3.1)$$

Где $Q_{изб}$ = избыточное выделение явной теплоты, кДж/ч;

с = удельная теплоемкость воздуха, при постоянном давлении, равная с = 1 кДж/ кг* ^oC;

рпр = плотность поступающего в помещение воздуха, равная 1,2 кг/м3;

tвыт = температура удаляемого из помещения воздуха за пределы рабочей или обслуживаемой зоны, ^oC;

tпр = температура приточного воздуха, ^{o}C;

Температура удаляемого из помещения воздуха tвыт оС определяется по формуле:

$$t_{выт} = tРЗ + \Delta t^{*}(h_{вп}-z) \quad (3.2)$$

где $t_{пз}$ – температура в рабочей зоне, которая не должна превышать допустимую по нормам (tпз < tдоп), оС;

$h_{вп}$ - расстояние от пола до центра вытяжных проемов (кондиционера),м;

Н – высота рабочей зоны, м;

Посколько расчет производится для теплого периода года, то примем $t_{пз}$ = 22 ^{o}C

Внутренняя часть кондиционера расположена на высоте h = 2,5 м

$t_{выт}$ = 22 + 1,2*(2,5 – 3) ^{o} C

$t_{пр}$ = 22 – 7 = 15 о С

Величину избыточного выделения явной теплоты Оизб находят на основании баланса теплоты в помещении по формуле

$$Q_{изб} = \sum Q - \sum Q_{ух} \quad (3.3)$$

где ∑Q - суммарное количество поступающей в помещение явной теплоты;

$\sum Q_{ух}$ - суммарное количество уходящей из помещения теплоты;

$$Q_2 = 1000 * N \quad (3.4)$$

где N, мощность расходуемая светильниками, кВт

$$Q_2 = 1000 * 0.28 * 4 = 1120 \text{ кВт}$$

Тепловыделение людей Q3 определяют по формуле:

$$Q_2 = n * q_ч \quad (3.5)$$

где n - число рабочих;

$q_ч$ – количества тепла, выделяемое одним человеком, представлено в таблице 3.2

Таблица 3.2 – Количество тепла, выделяемое одним человеком в зависимости от категории работ и температуры окружающей среды

Категория работ	Тепло, Вт			
	Полное		Явное	
	при 100^0 С	При 350^0 С	при 100^0 С	При 350^0 С

Легкая	180 Вт	145 Вт	150 Вт	5 Вт

$$Q_З = 1 * 145 = 145 \text{ Вт}$$

Количество тепла, поступающего в помещение от солнечной радиации $Q_{ост\ рад}$, определяют по формуле

$$Q_{ост\ рад} = F_{ост} * q_{ост} * A_{ост} \quad (3.6.)$$

Для покрытий определяют по формуле

$$Q_{п\ рад} = F_п * q_п * k_п \quad (3.7.)$$

где $F_{ост}$ и $F_п$ - площадь поверхности и покрытия, $м^2$;

$Q_{ост}$ и $Q_п$ - теплопоступления через 1 $м^2$ поверхности остекления и поверхности покрытия, при коэффициенте теплопередачи, равном 1 Вт/ $м^2*^oC$;

$A_{ост}$ - коэффициент остеклененния;

$k_п$ - коэффициент теплопередачи покрытия 1 Вт/ $м^2*^oC$;

$$F_{ост} = 1,5*1,2*2 = 3,6 \text{ м}^2$$

Окно рабочего помещения направлено на север, поэтому примем значение qост равным 140 Вт/ $м^2*^oC$. Примем Аост = 0,35.

$$Q_{ост\ рад} = 3,6 * 140 * 0,35 = 176,4 \text{ Вт}$$

Среднее значение теплопоступления для покрытия с учетом географической широты примем равным $Q_{п\ рад} = 18$ Вт.

:

$$Q_{ух} = \lambda*S*(t_{выт}-t_{пр}) / \delta \quad (3.8)$$

где λ, теплопроводность стен Вт / м * оС

S – площадь, $м^2$

$$Q_{ух} = 1,2 * 24 * (21,4 - 15) / 0,5 = 368,64 \text{ Вт}$$

Вычислим суммарное количество поступающей в помещение явной теплоты при помощи формулы

$$\sum Q = Q_2 + Q_3 + Q_{ост\ рад} + Q_{п.р.р} \quad (3.9.)$$

$$\sum Q = 1120145 + 176,4 + 18 = 1120,3 \text{ кВт}$$

Так как расчет производится для летнего периода величина избыточного выделения явной теплоты равна:

$$Q_{изб} = 1120,3 \text{ кВт}$$

Вычислим количество приточного воздуха:

$$t_{пр} = 1120,3 / 1*1,2 * (21,4 – 15) = 145,9 \text{ м}^3/\text{ ч}$$

3.4 Расчет искусственного освещения точечным методом

Для расчета искусственного света воспользуемся точечным методом. Для этого посчитаем что лампы в комнате это точечный излучатель.Чтобы вычислить нам понадобится:

- нормативная освещенность – 400 лк.
- высота подвеса светильников над освещаемой поверхностью Н=2,45 м.

Для точки А определяем суммарную условную освещенность всех светильников следующим образом.

Определяем проекцию расстояния на потолок от точки А до светильников. В дальнейшем определяем косинус угла между потолком и прямой.

$$\alpha = arctg(\frac{d_i}{H}) \qquad (3.11)$$

d_i – расстояние от контрольной точки до i-ой лампы.

Н – высота подвеса

На рисунке 3.2 схема освещенности помещения

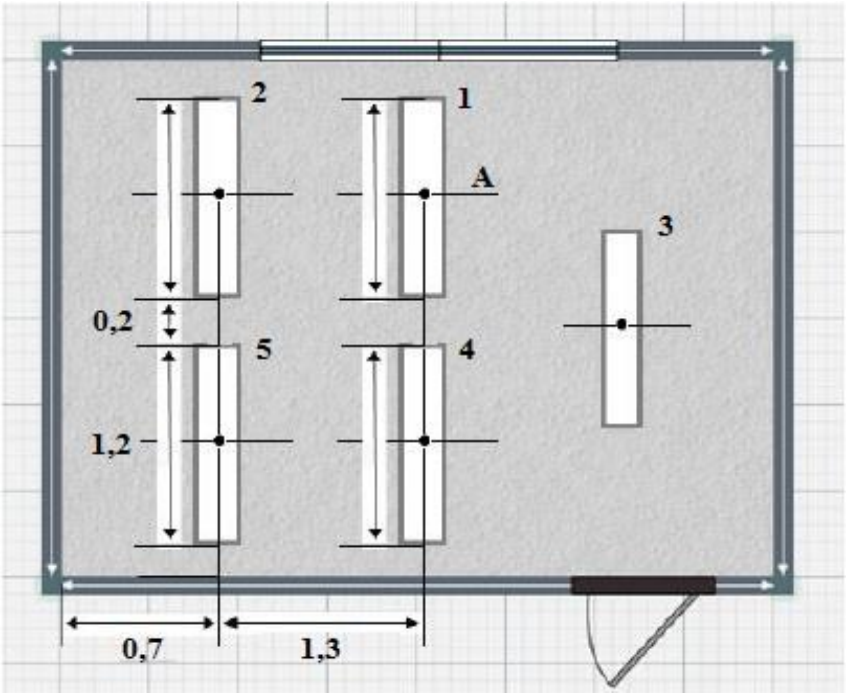


Рисунок 3.2 – Схема освещенности

$$e_i = \frac{I_\alpha \cdot \cos^3(\alpha) * \mu}{k * H^2} \qquad (3.12)$$

Световой поток установленной в светильнике лампы ДРЛ мощностью 100 Вт равен 4750 лм. Конечный результат I_α = 365*(4750/1000) =1735 кд.

$$e_i = \frac{I_\alpha \cdot \cos^3(\alpha) * \mu}{k * H^2} = \frac{1735 * 1 * 1.05}{1.5 * 2{,}25^2} \approx 240 лк$$

Далее рассчитываем относительную освещенность от светильника 3

$$d_3 = \sqrt{(0.2 + 0.6)^2 + 1.3^2} = 1.53$$

$$\alpha = arctg(\frac{1.53}{2.25}) = 35$$

Определяем I_α. По кривой силе света светильников ЛД при условной лампе со световым потоком $Ф_л$ = 1000 лм I_α = 164,25 кд.

I_α = 164,25*(4750/1000) =780,1875 кд.

$$e_i = \frac{I_\alpha \cdot \cos^3(\alpha) * \mu}{k * H^2} = \frac{780{,}1875 * 0.5498 * 1.05}{1.5 * 2{,}25^2} \approx 59 лк$$

Далее рассчитываем относительную освещенность от светильника 2

$$\alpha = arctg(\frac{1.3}{2.25}) = 30$$

Определяем I_α. По кривой силе света светильников ЛД при условной лампе со световым потоком $Ф_л$ = 1000 лм I_α = 291 кд.

I_α = 291*(4750/1000) =1382,25 кд.

$$e_i = \frac{I_\alpha \cdot \cos^3(\alpha) * \mu}{k * H^2} = \frac{1382{,}25 * 0.65 * 1.05}{1.5 * 2{,}25^2} \approx 124 лк$$

4 Технико-экономическое обоснование

Целью данного дипломного проекта является разработка базы данных для учета IT имущества и управления IT инфраструктурой на предприятии. База данных реализована при помощи MS SQL, а интерфейс информационной системы написан на языке C#.

В данном разделе приводится рассмотрение экономической составляющей этого проекта, отражающей затраты.

4.1 Трудовые ресурсы, используемые в работе

В работе над данным проектом задействованы:

- Инженер-разработчик – разработка схемы ПП, разработка алгоритмов и программирование, установка, настройка и сдача отчета.
- Консультант – постановка задачи, подготовка материалов, заполнение информационной системы и сдача отчета.

Количество сотрудников, задействованных в проекте, и их месячная заработная плата представлено в таблице 4.1.

Таблица 4.1 – Данные о сотрудниках

Должность	Количество	Заработная плата в месяц
Инженер-разработчик	1	140 000
Консультант	1	70 000
Итого	2	210 000

4.2 Оборудование, используемое в работе

Характеристики оборудования, используемого в работе, а также его стоимость приведены в таблице 4.2.

Таблица 4.2 – Перечень оборудования

Название оборудования	Характеристики	Количество	Стоимость за единицу с учетом НДС, тенге
Ноутбук HP H0V90EA ProBook 450	Intel Core i5 3230M,4 Gb, 1000 Gb HDD, Radeon HD 7670M	1	283815
Итого		1	283815

4.3 Программное обеспечение, используемое в работе

При разработке приложения используется данное следующее программного обеспечения:

- Windows 10 – операционная система.
- Kaspersky Lab Small Office Security BOX – антивирус.
- MS Visual Studio 2015 Professional – среда для разработки. ПО и соответствующая ему стоимость приведены в таблице 4.3.

Таблица 4.3 – Программное обеспечение, использованное в работе

Наименование	Количество копий	Стоимость с учетом НДС, тенге
Windows 10	1	Предустановлено, цена ПО входит в стоимость ноутбука
Kaspersky Lab Small Office Security BOX	1	43 991 (Годовая подписка)
MS Visual Studio 2015 Professional	1	68991
Итого		112982

4.4 Сроки реализации проекта

Процесс проектирования и разработки приложения включает в себя 5 этапов:

- Постановка задачи и сбор данных.
- Разработка базы данных ПП.
- Разработка интерфейса ПП.
- Тестирование на работоспособность.
- Оформление и сдача отчета, заполнение информационной системы. В таблице 4.4 приведен график реализации проекта.

Таблица 4.4 – Этапы и сроки реализации проекта

Перечень работ	Неделя от начала работ							
	1	2	3	4	5	6	7	8

Окончание таблицы 4.4

1 этап	Постановка задачи, изучение проблем									
	Выбор технологических решений									
	Литература									
2 этап	Создание базы данных									
3 этап	Разработка интефейса									
4 этап	Тестирование ПП									
5 этап	Подготовка материала									
	Заполнение информационной системы									
	Установка, настройка и сдача отчета									

4.5 Расчет фонда оплаты труда

Используя формулу 4.4, рассчитываем заработную плату за один час:

1) Консультант

$$H = 3334/8 = 417 \text{ тенге/час}$$

2) Инженер-разработчик

$$D = 6667/8 = 833 \text{ тенге/час}$$

Длительность цикла в днях по каждому виду работ определяется по формуле

$$t_n = T/q_n * z * K \qquad (4.5)$$

где Т – трудоемкость этапа, норма-час;

q_n – количество исполнителей по этапу;

z – продолжительность рабочего дня, z = 8 часов;

K – коэффициент выполнения норма времени, K = 1,1.

По формуле 4.5 производим расчет длительности цикла. Полученную величину округляем в большую сторону до целых дней

t = 24/1*8*1,1 ≈ 3 дн; Консультант, постановка задачи;

t = 16/1*8*1,1 ≈ 2 дн; Инженер-разработчик, выбор технологических решений;

t = 24/1*8*1,1 ≈ 3 дн; Инженер-разработчик, подбор и изучение литературы;

t = 32/1*8*1,1 ≈ 4 дн; Инженер-разработчик, создание базы данных;

t = 8/1*8*1,1 ≈ 1 дн; Инженер-разработчик, разработка интерфейса;

t = 32/1*8*1,1 ≈ 1 дн; Инженер-разработчик, тестирование;

t = 48/1*8*1,1 ≈ 1 дн; Инженер-разработчик, отладка;

t = 16/1*8*1,1 ≈ 2 дн; Консультант, подготовка материала для заполнения;

t = 16/1*8*1,1 ≈ 2 дн; Инженер-разработчик, подготовка материала для заполнения;

t = 16/1*8*1,1 ≈ 2 дн; Консультант, заполнение информационной системы;

t = 16/1*8*1,1 ≈ 2 дн; Инженер-разработчик, заполнение информационной системы;

В таблице 4.5 приведены сводные результат расчета затрат на основную заработную плату сотрудников.

Таблица 4.5 – сводные результат расчета затрат на основную заработную плату сотрудников.

Наименование этапов работ	Исполнитель	Трудоемкость		Наименование Этапов рабоат	Зар.плата за час работы, тенге	Сумма зар.платы, тенге
		Нормы-часы	% от общей трудоемкости			
Постановка задачи	Консультант	24	6,7	3	417	10008
Выбор технологических решение	Инженер-разработчик	16	4,4	2	833	13328
Изучение литературы	Инженер-разработчик	24	6,7	3	833	19992
Создание базы данных	Инженер-разработчик	32	8,9	4	833	26656

Разработка интерфейса	Инженер-разработчик	64	17,8	8	833	53312
Тестирование ПП	Инженер-разработчик	32	8,9	4	833	26656
Отладка ПП	Инженер-разработчик	48	13,3	6	833	38894
Подготовка материала	Консультант	16	4,4	2	417	6672
	Инженер-разработчик	16	4,4	2	833	13328
Заполнение информа-ционной системы	Консультант	16	4,4	2	417	6672
	Инженер-разработчик	16	4,4	2	833	13328
Установка, настройка и сдача отчета	Инженер-Разработчик	16	4,4	2	833	10008
	Инженер-разработчик	24	8,9	3	417	10008
Итого		344	100	43		253272

Дополнительная заработная плата составляет 10% от основной заработной платы и рассчитывается по формуле

$$З_{доп} = З_{осн} * 0,1 \quad (4.6)$$

Рассчитываем дополнительную заработную плату по формуле 4.6

$$З_{доп} = 253272*0,1 = 25327,2 \text{ тенге}$$

В результате расчетов, согласно формуле 4.2, суммарный фонд оплаты труда составит

$$ФОТ = 253272 + 25327,2 = 278559,2 \text{ тенге}$$

4.6 Суммарные затраты на реализацию проекта

Таким образом, себестоимость разработки данного приложения, согласно формуле 4.1 составляет

$$С + 278599,2 + 27581+27047,5+ 136,5 + 139299,6 + 52000 + 9200 =$$
$$= 533863,8 \text{ тенге}$$

Сводные результаты расчета стоимости разработки приложения и их структура представлены в таблице 4.6

Таблица 4.6 – Затраты на разработку приложения

Наименование затрат	Сумма, тенге
ФОТ	278599,2
Социальный налог	27581
Амортизационные отчисления	27047,5
Затраты на электроэнергию	136,5
Накладные расходы	162249,5
Аренда	52000
Интернет	9200

4.7 Цена реализации проекта

Цена проекта складывается из себестоимости и желаемого чистого дохода, по формуле

$$Ц = С + П \quad (4.15)$$

где С – стоимость приложения;
П – чистый доход.

Для определния начальной цены используется желаемый уровень рентабельность и рассчитывается по формуле

$$Ц_п = С * (1+Р/100) \quad (4.16)$$

где Р – рентабельность
Таким образом на основании формулы 4.15
$Ц_п = 533863,8 * (1+25/100) = 667329,75$ тенге

Цена реализации проекта складывается из цены проекта и НДС, рассчитывается по формуле

$$Ц_р = Ц_п + НДС \quad (4.17)$$

где НДС – налог на добавленную стоимость.

Согласно Налоговому Кодексу Республики Казахстан НДС составляет 12%, то есть в данном случае равен

$$НДС = 80079,57 \text{ тенге}$$

В итоге получаем цену реализации проекта, рассчитывая по формуле 4.17

$$Ц_р = 667329,75 + 80079,57 = 747409,32 \text{ тенге}$$

Вывод

Итоговая стоимость разработки базы данных для учета IT имущества и управления IT инфраструктурой на предприятии составила 747409,32 тенге, в которую заложены все возможные затраты при разработке программного продукта.

Наибольшую стоимость проекта составляют затраты на оплату труда сотрудников, более 59%, задействованных в процессе данной разработки.

Данное ПО содержит в себе полный набор инструментов для автоматизации ключевых ИТ-задач. Комплексность системы обеспечивается глубокой интегрированностью ее элементов. Настройка системы не требует специальной квалификации и навыков программирования, легко изменяется под потребности пользователя.

Разработка информационной системы является дорогим проектом, требующих больших интеллектуальных и финансовых затрат. Данное ПО является эффективным благодаря наиболее легкому учету оборудования.

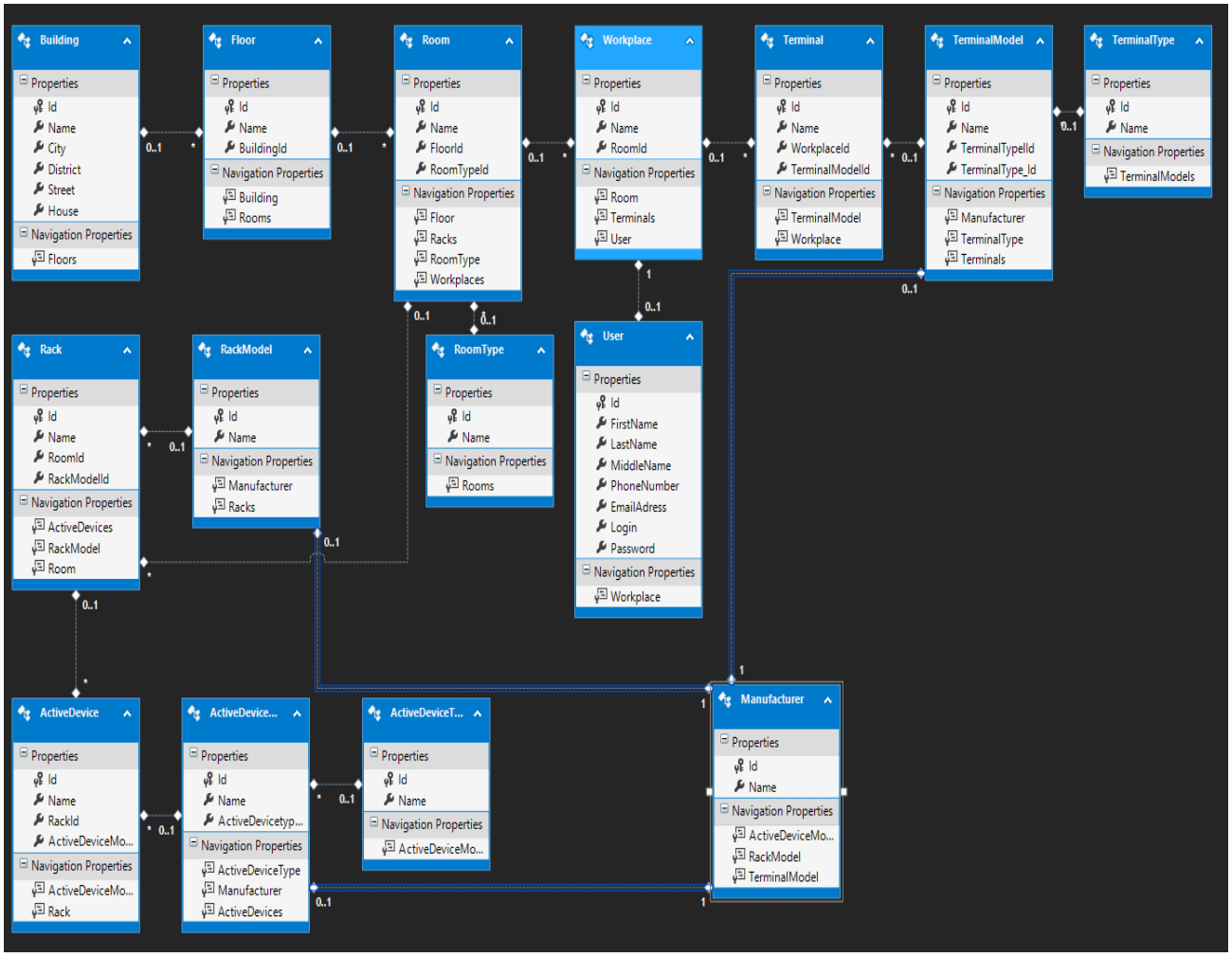
Заключение

Во время выполнения дипломного проекта были изучены основные принципы разработки базы данных, приложения к базе, а также требований, предъявляемых к ним. Помимо этого, было также установлено, что исследуемая тема является актуальной в реалиях современного рынка программных продуктов. Для доступа к данным был разработан пользовательский интерфейс.

В разделе «Безопасность жизнедеятельности» были рассчитаны необходимые условия для организации соответствующего всем установленным нормам, рабочего места для сотрудников, занятых в реализации проекта. Рассчитано необходимое естественное и искусственное освещение, а также система кондиционирование требующееся для комфортной работы без нанесения вреда здоровью сотрудников.

Была рассчитана общая стоимость продукта, на основе суммирования различных видов затрат, которые возможны при производстве. Итоговая стоимость разработки базы данных для учета IT имущества и управления IT инфраструктурой на предприятии составила 747409,32 тенге, в которую заложены все возможные затраты при разработке программного продукта. Наибольшую стоимость проекта составляют затраты на оплату труда сотрудников, более 59%, задействованных в процессе данной разработки.

Приложение А



Пример кода

Главное окно

```
<Window x:Class="diplom.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:diplom"
        mc:Ignorable="d"
        Title="Diplom" Height="600" Width="850"
        WindowStartupLocation="CenterScreen">

    <Window.Resources>
        <Style x:Key="ProjectStyle" TargetType="{x:Type TextBlock}">
            <Setter Property="Background" Value="Firebrick"/>
```

```
            <Setter Property="Foreground" Value="White"/>
            <Setter Property="FontSize" Value="12"/>
            <Setter Property="Padding" Value="2,2,2,2"/>
            <Setter Property="Margin" Value="2,2,2,2"/>

            <Style.Triggers>
               <Trigger Property="IsFocused" Value="True">
                  <Setter Property="Background" Value="Firebrick"/>
                  <Setter Property="Foreground" Value="White"/>
               </Trigger>
               <Trigger Property="IsFocused" Value="False">
                  <Setter Property="Background" Value="White"/>
                  <Setter Property="Foreground" Value="Firebrick"/>
               </Trigger>
            </Style.Triggers>
         </Style>

         <DataGrid x:Key="my_DataGridFloors" HorizontalGridLinesBrush="DarkGray"
VerticalGridLinesBrush="Transparent" AutoGenerateColumns="False"
AlternatingRowBackground="#FFCEEFFF" BorderBrush="Transparent"
BorderThickness="0">
            <DataGrid.Columns>
               <DataGridTextColumn Binding="{Binding Name}" Header="Этаж"
Width="500" />
            </DataGrid.Columns>
         </DataGrid>

         <DataGrid x:Key="my_DataGridRooms" HorizontalGridLinesBrush="DarkGray"
VerticalGridLinesBrush="Transparent" AutoGenerateColumns="False"
AlternatingRowBackground="#FFCEEFFF" BorderBrush="Transparent"
BorderThickness="0">
            <DataGrid.Columns>
               <DataGridTextColumn Binding="{Binding Name}" Header="Помещение"
Width="500" />
            </DataGrid.Columns>
         </DataGrid>

      </Window.Resources>

      <Grid x:Name="m_grid" ShowGridLines="True">
         <Grid.RowDefinitions>
            <RowDefinition Height="25" />
            <RowDefinition></RowDefinition>
            <RowDefinition Height="45" />
         </Grid.RowDefinitions>
         <Grid.ColumnDefinitions>
            <ColumnDefinition Width="220" />
            <ColumnDefinition Width="Auto" />
            <ColumnDefinition Width="*" />
```

```
        </Grid.ColumnDefinitions>

        <Menu x:Name="MainMenu" Grid.Column="0" Grid.Row="0" Grid.ColumnSpan="3"
Height="25" VerticalAlignment="Top">
            <MenuItem Header="Файл">
                <MenuItem Header="Выход" Click="menuExit_Click" />
            </MenuItem>
            <MenuItem Header="Справочники" >
                <MenuItem Header="Добавить типы комнат"
Click="menuShowTypeRoom_Click" ></MenuItem>
                <MenuItem Header="Добавить тип активных устройств"
Click="menuShowActiveDeviceType_Click" ></MenuItem>
                <MenuItem Header="Добавить тип оконечных устройств"
Click="menuShowTerminalType_Click" ></MenuItem>
                <MenuItem Header="Добавить производителя"
Click="menuShowManufacturer_Click" ></MenuItem>
            </MenuItem>
            <MenuItem Header="Объекты">
                <MenuItem Header="Добавить здание" Click="AddBuilding_Click"
></MenuItem>
                <MenuItem Header="Добавить этаж" Click="AddFloor_click" ></MenuItem>
                <MenuItem Header="Добавить комнату" Click="AddRoom_click"
></MenuItem>
                <Separator/>
                <MenuItem Header="Добавить рабочее место" Click="AddWorkplace_click"
></MenuItem>
                <MenuItem Header="Добавить пользователя"
Click="menuAddUsers"></MenuItem>
                <Separator/>
                <MenuItem Header="Добавить шкафы" Click="AddRack_click" ></MenuItem>
                <MenuItem Header="Добавить модели шкафов" Click="AddRackModel_click"
></MenuItem>
                <Separator/>
                <MenuItem Header="Добавить активное устройство"
Click="AddActiveDevice_click" />
                <MenuItem Header="Добавить модель активного устройства"
Click="AddActiveDeviceModel_click" ></MenuItem>
                <MenuItem Header="Добавить оконечное устройство"
Click="AddTerminal_click" ></MenuItem>
                <MenuItem Header="Добавить модель оконечного устройства"
Click="AddTerminalModel_click" ></MenuItem>
            </MenuItem>
            <MenuItem Header="Отчеты" >
                <MenuItem Header="Отчет по оконечным устройствам"
Click="reportTerminals_Click"></MenuItem>
            </MenuItem>
        </Menu>
```

```
      <TreeView x:Name="Tree_MainWindow" Grid.Column="0" Grid.Row="1"
Grid.RowSpan="2" ItemsSource="{Binding Path= Buildings, Mode=TwoWay}"
MouseDoubleClick="to_tv_Click" GotFocus="testDock" >
         <TreeView.ItemTemplate>
            <HierarchicalDataTemplate ItemsSource="{Binding Path=Floors, Mode=TwoWay
}" >
               <TextBlock Text="{Binding Path = Name}" Style="{StaticResource
ProjectStyle}" />
               <HierarchicalDataTemplate.ItemTemplate>
                  <HierarchicalDataTemplate ItemsSource="{Binding Path=Rooms,
Mode=TwoWay}">
                     <TextBlock Text="{Binding Path = Name}" Style="{StaticResource
ProjectStyle}" />
                  </HierarchicalDataTemplate>
               </HierarchicalDataTemplate.ItemTemplate>
            </HierarchicalDataTemplate>
         </TreeView.ItemTemplate>
      <TreeView.ItemContainerStyle>
            <Style TargetType="TreeViewItem">
               <!-- <EventSetter Event="PreviewMouseLeftButtonDown"
Handler="to_tv_Click" /> -->
               <Setter Property="IsSelected" Value="{Binding IsSelected, Mode=TwoWay}"
/>
               <Setter Property="IsExpanded" Value="{Binding IsExpanded, Mode=TwoWay
}" />
            </Style>
         </TreeView.ItemContainerStyle>
      </TreeView>

      <GridSplitter Grid.Column="1" Grid.Row="1" ShowsPreview="False" Width="5"
HorizontalAlignment="Center" VerticalAlignment="Stretch" />

      <DockPanel x:Name="m_Dock" Grid.Column="2" Grid.Row="1">
         <DataGrid x:Name="usersGrid" HorizontalGridLinesBrush="DarkGray"
AutoGenerateColumns="False" AlternatingRowBackground="#FFCEEFFF"
BorderBrush="Transparent" BorderThickness="0">
            <DataGrid.Columns>
               <DataGridTextColumn Binding="{Binding Name}" Header="Здание"
Width="200" />
               <DataGridTextColumn Binding="{Binding City}" Header="Город"
Width="100" />
               <DataGridTextColumn Binding="{Binding District}" Header="Район"
Width="120" />
               <DataGridTextColumn Binding="{Binding Street}" Header="Улица"
Width="130" />
               <DataGridTextColumn Binding="{Binding House}" Header="Дом"
Width="Auto" />
            </DataGrid.Columns>
         </DataGrid>
```

```
        </DockPanel>

        <DockPanel Grid.Row="2" Grid.Column="2" Background="#FF407893">
          <Button x:Name="updateButton" Content="Обновить" Click="updateButton_Click"
Canvas.Left="173" Margin="120,0,0,0" Canvas.Top="10" VerticalAlignment="Center"
HorizontalAlignment="Left" />
          <Button x:Name="deleteButton" Content="Удалить" Click="deleteButton_Click"
Canvas.Left="325" Margin="100,0,0,0" Canvas.Top="10" VerticalAlignment="Center" />
        </DockPanel>
    </Grid>
</Window>
```

Добавление пользователя

```
<Window x:Class="diplom.AddUserWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:WpfApplication2"
        mc:Ignorable="d"
        Title="Добавление нового пользователя" Height="450" Width="390"
        WindowStartupLocation="CenterOwner" Closed="addUser_Close">
    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition></RowDefinition>
            <RowDefinition Height="100" />
        </Grid.RowDefinitions>


        <Label x:Name="label_lastname" Content="Фамилия" HorizontalAlignment="Left"
Height="25" Margin="40,35,0,0" VerticalAlignment="Top" Width="100"/>
        <Label x:Name="label_firstname" Content="Имя" HorizontalAlignment="Left"
Height="25" Margin="40,65,0,0" VerticalAlignment="Top" Width="100"/>
        <Label x:Name="label_middlename" Content="Отчество"
HorizontalAlignment="Left" Height="25" Margin="40,95,0,0" VerticalAlignment="Top"
Width="100"/>
        <Label x:Name="label_phone" Content="Телефон" HorizontalAlignment="Left"
Height="25" Margin="40,125,0,0" VerticalAlignment="Top" Width="100"/>
        <Label x:Name="label_mail" Content="E-mail" HorizontalAlignment="Left"
Height="25" Margin="40,155,0,0" VerticalAlignment="Top" Width="100"/>
        <Label x:Name="label_login" Content="Логин" HorizontalAlignment="Left"
Height="25" Margin="40,185,0,0" VerticalAlignment="Top" Width="100"/>
        <Label x:Name="label_pass" Content="Пароль" HorizontalAlignment="Left"
Height="25" Margin="40,215,0,0" VerticalAlignment="Top" Width="100"/>
        <Label x:Name="label_lastname_Copy" Content="Рабочее место"
HorizontalAlignment="Left" Height="25" Margin="40,245,0,0" VerticalAlignment="Top"
Width="100"/>
        <TextBox x:Name="textBox_lastname" HorizontalAlignment="Left" Height="25"
Margin="165,35,0,0" VerticalAlignment="Top" Width="160" MaxLines="1"
```

```
MaxLength="49" TextWrapping="Wrap" HorizontalContentAlignment="Center"
VerticalContentAlignment="Center"/>
      <TextBox x:Name="textBox_firstname" HorizontalAlignment="Left" Height="25"
Margin="165,65,0,0" VerticalAlignment="Top" Width="160" MaxLines="1"
MaxLength="49" TextWrapping="Wrap" HorizontalContentAlignment="Center"
VerticalContentAlignment="Center"/>
      <TextBox x:Name="textBox_middlename" HorizontalAlignment="Left" Height="25"
Margin="165,95,0,0" VerticalAlignment="Top" Width="160" MaxLines="1"
MaxLength="49" TextWrapping="Wrap" HorizontalContentAlignment="Center"
VerticalContentAlignment="Center"/>
      <TextBox x:Name="textBox_phone" HorizontalAlignment="Left" Height="25"
Margin="165,125,0,0" VerticalAlignment="Top" Width="160" MaxLines="1"
MaxLength="49" TextWrapping="Wrap" HorizontalContentAlignment="Center"
VerticalContentAlignment="Center"/>
      <TextBox x:Name="textBox_mail" HorizontalAlignment="Left" Height="25"
Margin="165,155,0,0" VerticalAlignment="Top" Width="160" MaxLines="1"
MaxLength="49" TextWrapping="Wrap" HorizontalContentAlignment="Center"
VerticalContentAlignment="Center"/>
      <TextBox x:Name="textBox_login" HorizontalAlignment="Left" Height="25"
Margin="165,185,0,0" VerticalAlignment="Top" Width="160" MaxLines="1"
MaxLength="49" TextWrapping="Wrap" HorizontalContentAlignment="Center"
VerticalContentAlignment="Center"/>
      <TextBox x:Name="textBox_pass" HorizontalAlignment="Left" Height="25"
Margin="165,215,0,0" VerticalAlignment="Top" Width="160" MaxLines="1"
MaxLength="49" TextWrapping="Wrap" HorizontalContentAlignment="Center"
VerticalContentAlignment="Center"/>
      <ComboBox x:Name="comboBox" HorizontalAlignment="Left"
Margin="165,245,0,0" VerticalAlignment="Top" Width="160" Height="25"
VerticalContentAlignment="Center" HorizontalContentAlignment="Center"/>

      <DockPanel Grid.Row="1" Background="#FF407893">
        <Button x:Name="addUser_buttonAdd" Content="Добавить"
HorizontalAlignment="Left"  VerticalAlignment="Center" Width="75"
Click="addUserClick" Margin="50,0,0,0"/>
        <Button x:Name="addUser_buttonCancel" Content="Отмена"
HorizontalAlignment="Right"  VerticalAlignment="Center" Width="75"
Click="addUser_buttonCancel_Click" Margin="0,0,50,0"/>
      </DockPanel>
  </Grid>
</Window>
```

Справочник по производителям

```
<Window x:Class="diplom.ShowManufacturer"
      xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
      xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
      xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
      xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
      xmlns:local="clr-namespace:WpfApplication2"
      mc:Ignorable="d"
```

```
        Title="Добавление производителей" Height="450" Width="370"
        WindowStartupLocation="Manual" Left="200" Top="200">
    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition></RowDefinition>
            <RowDefinition Height="100" />
        </Grid.RowDefinitions>

        <DockPanel Grid.Row="0">
            <DataGrid x:Name="gridManufacturer" HorizontalGridLinesBrush="DarkGray"
AutoGenerateColumns="False"  AlternatingRowBackground="#FFCEEFFF"
VerticalGridLinesBrush="Transparent" BorderBrush="Transparent" BorderThickness="0">
                <DataGrid.Columns>
                    <DataGridTextColumn Binding="{Binding Name}" Header="Производитель"
Width="340"  />
                </DataGrid.Columns>
            </DataGrid>
        </DockPanel>
        <DockPanel Grid.Row="1" Background="#FF407893">
            <Button x:Name="buttonAdd" Content="Обновить" Height="26"
Margin="50,0,0,0" Width="74" HorizontalAlignment="Left" VerticalAlignment="Center"
Click="buttonAddManufacturer_Click"/>
            <Button x:Name="buttonDelete" Content="Удалить" Height="26"
Margin="0,0,50,0" Width="74" HorizontalAlignment="Right" VerticalAlignment="Center"
Click="buttonDeleteManufacturer_Click"/>

        </DockPanel>
    </Grid>
</Window>
```

Пример заполнения базы

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Data.Entity;

namespace diplom
{
    class DbContextInitializer : DropCreateDatabaseAlways<UserContext>
    {
        protected override void Seed (UserContext db)
        {
            db = new UserContext();

            // Заполнение базы первичными данными
            // Здания
```

```
        Building build_0 = new Building() { Name = "Здание головного подразделения",
City = "Алматы", District = "Бостандыкский", Street = "Джандосова", House = "115" };
        Building build_1 = new Building() { Name = "Здание филиала", City =
"Караганда", District = "Октябрьский", Street = "Железнодорожная", House = "20" };
        db.Buildings.Add(build_0);
        db.Buildings.Add(build_1);

        //Этажи
        Floor floor_0 = new Floor() { Name = "1 этаж", Building = build_0 };
        Floor floor_1 = new Floor() { Name = "2 этаж", Building = build_0 };
        Floor floor_2 = new Floor() { Name = "3 этаж", Building = build_0 };
        Floor floor_3 = new Floor() { Name = "1 этаж", Building = build_1 };
        Floor floor_4 = new Floor() { Name = "2 этаж", Building = build_1 };
        db.Floors.AddRange(new List<Floor>() {floor_0, floor_1 ,floor_2, floor_3, floor_4
});

        //Типы помещений
        RoomType room_type_0 = new RoomType() { Name = "Холл"};
        RoomType room_type_1 = new RoomType() { Name = "Коридор" };
        RoomType room_type_2 = new RoomType() { Name = "Кабинет" };
        RoomType room_type_3 = new RoomType() { Name = "Гермозона" };
        db.RoomTypes.AddRange(new List<RoomType>() { room_type_0, room_type_1,
room_type_2, room_type_3 });

        //Помещения
        Room room_0 = new Room() {Name = "Холл", Floor = floor_0, RoomType =
room_type_0 };
        Room room_1 = new Room() { Name = "Коридор", Floor = floor_0, RoomType =
room_type_1 };
        Room room_2 = new Room() { Name = "Кабинет №100", Floor = floor_0,
RoomType = room_type_2 };
        Room room_3 = new Room() { Name = "Кабинет №101", Floor = floor_0,
RoomType = room_type_2 };
        Room room_4 = new Room() { Name = "Кабинет №102", Floor = floor_0,
RoomType = room_type_2 };
        db.Rooms.AddRange(new List<Room>() { room_0, room_1, room_2, room_3,
room_4 });

        // Рабочее место
        Workplace workplace_0 = new Workplace() { Name = "Рабочее место №100-1",
Room = room_2 };
        Workplace workplace_1 = new Workplace() { Name = "Рабочее место №100-2",
Room = room_2 };
        Workplace workplace_2 = new Workplace() { Name = "Рабочее место №101-1",
Room = room_3 };
        Workplace workplace_3 = new Workplace() { Name = "Рабочее место №101-2",
Room = room_3 };
        db.Workplaces.AddRange(new List<Workplace>() { workplace_0, workplace_1,
workplace_2, workplace_3 });
```

```
        // Пользователи
        User user_0 = new User() { FirstName = "Руслан", LastName = "Алимбетов",
MiddleName = "Эрикович", PhoneNumber = "8705", EmailAdress = "retard@mail.com",
Login = "admin", Password = "admin" , Workplace = workplace_0 };
        User user_1 = new User() { FirstName = "Егор", LastName = "Киберспортсмен",
MiddleName = "Хаймрович", PhoneNumber = "8706", EmailAdress =
"cybersport@mail.com", Login = "egor", Password = "egor", Workplace = workplace_1 };
        db.Users.AddRange(new List<User>() { user_0, user_1 });

        TerminalType tt_0 = new TerminalType() { Name = "Компьютер" };
        TerminalType tt_1 = new TerminalType() { Name = "Ноутбук" };
        TerminalType tt_2 = new TerminalType() { Name = "Принтер" };
        TerminalType tt_3 = new TerminalType() { Name = "Сканер" };
        db.TerminalTypes.AddRange(new List<TerminalType>() { tt_0, tt_1, tt_2, tt_3 });

        // Производитель
        Manufacturer m_0 = new Manufacturer() { Name = "Asus" };
        Manufacturer m_1 = new Manufacturer() { Name = "Dell" };
        Manufacturer m_2 = new Manufacturer() { Name = "HP" };
        db.Manufacturers.AddRange(new List<Manufacturer>() { m_0, m_1, m_2 });

        //Модели Оконечных устройств
        TerminalModel tm_0 = new TerminalModel() { Name = "ASUS_X554LJ",
TerminalType = tt_1 , Manufacturer = m_0};
        TerminalModel tm_1 = new TerminalModel() { Name = "ASUS_F200MA",
TerminalType = tt_1, Manufacturer = m_0 };
        TerminalModel tm_2 = new TerminalModel() { Name = "ASUS_GL752VW",
TerminalType = tt_1, Manufacturer = m_0 };
        TerminalModel tm_3 = new TerminalModel() { Name = "DELL_720",
TerminalType = tt_2, Manufacturer = m_1 };
        TerminalModel tm_4 = new TerminalModel() { Name = "DELL_725",
TerminalType  = tt_2, Manufacturer = m_1 };
        TerminalModel tm_5 = new TerminalModel() { Name = "HP_ScanJet_G4010",
TerminalType = tt_3, Manufacturer = m_2 };
        TerminalModel tm_6 = new TerminalModel() { Name = "HP_ScanJet_8270",
TerminalType = tt_3, Manufacturer = m_2 };
        TerminalModel tm_7 = new TerminalModel() { Name = "HP_Scanjet_N6350",
TerminalType = tt_3, Manufacturer = m_2 };
        db.TerminalModels.AddRange(new List<TerminalModel>() { tm_0, tm_1, tm_2,
tm_3, tm_4, tm_5, tm_6, tm_7 });

        //Оконечные устройства
        Terminal t_0 = new Terminal() { Name = "Оконечное устройство №100",
TerminalModel = tm_1, Workplace = workplace_0 };
        Terminal t_1 = new Terminal() { Name = "Оконечное устройство №101",
TerminalModel = tm_3, Workplace = workplace_0 };
        Terminal t_2 = new Terminal() { Name = "Оконечное устройство №102",
TerminalModel = tm_7, Workplace = workplace_0 };
```

```
            Terminal t_3 = new Terminal() { Name = "Оконечное устройство №103",
TerminalModel = tm_1, Workplace = workplace_0 };
            Terminal t_4 = new Terminal() { Name = "Оконечное устройство №200",
TerminalModel = tm_5, Workplace = workplace_1 };
            Terminal t_5 = new Terminal() { Name = "Оконечное устройство №201",
TerminalModel = tm_4, Workplace = workplace_1 };
            Terminal t_6 = new Terminal() { Name = "Оконечное устройство №202",
TerminalModel = tm_2, Workplace = workplace_1 };
            Terminal t_7 = new Terminal() { Name = "Оконечное устройство №203",
TerminalModel = tm_3, Workplace = workplace_1 };
            Terminal t_8 = new Terminal() { Name = "Оконечное устройство №300",
TerminalModel = tm_5, Workplace = workplace_2 };
            Terminal t_9 = new Terminal() { Name = "Оконечное устройство №301",
TerminalModel = tm_0, Workplace = workplace_2 };
            Terminal t_10 = new Terminal() { Name = "Оконечное устройство №302",
TerminalModel = tm_3, Workplace = workplace_2 };
            Terminal t_11 = new Terminal() { Name = "Оконечное устройство №303",
TerminalModel = tm_5, Workplace = workplace_2 };
            Terminal t_12 = new Terminal() { Name = "Оконечное устройство №400",
TerminalModel = tm_1, Workplace = workplace_3 };
            Terminal t_13 = new Terminal() { Name = "Оконечное устройство №401",
TerminalModel = tm_3, Workplace = workplace_3 };
            Terminal t_14 = new Terminal() { Name = "Оконечное устройство №402",
TerminalModel = tm_6, Workplace = workplace_3 };
            Terminal t_15 = new Terminal() { Name = "Оконечное устройство №403",
TerminalModel = tm_7, Workplace = workplace_3 };
            db.Terminals.AddRange(new List<Terminal>() { t_0, t_1, t_2, t_3, t_4, t_5, t_6, t_7,
t_8, t_9, t_10, t_11, t_12, t_13, t_14, t_15 });
            db.SaveChanges();
        }
    }
}
```

Примеры классов

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
namespace diplom
{
    public class Building //: Test_event
    {
        // Аттрибуты (поля) здания
        public int Id { get; set; }
        public virtual string Name { get; set; }
        public string City { get; set; }
```

```
        public string District { get; set; }
        public string Street { get; set; }
        public string House { get; set; }

        // Поле - связь одно здание к много этажей
        public virtual ICollection<Floor> Floors { get; set; }

        // Конструктор класса, в него входит список этажей
        public Building()
        {
            Floors = new List<Floor>();
        }
    }
}
namespace diplom
{
    public class Floor //: Test_event
    {
        // Аттрибуты (поля) этажа
        public int Id { get; set; }
        public virtual string Name { get; set; }

        // Поля - связь много этажей к одному зданию
        public int? BuildingId { get; set; }
        public virtual Building Building { get; set; }

        // Поле - связь один этаж к много помещений
        public ICollection<Room> Rooms { get; set; }

        // Конструктор включает в себя список помещений
        public Floor()
        {
            Rooms = new List<Room>();
        }

    }
}
namespace diplom
{
    public class Room
    {
        // Аттрибуты (поля) помещения
        public int Id { get; set; }
        public string Name { get; set; }

        // Поля - связь много помещений к одному этажу
        // Поля - связь много помещений к одному типу помещений
        public int? FloorId { get; set; }
        public Floor Floor { get; set; }
```

```
        public int? RoomTypeId { get; set; }
        public RoomType RoomType { get; set; }

        // Поля - связь одно помещение к множеству рабочих мест и коммуникационных
шкафов
        public ICollection<Workplace> Workplaces { get; set; }
        public ICollection<Rack> Racks { get; set; }

        // Конструктор включает в себя списки рабочих мест и коммуникационных
шкафов
        public Room()
        {
            Workplaces = new List<Workplace>();
            Racks = new List<Rack>();
        }

    }
}
```

Отчет по оконечным устройствам

```
<Window x:Class="diplom.ReportTerminalWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:WpfApplication2"
        mc:Ignorable="d"
        Title="Отчет по оконечным устройствам" Height="600" Width="760"
        WindowStartupLocation="CenterOwner">
    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition></RowDefinition>
            <RowDefinition Height="100" />
        </Grid.RowDefinitions>

        <DockPanel Grid.Row="0">
            <DataGrid x:Name="reportGrid" HorizontalGridLinesBrush="DarkGray"
AutoGenerateColumns="False" AlternatingRowBackground="#FFCEEFFF"
BorderBrush="Transparent" BorderThickness="0">
                <DataGrid.Columns>
                    <DataGridTextColumn Binding="{Binding Name}" Header="Идентификатор"
Width="300" IsReadOnly="True" />
                    <DataGridTextColumn Binding="{Binding Model}" Header="Модель"
Width="150" IsReadOnly="True"/>
                    <DataGridTextColumn Binding="{Binding Workplace}"
Header="Размещение" Width="150" IsReadOnly="True"/>
```

```
                <DataGridTextColumn Binding="{Binding Other}" Header="Параметры"
Width="150" IsReadOnly="True"/>
            </DataGrid.Columns>
        </DataGrid>
    </DockPanel>
    <Canvas Grid.Row="1" Background="#FF407893">
        <ComboBox x:Name="comboBox_filter" HorizontalAlignment="Left"
Margin="148,37,0,0" Grid.Row="1" VerticalAlignment="Top" Width="120" Height="24"
SelectedIndex="0">
            <ComboBoxItem Content="Модель" HorizontalAlignment="Left"
Width="118"/>
            <ComboBoxItem Content="Помещение" HorizontalAlignment="Left"
Width="118"/>
        </ComboBox>
        <Label x:Name="label" Content="Фильтровать по" HorizontalAlignment="Left"
Margin="40,36,0,0" Grid.Row="1" VerticalAlignment="Top" Width="103" Height="24"
HorizontalContentAlignment="Center" VerticalContentAlignment="Center"/>
        <Label x:Name="label_Copy" Content="содержит" HorizontalAlignment="Left"
Margin="273,36,0,0" Grid.Row="1" VerticalAlignment="Top" Width="82" Height="24"
HorizontalContentAlignment="Center" VerticalContentAlignment="Center"/>
        <TextBox x:Name="textBox_filter" HorizontalAlignment="Left" Height="24"
Margin="360,37,0,0" Grid.Row="1" TextWrapping="Wrap" VerticalAlignment="Top"
Width="181" VerticalContentAlignment="Center"/>
        <Button x:Name="button_filter" Content="Применить фильтр"
HorizontalAlignment="Left" Margin="571,37,0,0" Grid.Row="1" VerticalAlignment="Top"
Width="123" Height="24" Click="applyFilter_click"/>
    </Canvas>
  </Grid>
</Window>


namespace diplom
{
  /// <summary>
  /// Логика взаимодействия для ReportTerminalWindow.xaml
  /// </summary>
  public partial class ReportTerminalWindow : Window
  {
    UserContext db;
    string param;

    public ReportTerminalWindow()
    {
      InitializeComponent();
      db = new UserContext();


      param = "";
      Filter_model(param);
```

```
        }

        public void Filter_model(string param)
        {
            List<Terminal> terminal_lst = db.Terminals
                .Include(c => c.TerminalModel)
                .Include(c => c.Workplace)
                .Where(c => c.TerminalModel.Name.Contains(param))
                .ToList();

            List<ReportTerminal> report_lst = new List<ReportTerminal>();
            foreach (Terminal t in terminal_lst)
            {
                report_lst.Add(new ReportTerminal() { Name = t.Name, Model =
t.TerminalModel.Name, Workplace = t.Workplace.Name });
            }

            reportGrid.ItemsSource = report_lst;
        }

        public void Filter_place(string param)
        {
            List<Terminal> terminal_lst = db.Terminals
                .Include(c => c.TerminalModel)
                .Include(c => c.Workplace)
                .Where(c => c.Workplace.Name.Contains(param))
                .ToList();

            List<ReportTerminal> report_lst = new List<ReportTerminal>();
            foreach (Terminal t in terminal_lst)
            {
                report_lst.Add(new ReportTerminal() { Name = t.Name, Model =
t.TerminalModel.Name, Workplace = t.Workplace.Name });
            }

            reportGrid.ItemsSource = report_lst;
        }

        private class ReportTerminal
        {
            public string Name { get; set; }
            public string Model { get; set; }
            public string Workplace { get; set; }
            public string Other { get; set; }
         }

        private void applyFilter_click(object sender, RoutedEventArgs e)
```

```
        {
            if (comboBox_filter.Text != null)
            {
                if (comboBox_filter.Text == "Модель")
                {
                    Filter_model(textBox_filter.Text);
                }
                if (comboBox_filter.Text == "Помещение")
                {
                    Filter_place(textBox_filter.Text);
                }
            }

        }
    }


}
```

Список литературы

1 Бен Чанг, Марк Скардина, Стефан Киритцов Oracle9i XML. Разработка приложений электронной коммерции с использованием технологии XML, изд. Лори, 2003.-492 с.

2 Питер Роб, Карлос Коронелл. Система баз данных: проектирование, реализация и управление. –5-е изд. – СПб.: БХВ-Петербург, 2004. - 854с.

3 Санжей Мишра, Алан Бьюли Секреты Oracle SQL, изд. Символ, 2003.-360 с.

4 Сатимова Е.Г. Проектирование баз данных. Методические указания к выполнению лабораторных работ (для студентов всех специальностей). – Алматы: АИЭС, 2009

5 Алан Саймон Стратегические технологии БД. – М.: Финансы и статистика, 1999. – 484 с.

6 Хансен Г., Хансен Д. Базы данных. Разработка и управление. – М.: Бином, 2000. – 704 с.

7 Дж. Ульман, Дж. Видом. Введение в системы баз данных. – М.: Лори.- 2000. – 374 с.

8 Ревунков Г.И., Самохвалов Э.Н., Чистов В.В. Базы и банки данных и знаний: Учебник для вузов по специальности АСУ. – М.:Высшая школа, 1992. – 367 с.

9 Мещеряков Е.В., Хомоненко А.Д. Публикация баз данных в Интернете. – СПб.: БХВ-Петербург, 2001. – 572 с.

10 C# для приложений (версия 5) в подлиннике П. Сканна и др – СПб.: «BHV-Санкт-Петербург», 97. – 468 с.

11 Бекаревич Ю.Б., Пушкина Н.В. СУБД Access для Windows 95 в примерах. – СПб.:BHV, 1997. – 400с.

12 Базылов К.Б., Алибаева С.А., Бабич А.А. Методические указания по выполнению экономического раздела выпускной работы бакалавров. – Алматы: АИЭС, 2008.

13 Экономика предприятия./Под ред. Горфинкеля В.Я., Швандара В.А. 4-е изд., – М.:Юнити, 2007.-670 с.

14 Волков О.И. Экономика предприятия. Курс лекций. – М.: Инфра-М, 2006.-328 с.

15 Абдимуратов Ж.С., Мананбаева С.Е. Безопасность жизнедеятельности. Методические указания к выполнению раздела «Расчет производственного освещения» в выпускных работах для всех специальностей. Бакалавриат. А.: АИЭС, 2009.

16 СНиП РК 2.04-05-2002. Естественное и искусственное освещение. Общие требования. – Алматы, 2002.

17 Сивков В.П., Смирнов С.Г., Козьяков А.Ф. и др. Сборник типовых расчетов по курсу «Охрана труда». – М.: МВТУ, 1979. – 79с.

18 Грузинов В.П., Грибов В.Д. Экономика предприятия. –2-е изд.,– Москва, 2001.