МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН

Некоммерческое акционерное общество
АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ

кафедра Компьютерных технологий

«Допущен к защите»
Заведующий кафедрой Купалбаев З.К.
д. ф.-м. наук, проф.
(Ф.И.О., ученая степень, звание)
_____ «_____»_____ **20__ г.**
(подпись)

ДИПЛОМНЫЙ ПРОЕКТ

На тему: Кроссплатформенная система управления транспортом, средствами на базе фреймворков XAK и EG

Специальность 5В070400 - ВТиПО

Выполнил (а) Тамгенов Д.Е. ПО-12-2
(Фамилия и инициалы) группа

Научный руководитель Купалбаев З.К.
(Фамилия и инициалы, ученая степень, звание)

Консультанты:

по экономической части:
Бекмухаева А.И., к.э.н., доцент
(Фамилия и инициалы, ученая степень, звание)
_____ «02» 06 2016 г.
(подпись)

по безопасности жизнедеятельности:
Приходько Н.Г., д.х.н., проф.
(Фамилия и инициалы, ученая степень, звание)
_____ «25» 05 2016 г.
(подпись)

по применению вычислительной техники:
Купалбаев З.К.
(Фамилия и инициалы, ученая степень, звание)
_____ «03» 06 2016 г.
(подпись)

Нормоконтролер: Купалбаев З.К.
(Фамилия и инициалы, ученая степень, звание)
_____ «03» 06 2016 г.
(подпись)

Рецензент: Баймуканова А.А., проф., д.ф.-м. наук
(Фамилия и инициалы, ученая степень, звание)
_____ «02» 06 2016 г.
(подпись)

Алматы 2016 г.

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН

Некоммерческое акционерное общество
АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ

Факультет Аэрокосмических и информационных технологий
Специальность Вычислительная техника и программное обеспечение
Кафедра Компьютерных технологий

ЗАДАНИЕ
на выполнение дипломного проекта

Студент Тамгенов Диляра Ерболатовна
(фамилия, имя, отчество)

Тема проекта Кроссплатформенная система управления транспортом, средствами на базе фреймворков ХАИ и ЕК

утверждена приказом ректора № 148 от «19» октября 2015 г.
Срок сдачи законченной работы «___» _____ 20___ г.
Исходные данные к проекту требуемые параметры результатов проектирования (исследования) и исходные данные объекта
Система управления транспортом

Перечень подлежащих разработке дипломного проекта вопросов или краткое содержание дипломного проекта:
1. Общее описание
2. Выбор среды реализации
3. Разработка программного обеспечения
4. Безопасность жизнедеятельности
5. Технико - экономическое обоснование

Перечень графического материала (с точным указанием обязательных чертежей)

Рекомендуемая основная литература

Консультанты по проекту с указанием относящихся к ним разделов

Раздел	Консультант	Сроки	Подпись
БЖД	Приходько Н.Г.	18.03-05.05.16	
Экон. часть	Векленко А.И.	27.05-02.06.16	
Общее описание	Курмалбаев З.К.	21.02.16	
Выбор среды реализации	Жунусов М.	07.03.16	
Разработка ПО	Курмалбаев З.К.	03.05.16	
Нормоконтроль	Курмалбаев З.К.	03.06.16	

ГРАФИК
подготовки дипломного проекта

№ п/п	Наименование разделов, перечень разрабатываемых вопросов	Сроки представления руководителю	Примечание
1	Общее описание	21.02.16	
	Деятельность предприятия		
	Постановка задач		
2	Выбор средств реализации	07.03.16	
	Инструментов и технологий		
	для разработки		
	фреймворка ХАП и ЕП		
3.	Разработка программного	03.05.16	
	обеспечения		
	ER - моделирование		
	описание интерфейса		
	программного обеспечения		
4	Технико - экономическое	02.06.16	
	обоснование		
5	Безопасность жизнедеятель-	25.05.16	
	ность		

Дата выдачи задания «14» сентября 2015 г.

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

Руководитель _____ Купалбаев З.К.
(подпись) (Фамилия и инициалы)

Задание принял к исполнению
студент _____ Тлиенов Д.Е.
(подпись) (Фамилия и инициалы)

Андатпа

Диплом жобасы «Эйр Астана». компаниясының транспорт бөлімі үшін Windows-қосымша дайындауға арналған.

Жобада транспортты басқаруда ресурстарды тиімді пайдалану мен жұмыс сапасын арттыруға арналған басқару жүйесі құрастырылған.Сонымен қатар, транспортты басқаруды жеңілдетіп, оның қызмет жасау мүмкіншіліктерін кеңейтеді. Барлық деректерге жылдам қол жетімді. Отынның жұмсалуын талдау және есеп беру келтірілген. Деректерге қол жеткізу үшін қауіпсіздік жүйесі құралған.

Сонымен қатар шығындардың техникалық – экономикалық есептеулері жобада келтірілген және құрастырушы үшін зиянды факторларға талдау жасалынған.

Аннотация

Дипломный проект посвящен разработке Windows-приложения для транспортного отдела компании «Эйр Астана».

В дипломном проекте разработана система управления транспортом для эффективности использования ресурсов и качества работы. Также упростило управления транспортом и расширило ее функциональные возможности. Быстрый доступ ко всем данным. Формирование отчетов и анализ расхода топлива. Также была создана система безопасности для доступа данных.

Также в проекте произведен расчет технико-экономических расходов, и анализ вредных факторов для разработчика.

Annotation

The diploma project is dedicated to the development of Windows-applications for the transport department of "Air Astana".

The diploma project is developed transport management system for efficient use of resources and quality of work. Also, simplify traffic management and expanded its functionality. Quick access to all data. Reporting and analysis of fuel flow. In addition, the security system has been created for data access.

Also in the project calculated the costs of feasibility and analysis of hazards for the developer.

Содержание

Введение

Система управления транспортом поможет вам автоматизировать весь процесс управления перевозками экипажа и пассажиров. Ваш центр может находиться в одном городе, а погрузку производить в другом городе. Но так, как база будет единой, то и учет в ней будет доступен пользователем с доступом на все города. Мониторинг транспорта поможет держать вас в курсе всех событий.

Система управления транспортом является подмножеством управления цепями поставок относительно транспортных операций и может быть частью системы планирования ресурсов предприятия.

Целью данного дипломного проекта, является разработка системы управления транспортом, необходимо:

- обеспечить повышение эффективности использования ресурсов и качества работы
- упростить систему управления транспортными средствами и расширениями функциональных ее возможностей
- Оптимизация и автоматизация системы управления транспортными средствами
- Повышение адаптивности и производительности процессов транспортировки
- Минимизация ошибок и сбоев за счет устранения «человеческого фактора»
- Система значительно облегчит работу всем пользователям

Актуальность темы. Современная логистика не может существовать без активного использования информационных технологий. Практически невозможно представить организацию цепи поставки грузов без постоянного обмена информацией.

Система управления транспортом имеет четыре ключевых процесса управления транспортом:

- Планирование и принятие решений – определяет самые эффективные транспортные схемы в соответствии с заданными параметрам, которые имеют более низкую или более высокую важность согласно пользовательской политике.
- Выполнения транспортировки - позволит обеспечить выполнение плана перевозок
- Транспортная последующая обработка и анализ - позволят после любой физической или административной деятельности относительно транспортировки: прослеживать статус транспорта, редактирования приема, выставления счет-фактур, отправка транспортных оповещений (задержка, авария, остановки и т.д.)
- Измерение – имеет функцию создания отчетов ключевого показателя эффективности (КПД) логистики для транспорта.

Различные функции системы включают в себя:

- Планирование и оптимизация транспортных туров
- В режиме реального времени отслеживать перевозки
- Контроль качества обслуживания в виде ключевых показателей эффективности
- Нагрузка автомобиля
- Транспортные расходы
- Контроль затрат, KPI (ключевые показатели эффективности) отчетность и статистика

1 Общее описание

1.1 Деятельность предприятия

АО «Эйр Астана» — национальный авиаперевозчик Казахстана, крупнейшая авиакомпания страны, выполняющая внутренние и международные регулярные рейсы из аэропортов «Алматы» и «Астана».
Компания является совместным предприятием Фонда национального благосостояния «Самрук-Қазына» (51 %) и британской компании BAE Systems PLC (49 %), образованным в октябре 2001 года. 15 мая 2002 года выполнила свой первый рейс направлением Алма-Ата — Астана.

«Эйр Астана» является официальным перевозчиком EXPO-2017.

Air Astana является единственной авиакомпанией Казахстана, имеющей допуск для полётов в страны Евросоюза.

В 2015 году Air Astana в четвёртый раз подряд была названа «Лучшей авиакомпанией Центральной Азии и Индии» по версии агентства Skytrax. Компания также в третий раз удостоилась награды за «Лучшее бортовое обслуживание в Центральной Азии и Индии». Таким образом авиакомпания является первой и единственной среди авиакомпаний СНГ и Восточной Европы, удостоенной четырьмя наградами агентства Skytrax.

- индивидуальный идентификатор ИАТА (Международная ассоциация воздушного транспорта): КС
- индивидуальный идентификатор ИКАО (Международная организация гражданской авиации): KZR

На сегодняшний день парк воздушных судов авиакомпании «Air Astana» самый молодой в Европе и состоит из 30 (по состоянию на июль 2015 года.) самолетов западного производства. В результате реструктуризации парка и замены всех Airbus A320 и Boeing 767, средний возраст флота «Air Astana» снизился до 6 лет по состоянию на 2015 год. Компания планирует расширить свой флот до 34 самолетов к 2016 году. и до 43 в 2020 году. Кроме того, перевозчик планирует начать полеты в США с приходом Boeing 787 Dreamliner.

Маршрутная сеть включает в себя 64 направлений «Эйр Астаны» включают большинство крупных городов Казахстана и увеличивающееся число соседних центральноазиатских и российских городов. Последнее является результатом реализации так называемой «стратегии суррогатного внутреннего рынка» для улучшения репутации компании и при соблюдении высоких стандартов безопасности в регионе с развивающемся авиарынком. С 2009 года перевозчик открыл рейсы по новым направлениям: Баку, Ташкент, Урумчи, Тбилиси, Душанбе, Бишкек, Новосибирск, Самара, Екатеринбург и Санкт-Петербург и с середины 2012 г. — Казань и Омск. Маршруты Алма-Ата — Киев и Астана — Киев были открыты весной 2013 года. Но в июле 2014 года, в связи с падением рейса 17 Malaysia Airlines,

авиакомпания отменила рейсы Астана — Киев и уменьшила частоту рейсов на рейсах Алма-Ата — Киев.

Долгосрочный рост компании направлен на юг и восток Азии с полетами в Дели, Сеул (осуществляющийся совместно с Asiana Airlines по код-шер соглашению), Пекин, Бангкок, Куала-Лумпур, Гонконг (28 августа 2012 г.) и Хошимин (январь 2013 г.). «Air Astana» также осуществляет полеты из Астаны во Франкфурт ежедневно, и в Хитроу 3 раза в неделю. С 29 марта 2015 года запущены рейсы по маршруту Астана — Париж трижды в неделю. С 1 июня 2015 года авиакомпания представляет сезонные рейсы по маршруту Астана — Тбилиси.

Сервис и брендинг

На 41-й ежегодной церемонии вручения премии Annual Airline Industry Achievement Awards за достижения в авиаиндустрии, организованной изданием Airline Transport World (ATW), которая состоялась в Вашингтоне (округ Колумбия) 25 февраля 2015 года, компания «Air Astana» была удостоена звания «Лидер авиационного рынка».

В 2014 году «Air Astana» третий раз подряд была признана «Лучшей авиакомпанией в Центральной Азии и Индии» в рамках премии Skytrax World Airline Awards. На этой же церемонии «Air Astana» во второй раз получила премию в номинации «Лучше обслуживание в Центральной Азии и Индии». Кроме того, по итогам аудита, проведенного агентством Skytrax в мае 2012 года, компания стала первым авиаперевозчиком в России / СНГ / Восточной Европе, удостоившимся престижного рейтинга «4 звезды» на церемонии Skytrax World Airline Awards 2012 года, и остается единственным авиаперевозчиком в регионе, имеющим этот статус. В июне 2015 года на церемония награждения премии Skytrax World Airline Awards 2015 «Air Astana» удостоилась рейтинга в 4 звезды, а также победы в категориях «Лучшая авиакомпания Центральной Азии и Индии» в четвёртый раз подряд, и «Лучшее бортовое обслуживание в Центральной Азии и Индии» в третий раз.

Награды:

- рейтинг 4-х звездочной авиакомпании за качество обслуживания по оценкам компании Skytrax. «Air Astana» — единственная 4-х звездочная авиакомпания в Восточной Европе и СНГ;
- премия «Авиакомпании мира» в 2012, 2013 и 2014 гг. в номинации «Лучшая авиакомпания в Центральной Азии и Индии»;
- премия «Авиакомпании мира» в 2013 и 2014 гг. в номинации «Лучшее обслуживание в Центральной Азии и Индии»;
- «Лидер авиационного рынка 2015» по версии издания Air Transport World;

1.2 Постановка задачи

Постановка задачи — точная формулировка условий задачи с описанием входной и выходной информации

Входная информация по задаче — данные, поступающие на вход задачи и используемые для её решения.

Выходная информация может быть представлена в виде документов, кадров на экране монитора, информации в базе данных, выходного сигнала устройству управления.

Постановка задачи разрабатывается организацией, разработчиком программной продукции, на основании технического задания совместно с заказчиком. Главный исполнитель — это разработчик.

Задачи:

- Ознакомление с фреймворками XAF и EF
- Анализ текущего положения
- Составление функциональной спецификации
- Составление требований к программному обеспечению
- Проектирование модели базы данных
- Разработка Windows-приложения

Требования к программному обеспечению — совокупность утверждений относительно атрибутов, свойств или качеств программной системы, подлежащей реализации.

Требования:

- Ведение учета о нарядах
- Формирование отчетов и графиков
- Ведение статистики транспортного средства
- Быстрый и оптимальный поиск нужных данных
- Система экспорта
- Оповещения пользователей
- Локализация системы
- Система безопасности

2 Выбор сред реализации

2.1 Инструменты и технологии для разработки

2.1.1 Язык программирования C#

C# — элегантный, типобезопасный объектно-ориентированный язык, предназначенный для разработки разнообразных безопасных и мощных приложений, выполняемых в среде .NET Framework. С помощью языка C# можно создавать обычные приложения Windows, XML-веб-службы, распределенные компоненты, приложения "клиент-сервер", приложения базы данных и т. д. Visual C# предоставляет развитый редактор кода, конструкторы с удобным пользовательским интерфейсом, встроенный отладчик и множество других средств, упрощающих разработку приложений на базе языка C# и .NET Framework.

На сегодняшний момент язык программирования C# одни из самых мощных, быстро развивающихся и востребованных языков в ИТ-отрасли. В настоящий момент на нем пишутся самые различные приложения: от небольших десктопных программок до крупных веб-порталов и веб-сервисов, обслуживающих ежедневно миллионы пользователей.

По сравнению с другими языками C# достаточно молодой, но в то же время он уже прошел большой путь. Первая версия языка вышла вместе с релизом MS Visual Studio .NET в феврале 2002 года. Текущей версией языка является версия C# 6.0, которая вышла в 20 июля 2015 года вместе с Visual Studio 2015.

C# является языком с Си-подобным синтаксисом и близок в этом отношении к C++ и Java. Поэтому, если вы знакомы с одним из этих языков, то овладеть C# будет легче.

C# является объектно-ориентированным и в этом плане много перенял у Java и C++. Например, C# поддерживает полиморфизм, наследование, перегрузку операторов, статическую типизацию. Объектно-ориентированный подход позволяет решить задачи по построению крупных, но в тоже время гибких, масштабируемых и расширяемых приложений. И C# продолжает активно развиваться, и с каждой новой версией появляется все больше интересной функциональности, как, например, лямбды, динамическое связывание, асинхронные методы и т.д.

В дополнение к основным описанным объектно-ориентированным принципам, язык C# упрощает разработку компонентов программного обеспечения благодаря нескольким инновационным конструкциям языка, в число которых входят следующие:

- Инкапсулированные сигнатуры методов, называемые делегатами, которые поддерживают типобезопасные уведомления о событиях.

- Свойства, выступающие в роли методов доступа для закрытых переменных-членов.

- Атрибуты с декларативными метаданными о типах во время выполнения.

- Встроенные комментарии XML-документации.

- LINQ, предлагающий встроенные возможности запросов в различных источниках данных.

Роль платформы .NET

Когда говорят C#, нередко имеют в виду технологии платформы .NET (WPF, ASP.NET). И, наоборот, когда говорят .NET, нередко имеют в виду C#. Однако, хотя эти понятия связаны, отождествлять их неверно. Язык C# был создан специально для работы с фреймворком .NET, однако само понятие .NET несколько шире.

Как-то Билл Гейтс сказал, что платформа .NET - это лучшее, что создала компания MS. Возможно, он был прав. Фреймворк .NET представляет мощную платформу для создания приложений. Можно выделить следующие ее основные черты:

- Поддержка нескольких языков. Основой платформы является общеязыковая среда исполнения Common Language Runtime (CLR), благодаря чему .NET поддерживает несколько языков: наряду с C# это также VB.NET, C++, F#, а также различные диалекты других языков, привязанные к .NET, например, Delphi.NET. При компиляции код на любом из этих языков компилируется в сборку на общем языке CIL (Common Intermediate Language) - своего рода ассемблер платформы .NET. Поэтому мы можем сделать отдельные модули одного приложения на отдельных языках.

- Кроссплатформенность. .NET является переносимой платформой (с некоторыми ограничениями). Например, последняя версия платформы на данный момент .NET Framework поддерживается на большинстве современных ОС Windows (Windows 10/8.1/8/7/Vista). А благодаря проекту Mono можно создавать приложения, которые будут работать и на других ОС семейства Linux, в том числе на мобильных платформах Android и iOS.

- Мощная библиотека классов. .NET представляет единую для всех поддерживаемых языков библиотеку классов. И какое бы приложение мы не собирались писать на C# - текстовый редактор, чат или сложный веб-сайт - так или иначе мы задействуем библиотеку классов .NET.

- Разнообразие технологий. Общеязыковая среда исполнения CLR и базовая библиотека классов являются основой для целого стека технологий, которые разработчики могут задействовать при построении тех или иных приложений. Например, для работы с базами данных в этом стеке технологий предназначена технология ADO.NET. Для построения графических приложений с богатым насыщенным интерфейсом - технология WPF. Для создания веб-сайтов - ASP.NET и т.д.

Также еще следует отметить такую особенность языка C# и фреймворка .NET, как автоматическая сборка мусора. А это значит, что нам в большинстве случаев не придется, в отличие от C++, заботиться об освобождении памяти. Вышеупомянутая общеязыковая среда CLR сама вызовет сборщик мусора и очистит память.

Управляемый и неуправляемый код

Нередко приложение, созданное на C#, называют управляемым кодом (managed code). Что это значит? А это значит, что данное приложение создано на основе платформы .NET и поэтому управляется общеязыковой средой CLR, которая загружает приложение и при необходимости очищает память. Но есть также приложения, например, созданные на языке C++, которые компилируются не в общий язык CIL, как C# или VB.NET, а в обычный машинный код. В этом случае .NET не управляет приложением.

В то же время платформа .NET предоставляет возможности для взаимодействия с неуправляемым кодом. Мы наряду со стандартными классами библиотеки .NET можем также использовать сборки COM.

Исходный код, написанный на языке C#, компилируется в промежуточный язык (IL) в соответствии со спецификацией CLI. Код IL и ресурсы, такие как растровые изображения и строки, хранятся на диске в исполняемом файле, называемом сборкой, с расширением EXE или DLL в большинстве случаев. Сборка содержит манифест со сведениями о типах сборки, версии, языке и региональных параметрах, и требованиях безопасности.

При выполнении программы на C# сборка загружается в среду CLR в зависимости от сведений в манифесте. Далее, если требования безопасности соблюдены, среда CLR выполняет JIT-компиляцию для преобразования кода IL в инструкции машинного кода. Среда CLR также предоставляет другие службы, относящиеся к автоматическому сбору мусора, обработке исключений и управлению ресурсами. Код, выполняемый средой CLR, иногда называют "управляемым кодом" в противопоставление "неуправляемому коду", который компилируется в машинный код, предназначенный для определенной системы. Далее показаны отношения во время компиляции и время выполнения между файлами с исходным кодом C#, библиотеками классов .NET Framework, сборками и средой CLR.

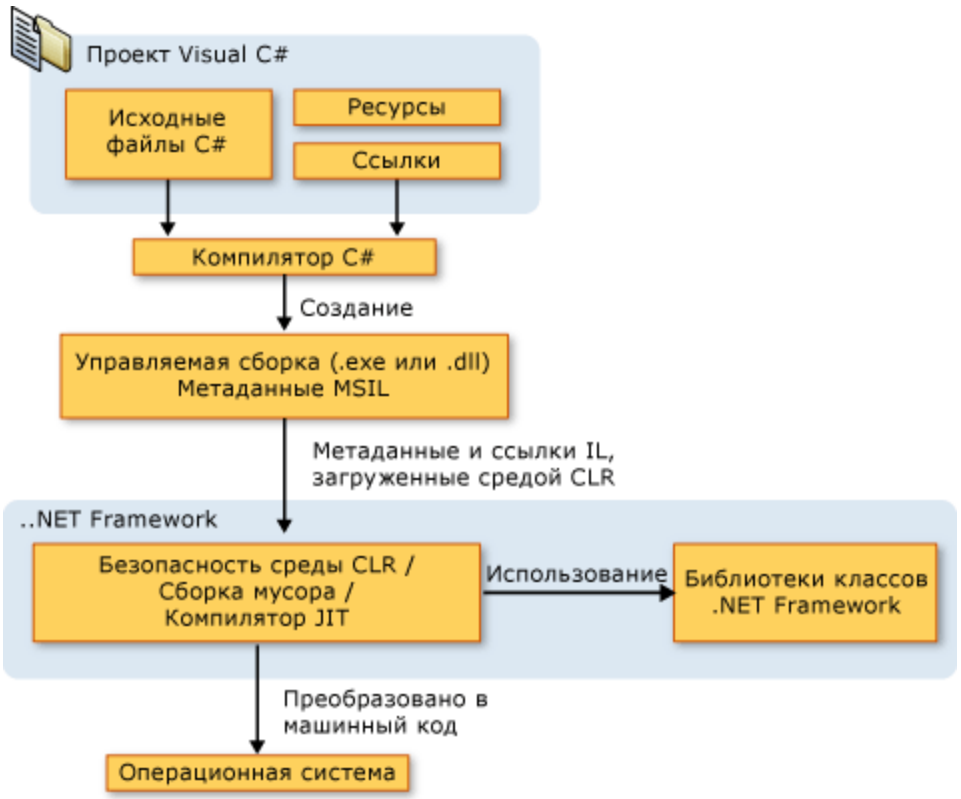


Рисунок 2.1 – Сборка

Взаимодействие между языками является ключевой особенностью .NET Framework.Поскольку код на промежуточном языке (IL), создаваемый компилятором C#, соответствует спецификации CTS, код IL на основе C# может взаимодействовать с кодом, создаваемым версиями языков Visual Basic, Visual C++, Visual J# платформы .NET Framework и еще более чем 20 CTS-совместимых языков. В одной сборке может быть несколько модулей, написанных на разных языках платформы .NET Framework, и типы могут ссылаться друг на друга, как если бы они были написаны на одном языке.

Помимо служб времени выполнения, в .NET Framework также имеется обширная библиотека, состоящая из более чем 4000 классов, организованных по пространствам имен, которые обеспечивают разнообразные полезные функции для любых действий, начиная от ввода и вывода файлов для управлением строками для разбивки XML, и заканчивая элементами управления Windows Forms. В обычном приложении на языке C# библиотека классов .NET Framework интенсивно используется для "устройства" кода.

JIT-компиляция

Как выше писалось, код на C# компилируется в приложения или сборки с расширениями exe или dll на языке CIL. Далее при запуске на выполнение подобного приложения происходит JIT-компиляция (Just-In-Time) в машинный код, который затем выполняется. При этом, поскольку наше приложение может быть большим и содержать кучу инструкций, в текущий момент времени будет компилироваться лишь та часть приложения, к которой непосредственно идет

обращение. Если мы обратимся к другой части кода, то она будет скомпилирована из CIL в машинный код. При том уже скомпилированная часть приложения сохраняется до завершения работы программы. В итоге это повышает производительность.

По сути это все, что вкратце надо знать о платформе .NET. А теперь создадим первое приложение.

2.1.2 MS Visual Studio 2015

MS Visual Studio — линейка продуктов компании MS, включающих интегрированную среду разработки программного обеспечения и ряд других инструментальных средств.

Таблица 2.1 – История релизов

Официальное название	Кодовое название	Внутренняя версия	Версии .NET Framework	Дата выхода
Visual Studio	–	4.0	–	Апрель 1995
Visual Studio 97	Boston	5.0	–	Февраль 1997
Visual Studio 6.0	Aspen	6.0	–	Июнь 1998
Visual Studio .NET (2002)	Rainier	7.0	1.0	2002-02-13
Visual Studio .NET 2003	Everett	7.1	1.1	2003-04-24
Visual Studio 2005	Whidbey	8.0	2.0, 3.0	2005-11-07
Visual Studio 2008	Orcas	9.0	2.0, 3.0, 3.5	2007-11-19
Visual Studio 2010	Dev10	10.0	2.0, 3.0, 3.5, 4.0	2010-04-12
Visual Studio 2012	Dev11	11.0	2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2	2012-08-15
Visual Studio 2013	Dev12	12.0	2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2	2013-10-17
Visual Studio 2015	Dev14	14.0	2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6	2015-07-20

Используя Visual Studio, можно создавать приложения для устройств Android, iOS и Windows.

Выберите нужные инструменты с учетом требований ваших приложений и языка, который вы хотите использовать.

Инструменты:

- Xamarin для Visual Studio: общая база кода C# для всех устройств;

- Apache Cordova с Visual Studio: общая база кода для HTML, CSS и JavaScript или Typescript;
- Средства Visual Studio Tools для Unity: разработка игр 2D/3D в C#;
- C++ для кроссплатформенной разработки: общие библиотеки кода и приложения в C++;
- Эмулятор Visual Studio для Android: эмулятор Visual Studio для Android – отладка и тестирование приложений Android независимо от интегрированной среды разработки.

Возможность подключения службам без дополнительных настроек (при условии наличия учетной записи для каждой из служб).

- Мобильные службы Azure;
- Хранилище Azure;
- Интеграция API-интерфейсов REST Office 365 в приложения для подключения к данным, хранящимся в облаке.

Создание, тестирование кода и управление им

Возврат кода. При труде в команде нужно снабдить размен важными версиями кода, чтоб все соучастники рабочей категории действовали с одной и той ведь основанием кода. Visual Studio просто интегрируется с Visual Studio Team Services либо Team Foundation Server, обеспечивая вероятность управления версиями с внедрением Git либо системы управления версиями Team Foundation (TFVC). Применяйте Visual Studio Team Services для сохранения кода в туче в отсутствии надобности сервиса локального сервера. Пользуйтесь Team Foundation Server, ежели вам необходим местный сервер. Наличествует вероятность прибавления всех творимых планов кода в систему управления версиями.

Возведение прибавления. Разрешено создать прибавление локально и удостовериться, будто оно верно действует. Потом воспользуйтесьсредствами отладки, чтоб поправить оплошности в прибавлении. Не считая такого, разрешено творить планы на единых серверах производства либо в туче. Автоматизируйте функцию производства, чтоб снабдить творение кода, кой создатели вашей рабочей категории сумеют возвращать в систему управления версиями. К примеру, разрешено творить Вотан либо наиболее планов любую ночь либо любой раз при возврате кода.

Испытание прибавления. Сотворите модульные испытания, чтоб нарастить свойство собственных прибавлений. Данные испытания разрешено делать любой раз при возврате либо в процессе производства. Намеревайтесь, творите и исполняйте автоматические и неавтоматические испытания для вашего прибавления. Намерения испытания, комплекты, тестовые случаи и итоги исследований сберегаются в туче (при применении Visual Studio Team Services) либо локально (при применении Team Foundation Server).

2.1.3 MS SQL Server 2012

SQL Server считается основой платформы этих компании Майкрософт и дает верную и стабильную продуктивность (в том количестве спасибо технологиям отделки этих в памяти) и подсобляет скорее вытянуть значимую информацию из всех этих, находящихся как в локальной среде, этак и в туче.

Наверное – техно книгохранилище сообразно SQL Server, в которой представлена информация о крайних версиях SQL Server, а еще и о наиболее ранешних. Крупная дробь инфы, связанной с крайними версиями SQL Server, еще применима к службам базы этих SQL и хранилища этих SQL в Azure.

Сервер базы этих MS SQL Server в качестве языка запросов употребляет версию языка SQL, возымевшую заглавие Transact-SQL (скупо T-SQL). Язычок T-SQL считается реализацией SQL-92 (эталон ISO для языка SQL) с многочисленными расширениями. T-SQL дозволяет применять доп синтаксис для хранимых операцй и гарантирует помощь транзакций (взаимодействие базы этих с правящим прибавлением).

При содействии с сетью MS SQL Server и Sybase ASE употребляют протокол значения прибавления перед заглавием Tabular Data Stream (TDS, протокол передачи табличных этих). Протокол TDS еще был продан в плане FreeTDS с целью снабдить разным прибавлениям вероятность взаимодействия с базами этих MS SQL Server и Sybase.

Для снабжения доступа к этим MS SQL Server поддерживает Open Database Connectivity (ODBC) — интерфейс взаимодействия прибавлений с СУБД. Версия SQL Server 2005 гарантирует вероятность включения юзеров чрез интернет-сервисы, использующие протокол SOAP. Наверное дозволяет клиентским програмкам, никак не уготованным для Windows, кроссплатформенно объединяться с SQL Server. Фирма MS еще выпустила сертифицированный драйвер JDBC, дозволяющий прибавлениям перед управлением Java (таковым как BEA и IBM WebSphere) объединяться с MS SQL Server 2000 и 2005.

Еще SQL Server поддерживает зеркалирование и кластеризацию баз этих. Кластер сервера SQL — наверное совокупа идиентично конфигурированных серверов; таковая методика подсобляет разделить рабочую нагрузку меж несколькими серверами. Все сервера имеют одно виртуальное фамилия, и эти распределяются сообразно IP-адресам автомашин кластера в движение рабочего цикла. Еще в случае отказа либо перебоя на одном из серверов кластера доступен самодействующий перенесение перегрузки на иной сервер.
SQL Server поддерживает лишнее резервирование этих сообразно 3 сценариям:

- Снимок: Делается “снимок” базы этих, кой сервер посылает получателям.

- Деяния конфигураций: Все конфигурации базы этих постоянно передаются юзерам.

- Синхронизация с иными серверами: Базы этих нескольких серверов синхронизируются меж собой. Конфигурации всех баз этих проистекают самостоятельно приятель от приятеля на любом сервере, а при синхронизации проистекает сверка этих. Этот вид дублирования предугадывает вероятность разрешения противоречий меж БД.

В SQL Server 2005 встроена помощь .NET Framework. Спасибо данному хранимые упражнения БД имеют все шансы существовать прописаны на всяком языке платформы .NET, применяя целый комплект библиотек, легкодоступных для .NET Framework, подключая Common Type System (система обращения с типами этих в MS .NET Framework). Но, в различие от остальных действий, .NET Framework, будучи базовой системой для SQL Server 2005, выделяет доп память и выстраивает средства управления SQL Server заместо такого, чтоб применять интегрированные средства Windows. Наверное увеличивает продуктивность в сопоставлении с едиными методами Windows, этак как методы распределения ресурсов умышленно настроены для применения в текстурах SQL Server.

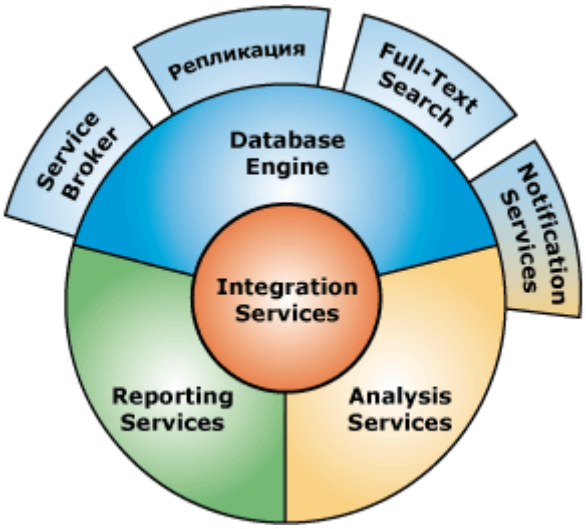
Электронная документация охватывает отображение задач и справочную информацию, в каком месте рассказывается о том, как исполнять управление данными и действовать со средствами бизнес-специалисты с поддержкою SQL Server.

MS SQL Server дает собой платформу для работы с базами этих, обеспечивающую вероятность крупномасштабной своевременной отделки транзакций (OLTP), сохранения этих и работы с прибавлениями для электронной торговли; а еще считается платформой бизнес-специалисты для сотворения решений сообразно интеграции этих, разбору и составлению докладов.

Составляющие MS SQL Server:

- Составляющую Database Engine
- Службы Analysis Services
- Службы Integration Services
- Составляющую репликации
- Службы Reporting Services
- Службы Notification Services
- Составляющую Full-Text Search
- Составляющую Service Broker

На рисунке ниже представлены взаимосвязи и взаимодействия меж составляющими MS SQL Server.



Риунок 2.2 – Взаимодействия компонентов MS SQL Server

2.1.4 ADO.NET Entity Framework

Entity Framework (EF) — это объектно-реляционный модуль сопоставления, позволяющий разработчикам .NET работать с реляционными данными с помощью объектов, специализированных для доменов. Это устраняет необходимость в написания большей части кода для доступа к данным, который обычно требуется разработчикам.

Преимущества

- Приложения могут работать концептуальной моделью в терминах предметной области — в том числе с наследуемыми типами, сложными элементами и связями.
- Приложения освобождаются от жестких зависимостей от конкретного ядра СУБД или схемы хранения.
- Сопоставления между концептуальной моделью и схемой, специфичной для конкретного хранилища, могут меняться без изменения кода приложения.
- Разработчики имеют возможность работать с согласованной моделью объектов приложения, которая может быть сопоставлена с различными схемами хранения, которые, возможно, реализованы в различных системах управления данными.
- Несколько концептуальных моделей могут быть сопоставлены с единой схемой хранения.
- Поддержка запросов LINQ обеспечивает проверку синтаксиса во время компиляции для запросов к концептуальной модели.

Центральной концепцией Entity Framework является понятие сущности или entity. Сущность представляет набор данных, ассоциированных с определенным объектом. Поэтому данная технология предполагает работу не с таблицами, а с объектами и их наборами.

Любая сущность, как и любой объект из реального мира, обладает рядом свойств. Например, если сущность описывает человека, то мы можем выделить такие свойства, как имя, фамилия, рост, возраст, вес. Свойства необязательно представляют простые данные типа int, но и могут представлять более комплексные структуры данных. И у каждой сущности может быть одно или несколько свойств, которые будут отличать эту сущность от других и будут уникально определять эту сущность. Подобные свойства называют ключами.

При этом сущности могут быть связаны ассоциативной связью один-ко-многим, один-ко-одному и многие-ко-многим, подобно тому, как в реальной базе данных происходит связь через внешние ключи.

Отличительной чертой Entity Framework является использование запросов LINQ для выборки данных из БД. С помощью LINQ мы можем не только извлекать определенные строки, хранящие объекты, из бд, но и получать объекты, связанные различными ассоциативными связями.

Другим ключевым понятием является Entity Data Model. Эта модель сопоставляет классы сущностей с реальными таблицами в БД.

Entity Data Model состоит из трех уровней: концептуального, уровень хранилища и уровень сопоставления (маппинга).

На концептуальном уровне происходит определение классов сущностей, используемых в приложении.

Уровень хранилища определяет таблицы, столбцы, отношения между таблицами и типы данных, с которыми сопоставляется используемая база данных.

Уровень сопоставления (маппинга) служит посредником между предыдущими двумя, определяя сопоставление между свойствами класса сущности и столбцами таблиц.

Таким образом, мы можем через классы, определенные в приложении, взаимодействовать с таблицами из базы данных.

Entity Framework предполагает три возможных способа взаимодействия с базой данных:

- Database first: Entity Framework создает набор классов, которые отражают модель конкретной базы данных

- Model first: сначала разработчик создает модель базы данных, по которой затем Entity Framework создает реальную базу данных на сервере.

- Code first: разработчик создает класс модели данных, которые будут храниться в бд, а затем Entity Framework по этой модели генерирует базу данных и ее таблицы

Применение моделей на практике

Многолетним и общим подходом к разработке является подход, при котором построение приложения или службы представляет собой его разделение на три части: модель домена, логическую модель и физическую модель. Модель домена определяет сущности и связи в моделируемой системе. Логическая модель для реляционной базы данных обеспечивает

нормализацию сущностей и связей в целях создания таблиц с ограничениями внешнего ключа. В физической модели учитываются возможности конкретной системы обработки данных путем определения зависящих от ядра базы данных подробных сведений о хранении данных, которые касаются секционирования и индексирования.

Физическая модель совершенствуется администраторами базы данных в целях повышения производительности, но программисты, которые разрабатывают код приложения, в основном вынуждены ограничиваться работой с логической моделью, подготавливая SQL-запросы и вызывая хранимые процедуры. Модели домена в основном используются как инструмент для представления и обмена мнениями о требованиях к приложению, поэтому чаще всего служат в качестве практически не изменяющихся схем, которые рассматриваются и обсуждаются на ранних стадиях проекта, после чего выходят из сферы внимания. Во многих коллективах разработчиков принято пропускать этап создания концептуальной модели и начинать с определения таблиц, столбцов и ключей в реляционной базе данных.

Платформа Entity Framework придает значимость моделям, позволяя разработчикам выполнять запросы к сущностям и связям в модели домена (которая называется концептуальной моделью в Entity Framework), при этом для перевода этих операций в команды, определяемые источником данных, используется сама платформа Entity Framework. Это позволяет отказаться от применения в приложениях жестко заданных зависимостей от конкретного источника данных.

При работе в режиме Code First концептуальная модель сопоставлена с моделью хранения в коде. Entity Framework может вывести концептуальную модель, основанную на типах объектов и дополнительных конфигурациях, которые можно задать. Метаданные сопоставления формируются во время выполнения на основе сочетания определений типов домена и дополнительной информации о конфигурации, которая указана в коде. Entity Framework при необходимости создает базу данных на основе метаданных

При работе со средствами работы с моделью EDM концептуальная модель, модель хранения и сопоставление между ними выражены в схемах на основе XML и определены в файлах с именами с соответствующими расширениями.

- Язык CSDL определяет концептуальную модель. Язык CSDL - это реализация модели EDM для платформы Entity Framework. Расширение файла - CSDL.

- Язык SSDL определяет модель хранения данных, которая также называется логической моделью. Расширение файла - SSDL.

- Язык MSL определяет сопоставление модели хранения и концептуальной модели. Расширение файла - MSL.

Модель хранения и сопоставления при необходимости могут быть изменены без изменения концептуальной модели, классов данных и кода приложения. Модели хранения зависят от поставщика, поэтому можно работать с согласованной концептуальной моделью через различные источники данных.

В Entity Framework файлы модели и сопоставления служат для выполнения операций создания, чтения, обновления и удаления, выполняемых над сущностями и связями концептуальной модели, в эквивалентные операции в источнике данных. Entity Framework поддерживает даже сопоставление сущностей в концептуальной модели с хранимыми процедурами в источнике данных.

Доступ к данным сущностей

В большей степени, чем остальные решения объектно-реляционного сопоставления, Entity Framework направлен на то, чтобы давать приложениям возможность чтения и изменения данных, представленных в виде сущностей и связей в концептуальной модели. Entity Framework использует данные в модели и файлах сопоставления для преобразования запросов объектов к типам сущностей, представленным в концептуальной модели, в запросы, зависящие от источника данных. Результаты запросов преобразуются в объекты, которыми управляют Entity Framework. Платформа Entity Framework реализует следующие способы выполнения запросов к концептуальной модели и возврата объектов.

- LINQ to Entities. Обеспечивает поддержку запросов LINQ для выполнения запросов к типам сущности, которые определены в концептуальной модели.

- Entity SQL. Независимый от хранилища диалект SQL, который работает непосредственно с сущностями в концептуальной модели и поддерживает основные понятия EDM (модель данных с использованием сущностей).Entity SQL используется и с запросами объектов, и с запросами, выполняемыми с помощью поставщика EntityClient.

Платформа Entity Framework включает в себя поставщик данных EntityClient. Поставщик управляет соединениями, переводит запросы сущностей в запросы, зависящие от источника данных, и возвращает модуль чтения данных, который используется Entity Framework для материализации данных сущности в виде объектов. Если материализация объектов не требуется, то поставщик EntityClient может также работать в качестве стандартного поставщика данных ADO.NET, позволяющий приложениям выполнять запросы Entity SQL и получать данные только для чтения, возвращаемые модулем чтения данных.

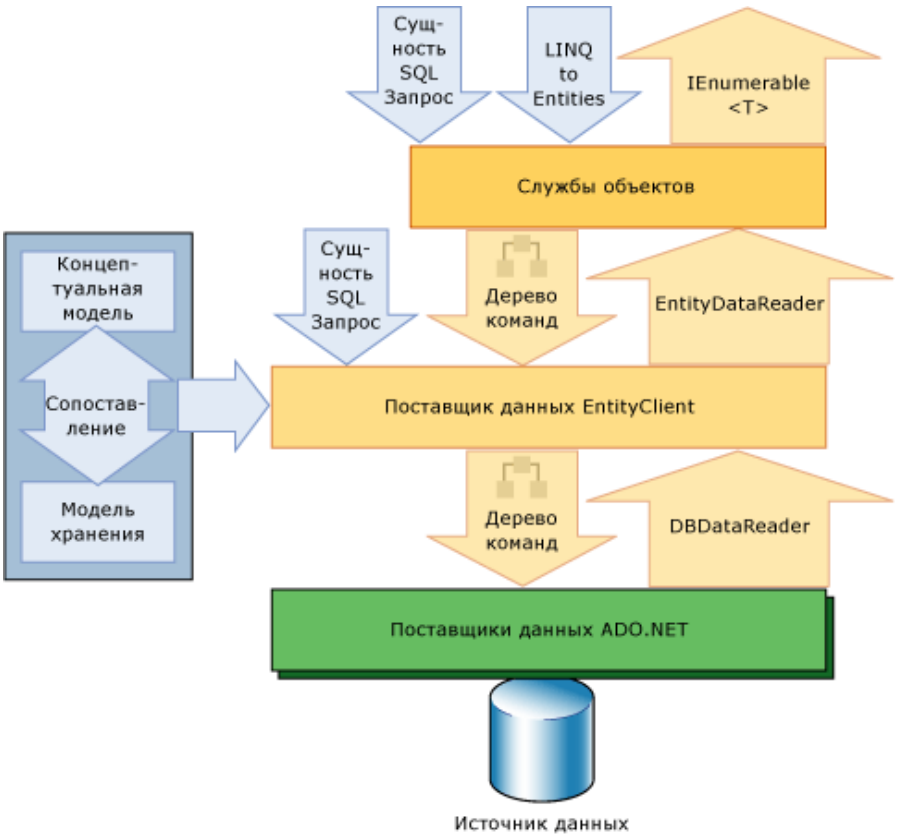


Рисунок 2.2 – архитектура доступа к данным EF

Платформа ADO.NET Entity Framework поддерживает приложения и службы, ориентированные на данные, и обеспечивает программирование с данными, позволяющее поднять уровень абстракции с логического реляционного до концептуального. Предлагая разработчикам возможность работать с данными на более высоком уровне абстракции, платформа Entity Framework поддерживает код, который является независимым от конкретного типа подсистемы хранилища данных или реляционной схемы. Дополнительные сведения см. в разделе Общие сведения о платформе Entity Framework.

Платформа Entity Framework поддерживает модель Entity Data Model (EDM) для определения данных на концептуальном уровне. При использовании конструктора ADO.NET Entity Data Model Designer концептуальная модель, модель хранения и сведения о сопоставлениях содержатся в файле EDMX. Платформа Entity Framework также позволяет разработчикам программировать непосредственно с использованием типов данных, определенных на концептуальном уровне в виде объектов среды CLR. Платформа Entity Framework предоставляет средства для формирования файла EDMX и связанных объектов CLR на основе существующей базы данных. Это значительно сокращает объем кода для доступа к данным, который раньше требовался для создания приложений и служб по работе с данными на базе объектов, а также ускоряет создание объектно-ориентированных приложений и служб по работе с данными на основе существующей базы данных.Эти средства также позволяют создать концептуальную модель в

первую очередь, а затем создать связанные объекты CLR и поддерживающую базу данных. Дополнительные сведения см. в разделе Generate Database Wizard.

Подразделы этого раздела помогают быстро приступить к использованию платформы Entity Framework благодаря разъяснению базовых технологий, упомянутых в учебнике Краткое руководство. В учебнике «Краткое руководство» демонстрируется создание приложения Entity Framework из существующей базы данных.

2.1.5 Devexpress ExpressApp Framework

eXpressApp Framework (XAF) от компании DevExpress представляет собой мощный и гибкий инструмент, для быстрой разработки бизнес-приложений. XAF позволяет разработчику сконцентрироваться на бизнес функциональности приложения, в то время как стандартные функции программы, такие как: права доступа, дизайнер отчетов, стандартный настраиваемый графический интерфейс, поддержка Web интерфейса, сохранение объектов в базу данных, система workflow уже реализованы внутри XAF. Технологически XAF представляет набор .NET библиотек, подключаемых к разрабатываемой программе и реализующих блоки функциональности, свойственные современному качественному программному обеспечению.

Программисту, имеющему в своем арсенале XAF, не надо ломать голову над тем, как сформировать из наших контролов красивый и функциональный пользовательский интерфейс и организовать работу с базой данных. Для начала достаточно написать на C# или VB.NET набор классов или интерфейсов, представляющих объекты, с которыми будет оперировать приложение (например «сотрудник», «клиент», «заказ» и т.п.). Всё остальное XAF берет на себя, предоставляя на выходе WinForms и ASP.NET приложения, в которых уже есть необходимый набор форм для ввода и представления данных.

Конечно, eXpressApp Framework был первоначально разработан, чтобы решить проблемы, которые имеют место при разработке приложения с нуля. Таким образом основные цели eXpressApp Framework:

Легко использовать повторно одну и ту же бизнес-логику в приложениях, предназначенных для различных платформ (Windows Forms и ASP.NET).

Обеспечение алгоритмов конструирования данных к UI. Это означает, что вам не нужно вручную создавать много подобных форм для просмотра и редактирования данных. Это также упрощает, поддерживания приложения, так как если вы измените данные, вы не должны изменять многочисленные формы и/или веб-страницы - это делается автоматически.

Делает чрезвычайно легким создания бизнес-приложения, предназначенные для хранения и представления данных

Позволяет приложению быть созданным с помощью любого языка .NET.

Есть много особенностей, которые делают eXpressApp Framework наилучшим вариантом:

- Легко расширяемый. Вы можете настроить или полностью заменить почти каждый встроенный элемент пользовательского интерфейса приложения или стандартное поведение.

- Качество главных составных частей гарантируется. eXpress Persistent Objects и ADO.NET Entity Framework, используемые в качестве доступа к реляционным данным.

Приложения, созданные с eXpressApp Framework состоят из нескольких функциональных блоков. На приведенном ниже рисунке показаны основные блоки, указывает на то, когда и как эти блоки создаются, и показывает вам те области, где вы можете расширить свое приложение.

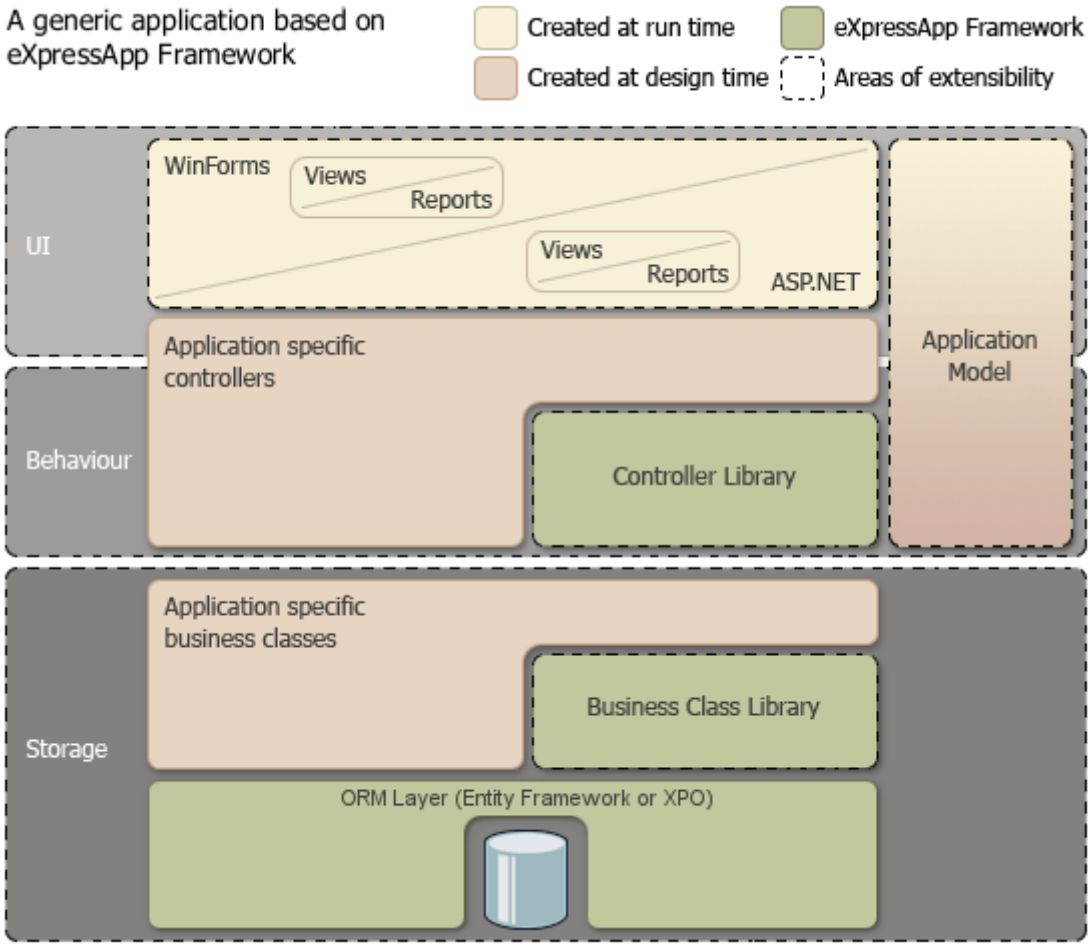


Рисунок 2.3 – архитектура XAF

Хранилище (Storage).

Уровень объектно-реляционного отображения (ORM Layer)

При создании бизнес-приложений, вы будете в конечном итоге иметь дело с данными. Если вы используете eXpressApp Framework, вам нет необходимости фактически создавать базу данных в СУБД, настраивать таблицы, поля и т.д. И вам не нужно будет использовать низкоуровневые конструкции ADO.NET для доступа к данным. Вместо этого вы будете

использовать один из поддерживаемых инструментов ORM - Entity Framework (EF) или eXpressPersistent Objects (XPO).

ORM инструменты позволяют описать данные для вашего приложения, используя знакомые кодовые структуры - классы, свойства и их атрибуты. Чтобы создать таблицу данных, вам необходимо объявить класс. Ее свойства будут определять поля данных в таблице. Конечно, вы можете создать столько таблиц, сколько вам нужно, и указать отношения между ними, используя специально разработанные атрибуты. Обратите внимание, что вам не придется делать много дополнительной работы при создании этих классов. Вам нужно всего лишь вывести их из собственных классов и поставить несколько атрибутов - это все. Чтобы помочь вам начать работу, мы предлагаем библиотеку бизнес-класса (Bussiness Class Library). Эта библиотека содержит несколько классов, готовых к использованию (как для EF и XPO), которые вы можете интегрировать в свои приложения. Вы также можете просмотреть исходный код этих классов для примеров по надлежащей декларации данных.

Так как таблица данных описывается классом, фактические данные представлены коллекцией экземпляров классов. Таким образом, чтобы изменить поле в той или иной записи, вам нужно, получить желаемый объект из коллекции и изменить его свойства. Это гораздо более простой и естественный метод обработки данных. Он скрывает все детали реализации, позволяя вам сконцентрироваться на бизнес-логике приложения.

Библиотека бизнес-класса (Bussiness Class Library)

Библиотке бизнес-класса предлагает вам следующее:

- Классы, которые определяют некоторые часто используемые объекты, такие как персона, заметки, организации и т.д. Вы можете использовать эти классы, или вывести свои собственные классы, если вам нужно расширить или изменить их. Вы также можете просмотреть исходный код этих классов, чтобы узнать, как правильно реализовать свои структуры данных.

- Интерфейсы, которые могут понадобиться для реализации ваших классов данных. Некоторые подсистемами в eXpressApp Framework требует, чтобы данные соответствовали определенным правилам. Например, в области безопасности подсистемы требует пользовательского класса для реализации интерфейса IUser. Итак, если вы решили разработать свой собственный класс для представления пользователей приложения, вы также должны реализовать этот интерфейс.

На рисунке ниже показаны некоторые классы, которые вы найдете в библиотеке бизнес-класса.

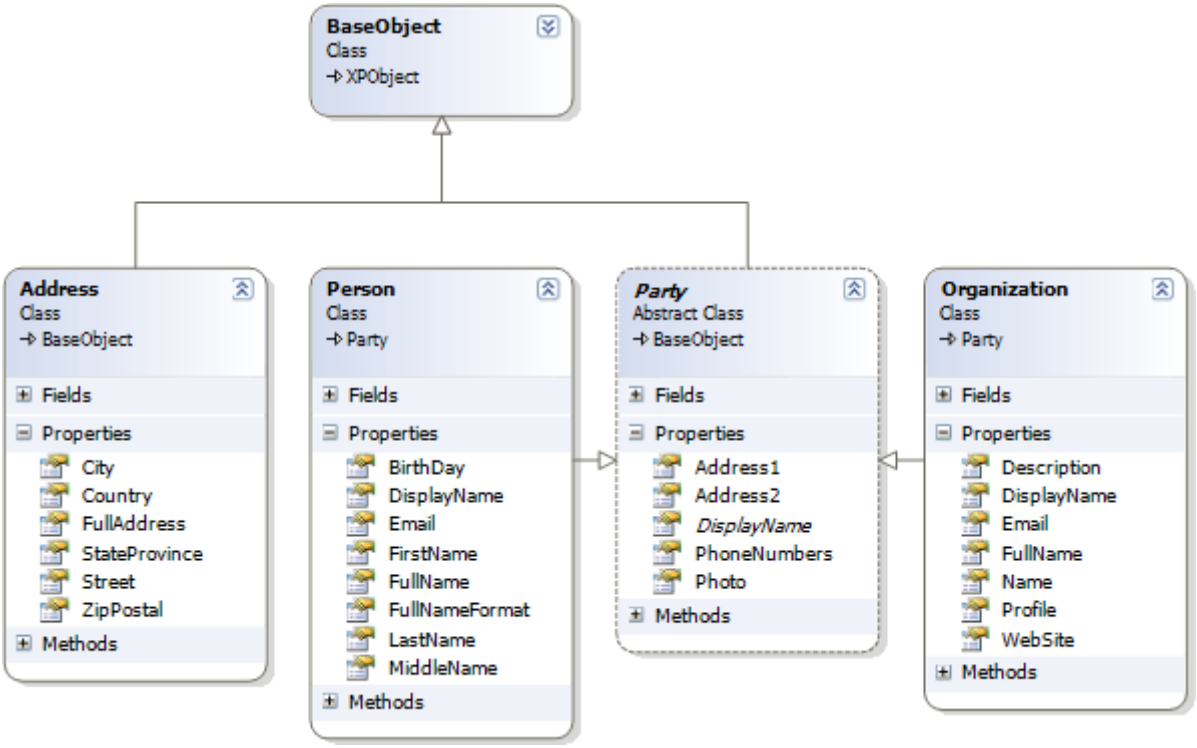


Рисунок 2.4 – Библиотека бизнес-классов

Пользовательский Интерфейс (UI)

Windows-приложения и веб-приложения ASP.NET

Одной из главных целей eXpressApp Framework, отделить бизнес-логику от визуального представления приложения. Это дает возможность создавать Windows-приложения и веб-приложения, основанные на одинаковой логике. При создании нового решения приложений с помощью eXpressApp Framework, решение, которое включает в себя два проекта запуска для рабочего стола и веб.

Представления (Views)

Одной из ключевых особенностей в eXpressApp Framework, является автоматическое создание пользовательского интерфейса на основе данных приложения. Предположим, вы объявили класс ORM, который описывает персону. Это все, что вам нужно, чтобы получить приложение для хранения контактной информации. Вы можете просто запустить приложение, и он будет отображать список персон, с помощью элемента управления сетки. Вы можете добавлять новые записи или модифицировать существующие. Эти операции выполняются с помощью автоматически сгенерированного набора отдельных редакторов; каждая связана с той или иной областью.

Автоматически сгенерированные элементы пользовательского интерфейса, используемые для отображения и управления данными называются представлениями. В XAF, существует три вида представлений:

- Представление списка (List View). Как правило, это таблица, в который отображают коллекцию, с которой вы работаете (таблицы данных). Вы видите

одну из них, когда вы запускаете свой проект, и вы можете переключаться между ними с помощью навигационной системы.

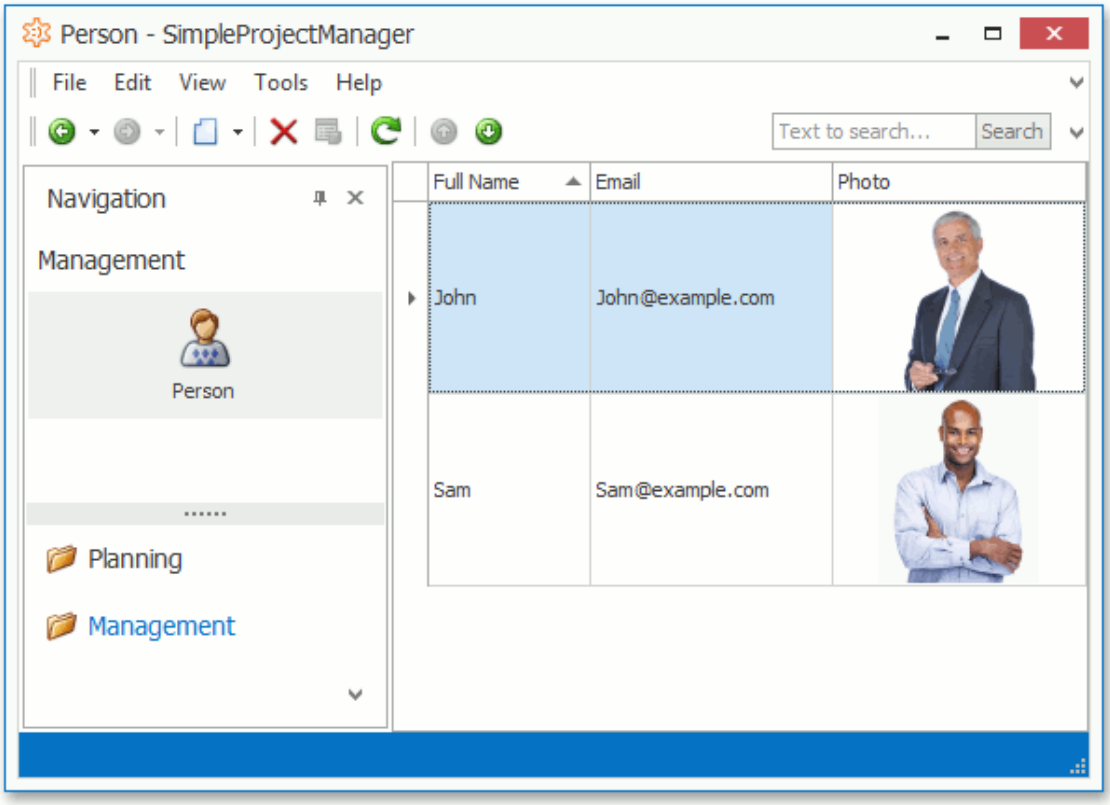


Рисунок 2.5 – Представление списка

- Представление деталей (Detail View). Этот тип представления рассматривается, как единый объект (запись данных). Представляет значения свойств с помощью автономных редакторов. Вы видите это представления при добавлении новой записи или при изменении

существующего.

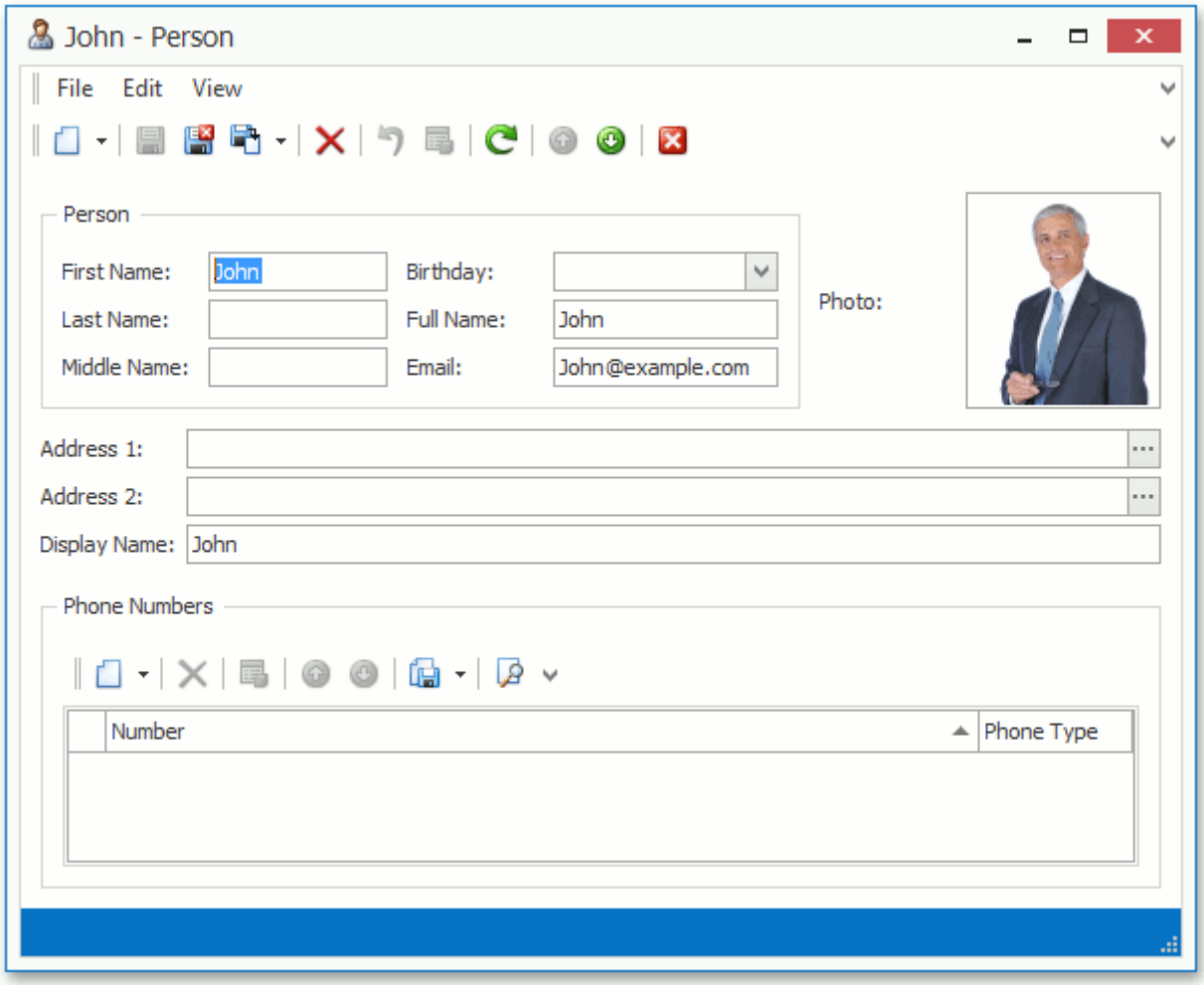


Рисунок 2.6 – Представление деталей

- Представление приборной панели (Dashboard View). Это особый тип представления, который позволяет отображать бок о бок несколько представлений на одном экране.

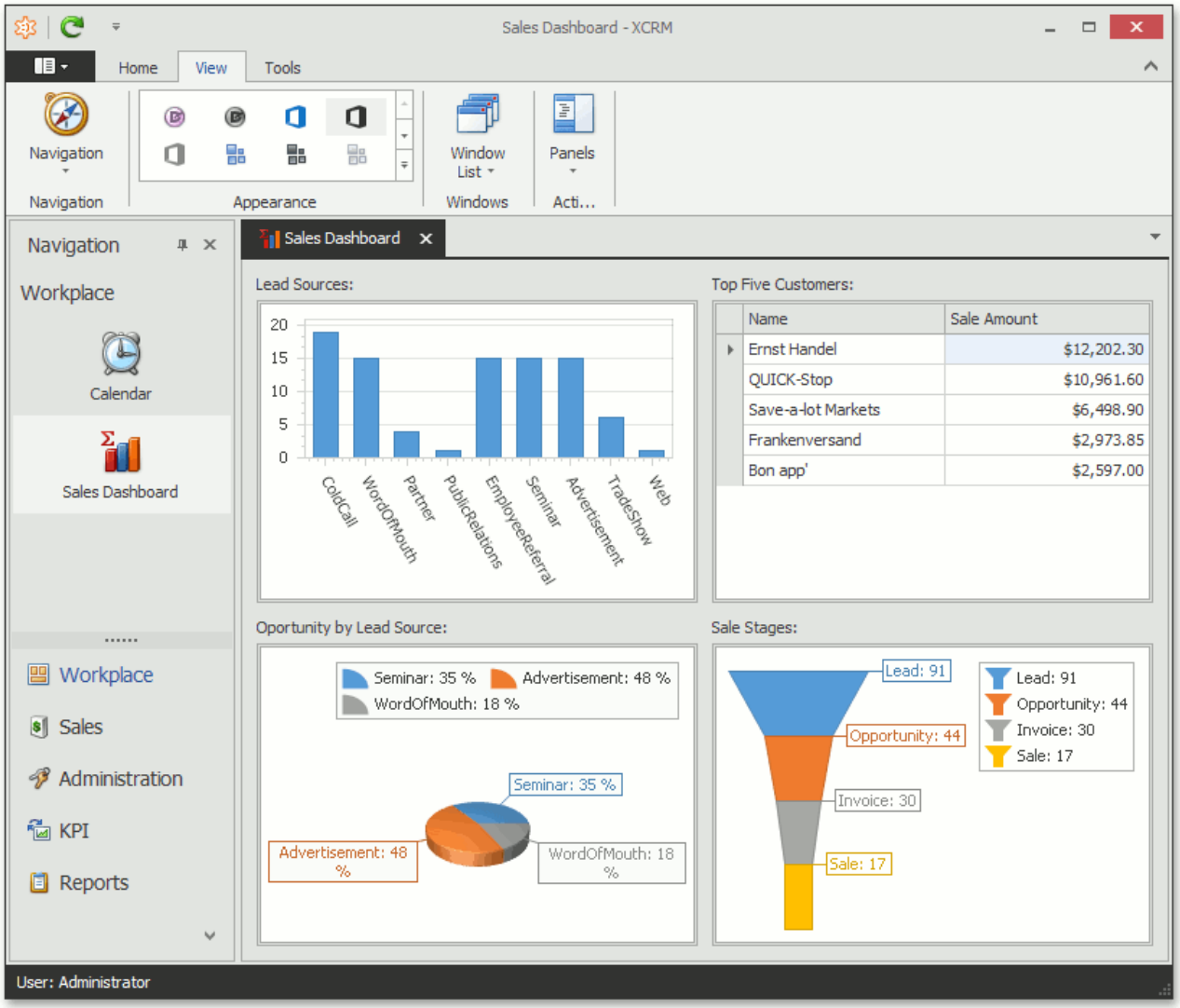


Рисунок 2.7 – Представление приборной панели

Представления, как правило, построены с помощью DevExpress WinForms и ASP.NET WebForms. Конечно, вы можете использовать любой элемент управления, вам требуется для просмотра представление списка (List View) или редактора внутри представления деталей (Detail View). Чтобы узнать, как поставить свой собственный контроль для приложения, обратитесь к Использование пользовательский элемент управления, который не интегрирован по умолчанию теме. Для получения информации о соображениях и других элементов, которые формируют пользовательский интерфейс, пожалуйста, просмотрите документы из раздела справки UI.

Отчеты

Бизнес-приложения не только используются для управления данными. Часто бывает полезно для анализа тенденций, проблемных областей, и узкие места в ресурсах. И, возможно, потребуется иметь свой отчет о данных не только на экране, но и на бумаге. Вы можете сделать это с помощью, встроенной, полностью функциональной отчетности двигатель - XtraReports

Suite. С ее помощью вы можете создавать любые отчеты, которые вам нужно. Просматривать их в Windows-приложении и веб-приложении ASP.NET. И конечно вы сможете распечатать их.

Все приложения, построенные с eXpressApp Framework могут включать в себя модуль отчетов. В Windows-приложении, конечный пользователь может добавить новый отчет и настроить его содержимое во время выполнения, с помощью встроенного конструктора конечного пользователя. Во время разработки, вы можете создавать отчеты для предопределенных конечных пользователей.

Отчета могут быть экспортированы в различных форматах, включая HTML, RTF и PDF.

В дополнение к особенностям отчета, eXpressApp Framework предоставляет дополнительные функции, которые могут быть использованы в приложениях XAF.

Поведение

Встроенные контроллеры

Контроллеры являются объектами, которые управляют потоком вашего приложения. Они также отвечают за взаимодействие с конечным пользователем. Даже самые простые приложения, построенные с eXpressApp Framework используют ряд встроенных контроллеров, поставляемых в комплекте с модулем системы и дополнительным модулем. Эти стандартные контроллеры в основном отвечает за управление данными. С их помощью вы можете добавлять новые записи, удалять существующие, выполнять полнотекстовый поиск и т.д.

По большей части, контроллеры служат в качестве контейнеров для действий. Как и ORM классы являются абстракциями таблиц данных, действия являются абстракциями элементов конечного пользователя взаимодействия - кнопки, меню и т.д. Действие определяет визуальное представление элемента пользовательского интерфейса и связанного с ним кода. Таким образом, вам не придется иметь дело с низкоуровневыми деталями реализации конкретных редакторов, систем панели инструментов контекстного меню или что-нибудь еще. И в то же время, это более высокий уровень абстракции позволяет то же самое действие, которые будут использоваться в Windows-приложении и веб-приложении ASP.NET.

Прикладная модель

Вся информация о том, что eXpressApp Framework использует для создания пользовательских интерфейсов происходит от модели приложения. Например, эта информация включает в себя классы редактора, используемые для конкретных типов данных, или метки, связанные с определенными полями. Модель приложения автоматически заполняется метаданными из компонентов приложения, как бизнес-объекты или контроллеры.

Файлы прикладной модели хранятся в формате XML, и поэтому их можно легко редактировать вручную. Но eXpressApp Framework обеспечивает

еще более простой способ – модель редактирования (Model Editor), который интегрирован с MS Visual Studio. Вы можете использовать его как для времени проектирования и выполнения настройки. Для того, чтобы запустить его во время разработки, дважды щелкните на .xafml файл из любого модуля или приложения проекта, расположенного в обозревателе решений.
Стандартная трехуровневая архитектура является хорошо зарекомендовавшей себя, и может стать основой для более сложных архитектур приложений.

- Хранилище данных (Data Source)
- Уровень доступа к данным(DAL)
- Бизнес логика (BL)
- Представление (UI)

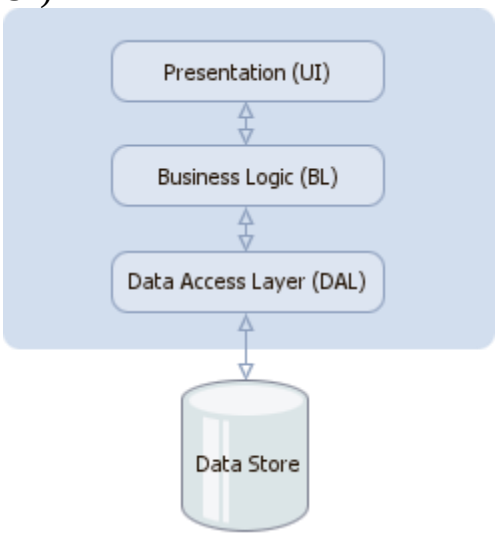


Рисунок 2.8 – Трехуровневая архитектура

Ускорение внедрения происходит за счет возможностей XAF по настройке готовой программы под Клиента. Настройка форм ввода, правил проверки данных, встроенная система контроля прав доступа, встроенный дизайнер отчетов – все это позволяет уже самому Клиенту сделать программу для себя комфортной, без привлечения разработчика.

3 Разработка программного обеспечения

3.1 ER-моделирование

Для разработки программного обеспечения был изучено текущего положение и составлены бизнес-правила.

Бизнес-правило 1

У каждого транспорта, водителя, пользователя, наряда, смены есть локация, данные сущности объединяет связь «один ко многим», и наоборот, множество транспортов, водителей, смен, пользователей, нарядов находятся в одной локации (см. рисунок 3.1).

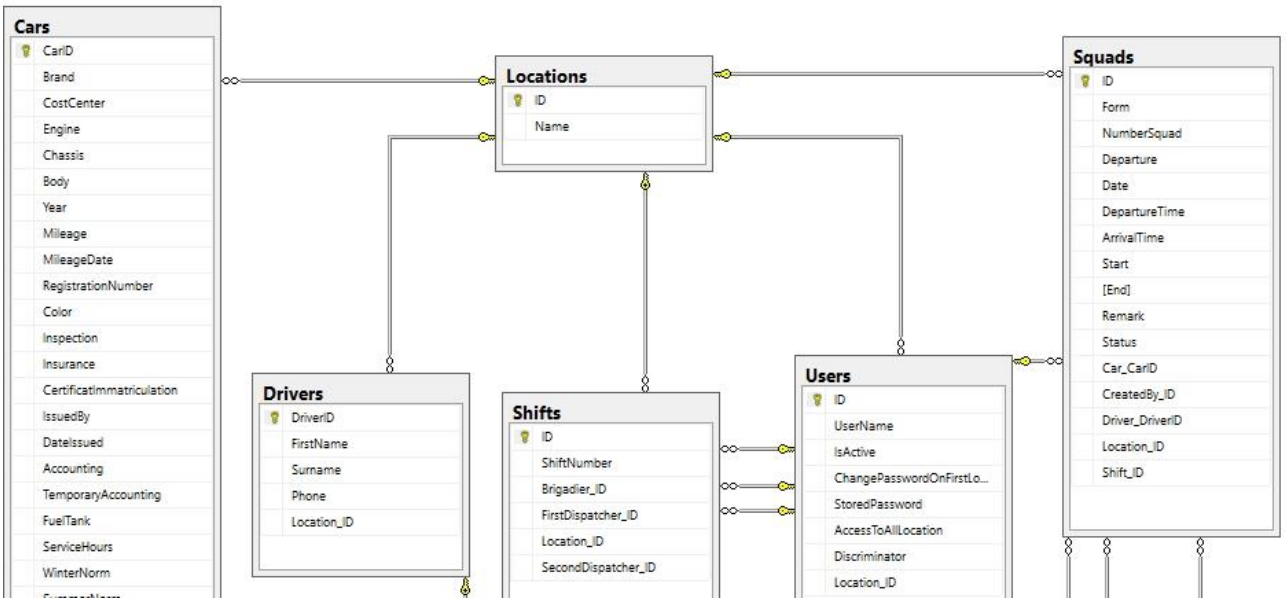


Рисунок 3.1 – Сегмент ER-модели для бизнес-правила 1

Бизнес-правило 2

Каждый транспорт нужно ремонтировать, данные сущности объединяет связь «один ко многим». Каждый тип ремонта состоит из множество наименований, данные сущности объединяет связь «один ко многим» (см. рисунок 3.2).

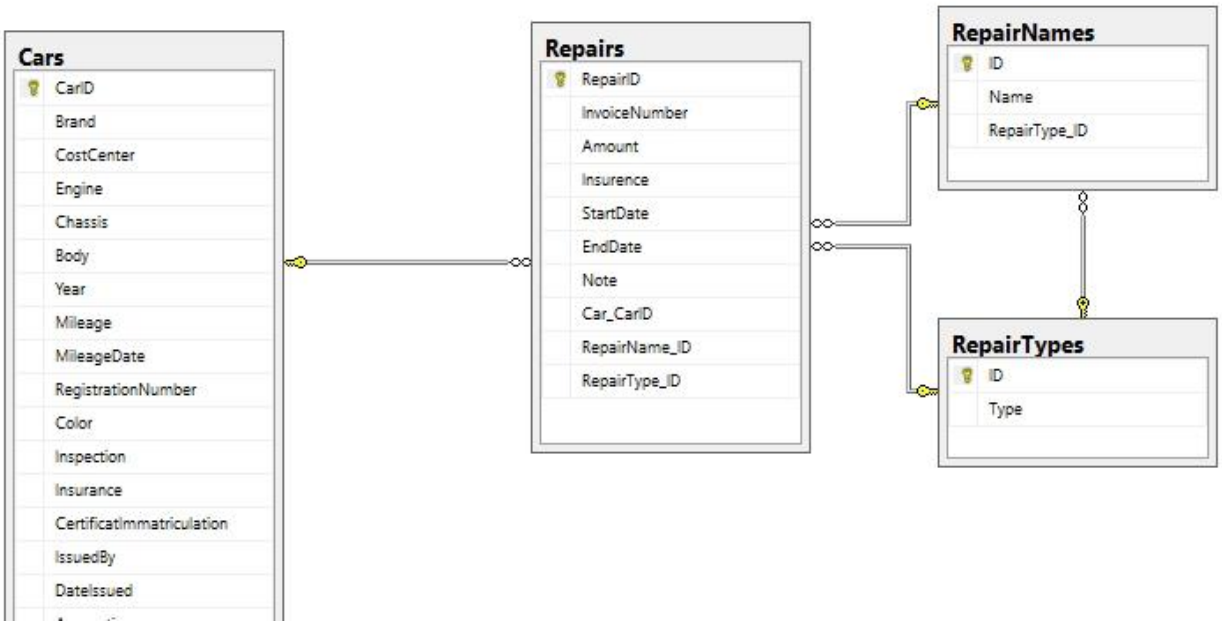


Рисунок 3.2 – Сегмент ER-модели для бизнес-правила 2

Бизнес-правило 3

Каждый транспорт имеет расходы за топливо, данные сущности объединяет связь «один ко многим». Каждый расход за топливо принадлежит одной машине «один ко многим» (см. рисунок 3.3).

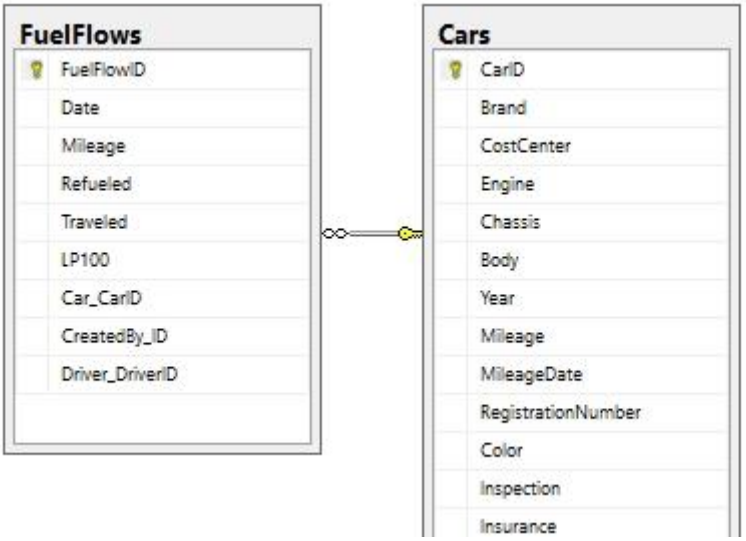


Рисунок 3.3 – Сегмент ER-модели для бизнес-правила 3

Бизнес-правило 4

Каждая смена состоит из двух диспетчеров, бригадира, данные сущности объединяет связь «один ко многим» (см. рисунок 3.4).

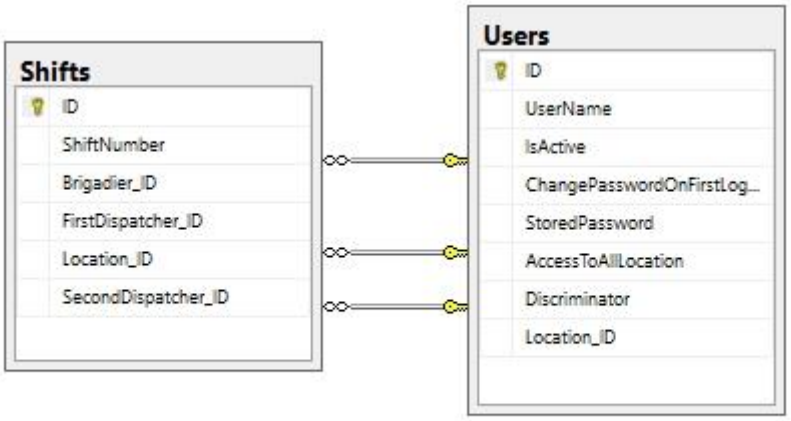


Рисунок 3.4 – Сегмент ER-модели для бизнес-правила 4

Бизнес-правило 5

Каждая водитель и транспорт состоит в смене, данные сущности объединяет связь «один ко многим» (см. рисунок 3.5).

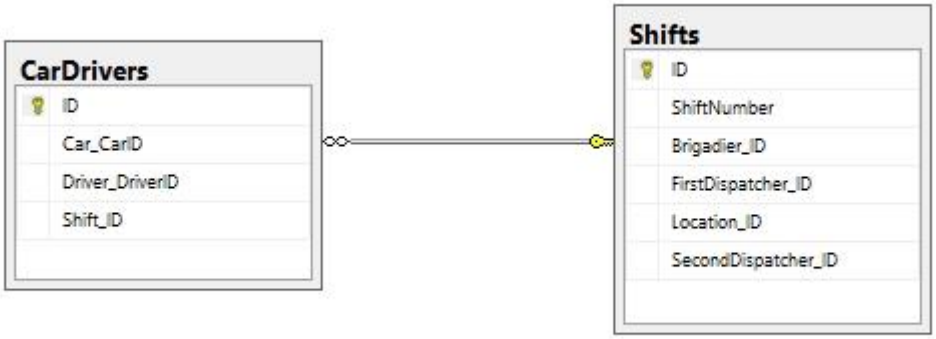


Рисунок 3.5 – Сегмент ER-модели для бизнес-правила 5

Бизнес-правило 6

В каждом наряде перевозятся множество пассажиров и экипаж, и поэтому данные объединяются ассоциативной таблицей (см. рисунок 3.6).

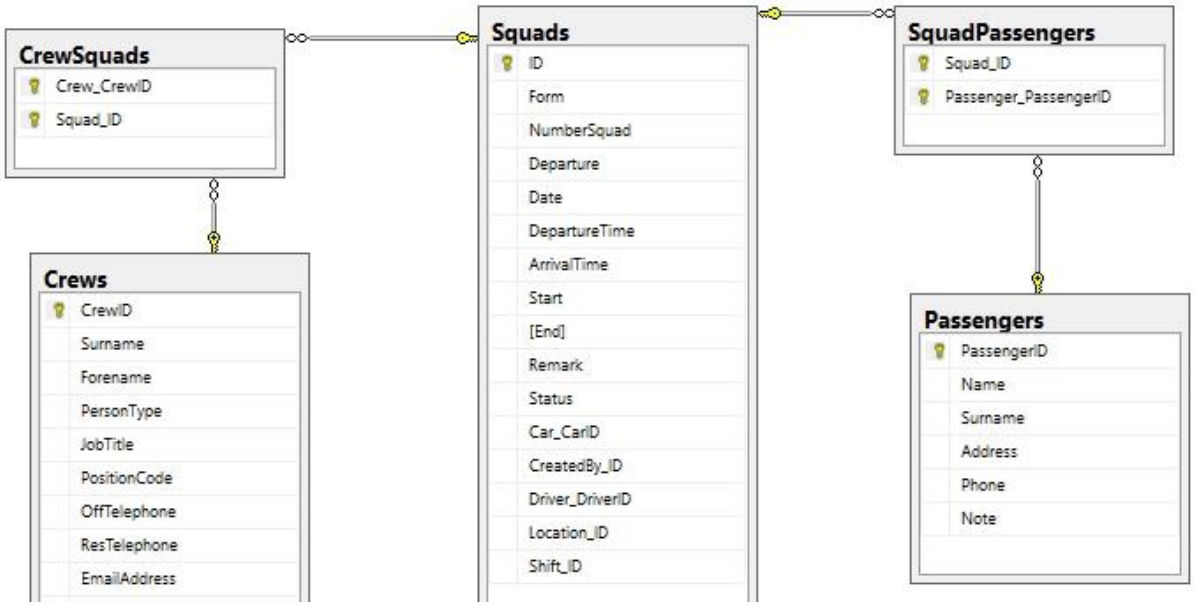


Рисунок 3.6 – Сегмент ER-модели для бизнес-правила 6

Бизнес-правило 7

Каждая наряд состоит из смены, водителя, транспорта, данные сущности объединяет связь «один ко многим» (см. рисунок 3.7).

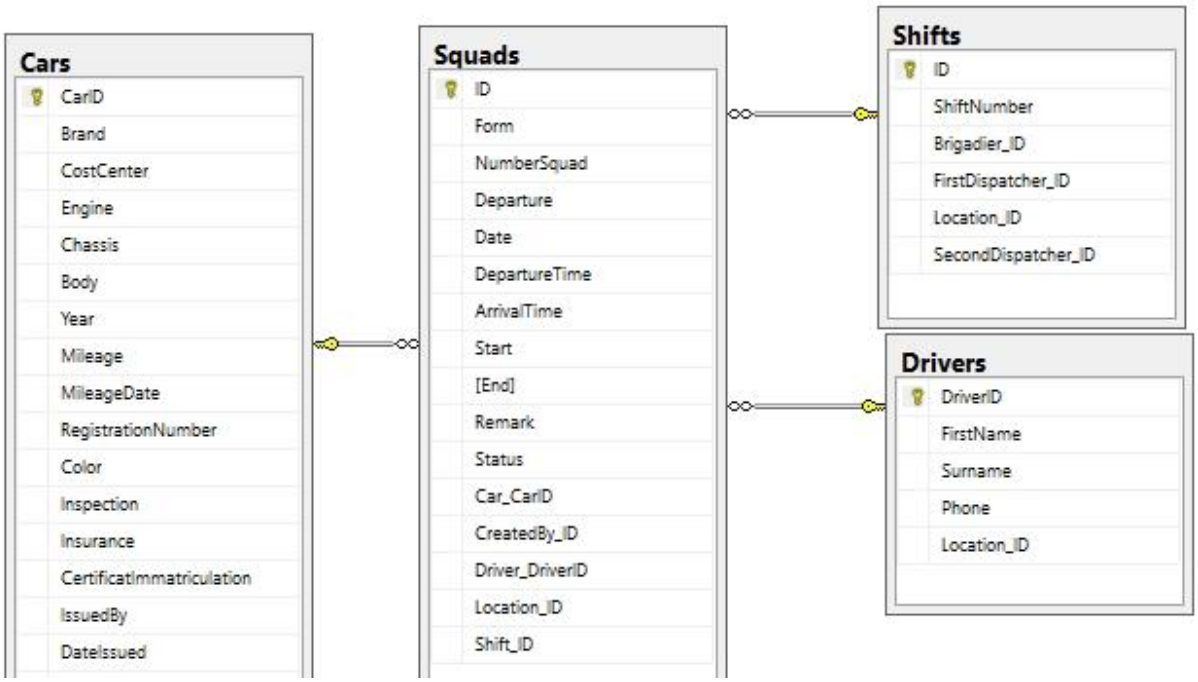


Рисунок 3.7 – Сегмент ER-модели для бизнес-правила 7

Бизнес-правило 8

Каждый пользователь может иметь множество ролей, и одна роль может быть у множества пользователей, и поэтому данные объединяются ассоциативной таблицей (см. рисунок 3.8).

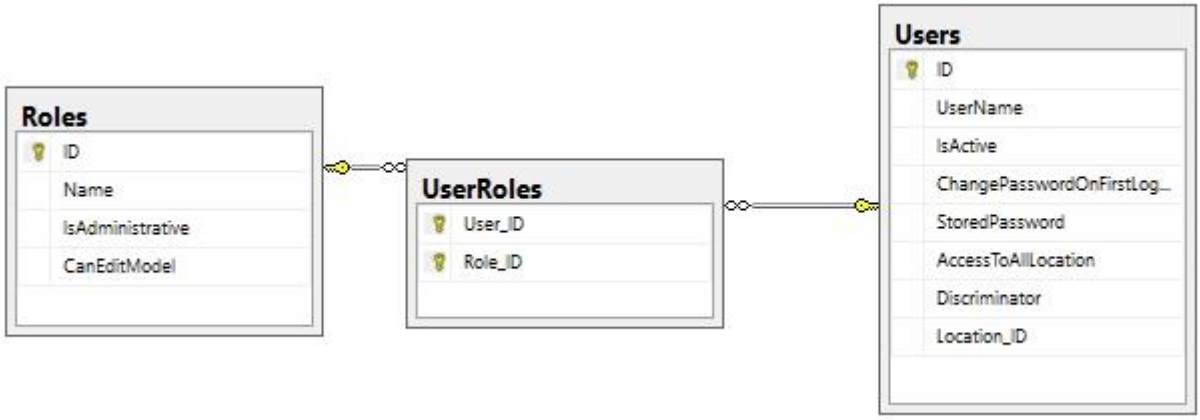


Рисунок 3.8 – Сегмент ER-модели для бизнес-правила 8

Бизнес-правило 9

Каждая роль может состоит из родительский и дочерних ролей, данные сущности объединяет связь «один ко многим» (см. рисунок 3.9).

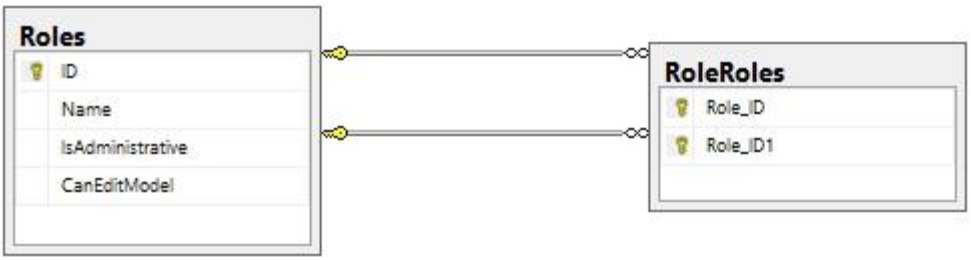


Рисунок 3.9 – Сегмент ER-модели для бизнес-правила 9

3.2 Описание интерфейса программного обеспечения

Пользовательский интерфейс очень удобен для использования. С данным программным продуктом могут работать множество пользователей. У каждого пользователя будет свой уровень доступа к данным.

После запуска приложения открывается форма «Вход в систему» (см. рисунок 3.10). Далее необходимо нажать вход в систему, без ввода пароля. Аутентификация пользователей происходит через AD (Active Directory). Active Directory – LDAP (Lightweight Directory Access Protocol — «облегчённый протокол доступа к каталогам») совместимая реализация службы каталогов корпорации MS для операционных систем семейства Windows Server. Если данного пользователя нет в базе данных, он не сможет пройти аутентификацию.

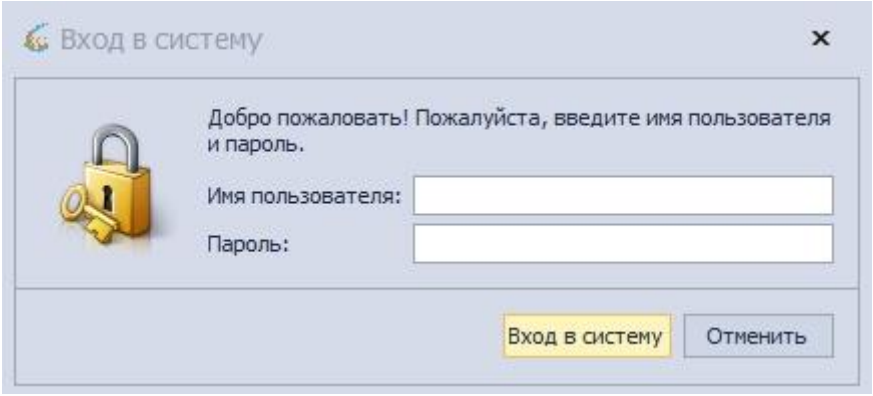


Рисунок 3.10 – Вход в систему

После входа в систему, открывается главное окно (см. рисунок 3.11), где вы сможете увидеть слева внизу имя пользователя.

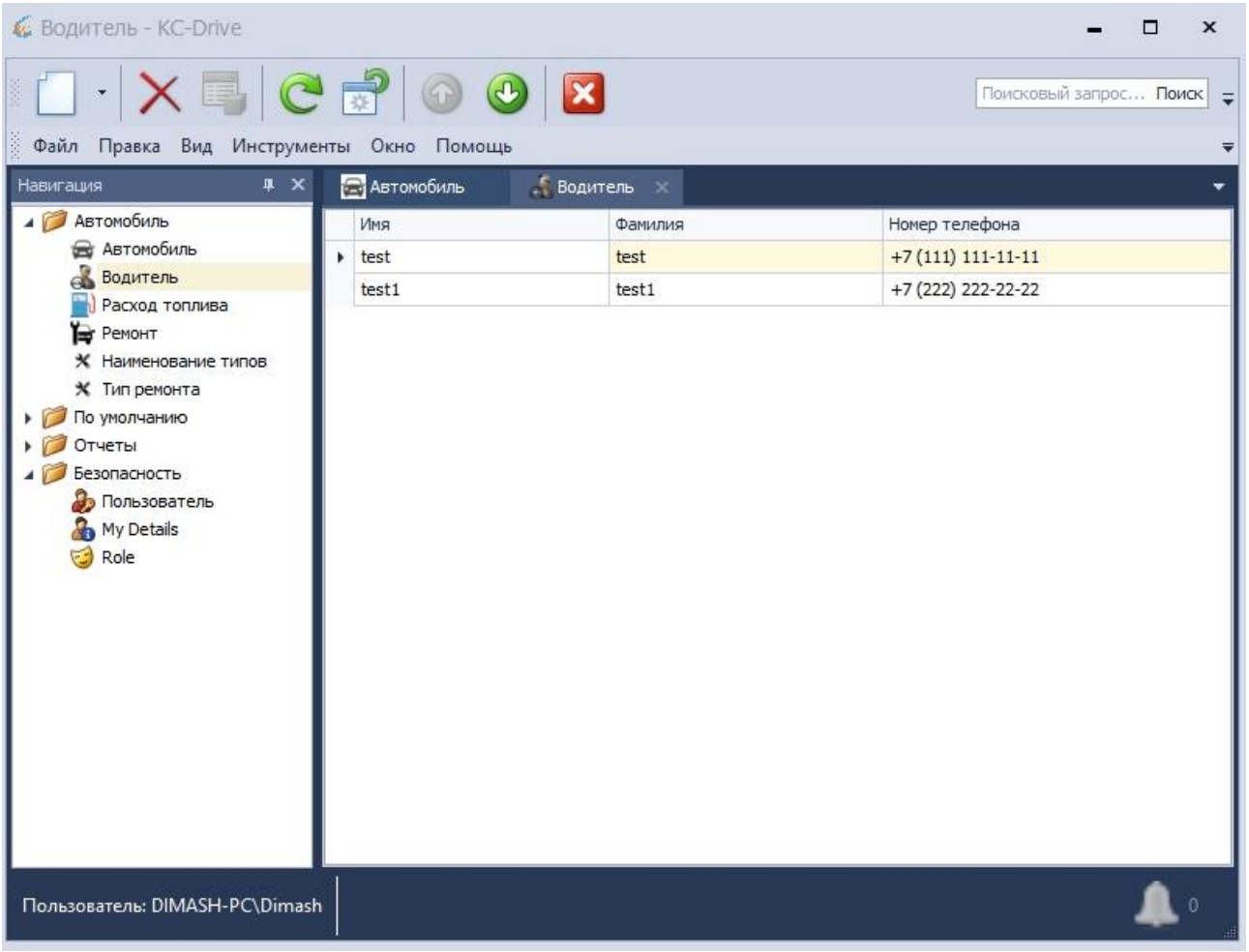


Рисунок 3.11 – Главное окно

Для создания нового пользователя нажимаем в панели «Навигация – Безопасность – Пользователь». Далее для создания нажимаем в правом верхнем угла создать (см. рисунок 3.12) или можно воспользоваться горячими клавишами «ctrl+N». В выпадающем списке выбираем «Пользователь» и

открывается окно представлений деталей (см. рисунок 3.14). Вводим нужные поля, и нажимаем кнопку сохранить или «ctrl+S».

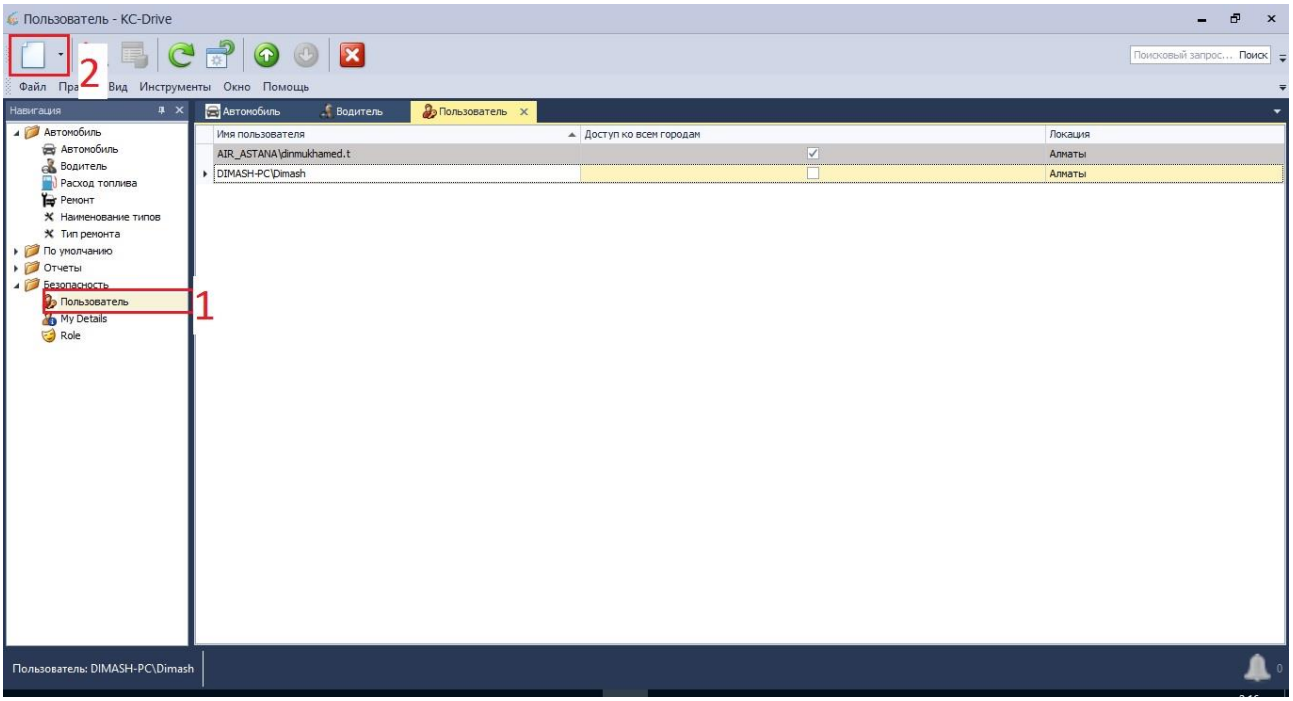


Рисунок 3.12 – Создание нового пользователя

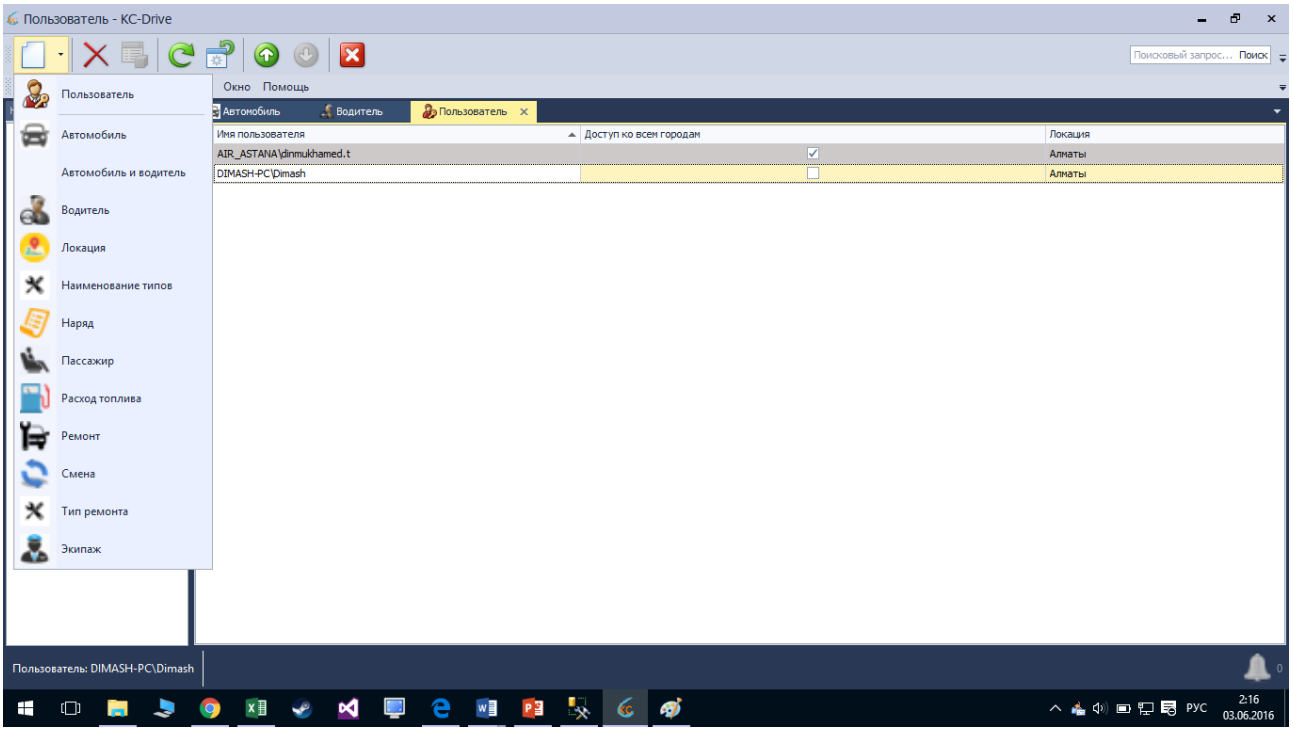


Рисунок 3.13 – Создание нового пользователя

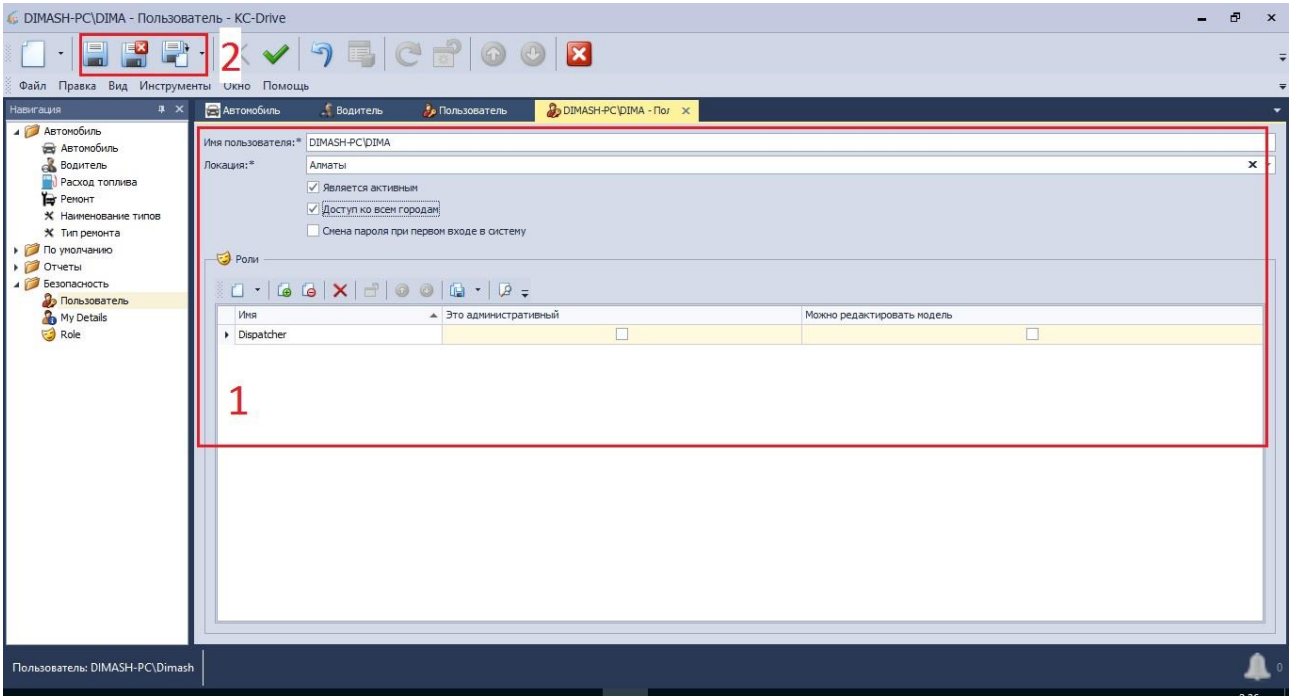


Рисунок 3.14 – Создание нового пользователя

Посмотрим расход по топливу. Для этого:

1. Выбираем «Навигация – Отчеты – Анализ».
2. Двойным кликом нажимаем на «Расход топлива».
3. В панели инструментов нажимаем на «Подключить данные для анализа».
4. Во вкладке «Сводная таблица» заполнится таблица (рисунок 3.15).
5. Перейдите во вкладку «Диаграмма» и увидите график (рисунок 3.16).

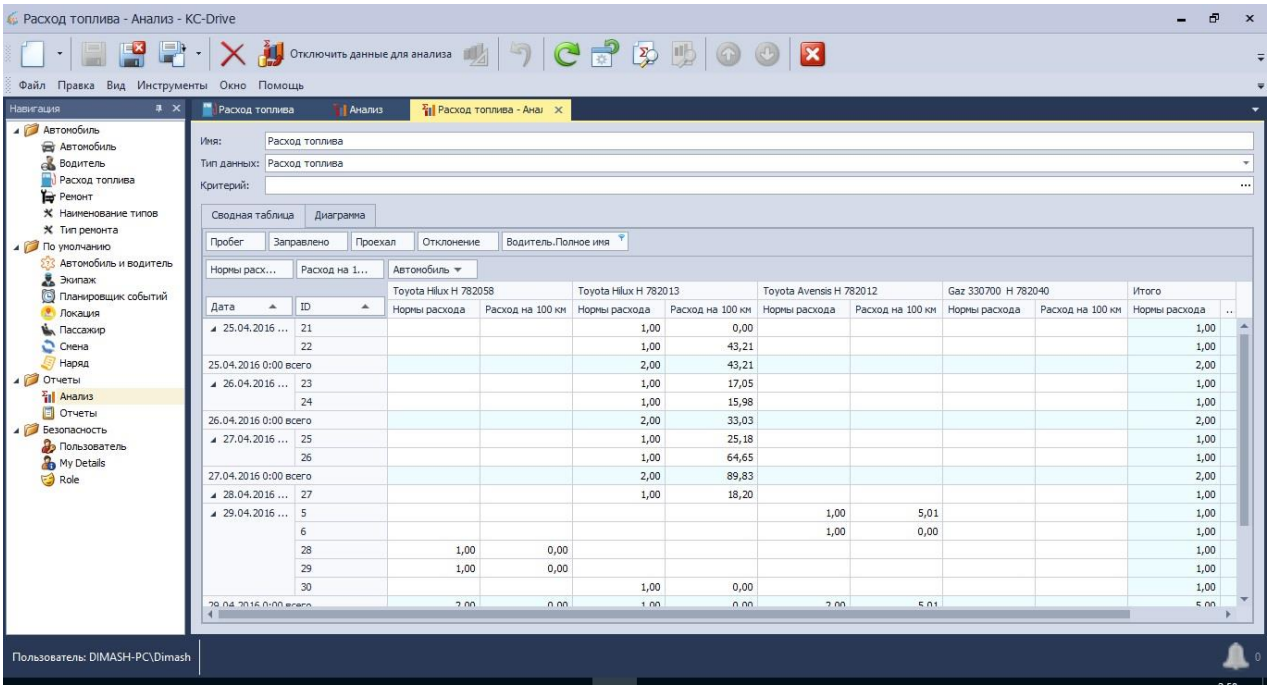


Рисунок 3.15 – Сводная таблица расхода топлива

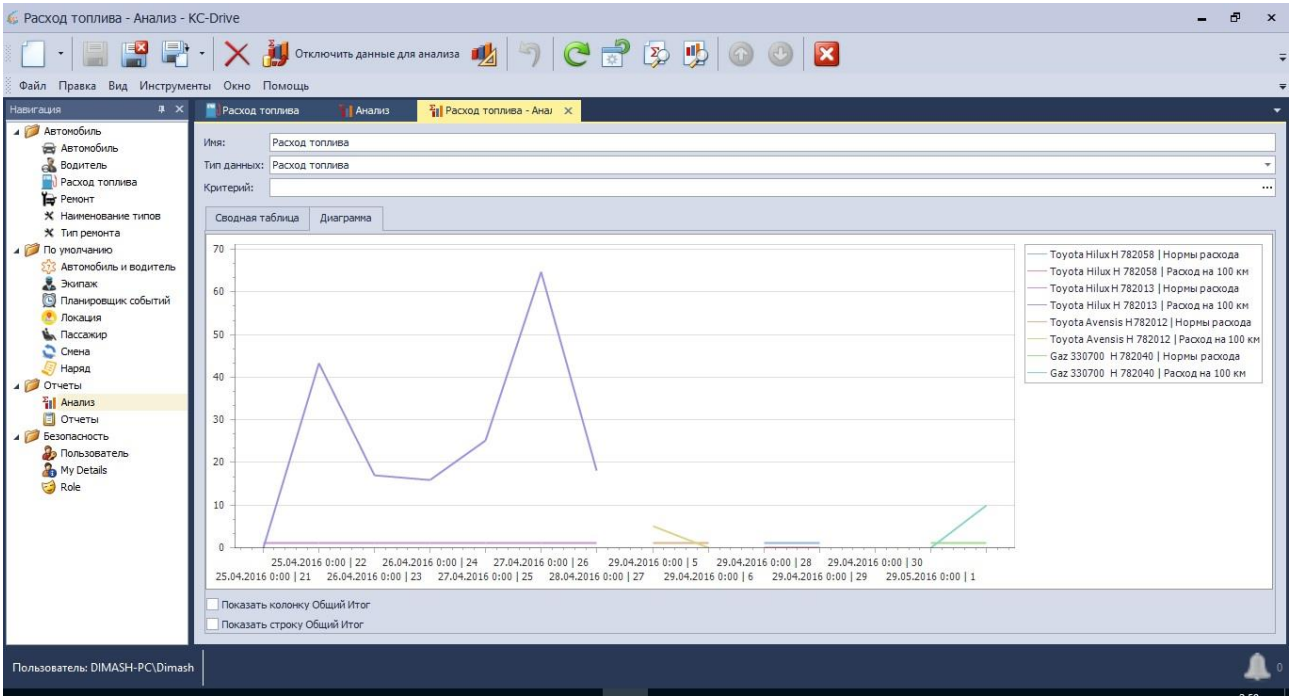


Рисунок 3.16 – Диаграмма расхода топлива

Создание наряда. Заполняем окно представления деталей. Добавляем пассажиров и с помощью веб-сервиса экипаж (рисунок 3.17).

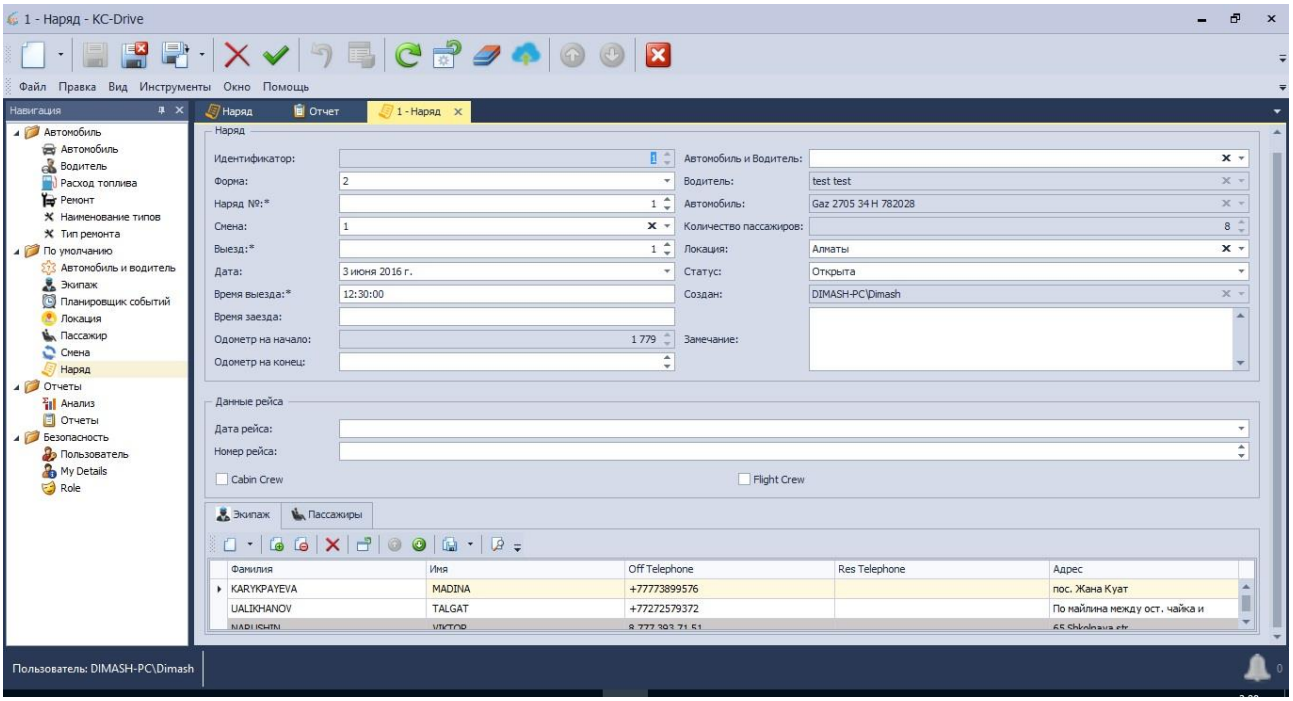


Рисунок 3.17 – Представление деталей наряда

Просмотр отчета по наряду. Выбираем отчет «Наряд (Между датами)». Выбираем даты и подгружаем отчет (рисунок 3.18).

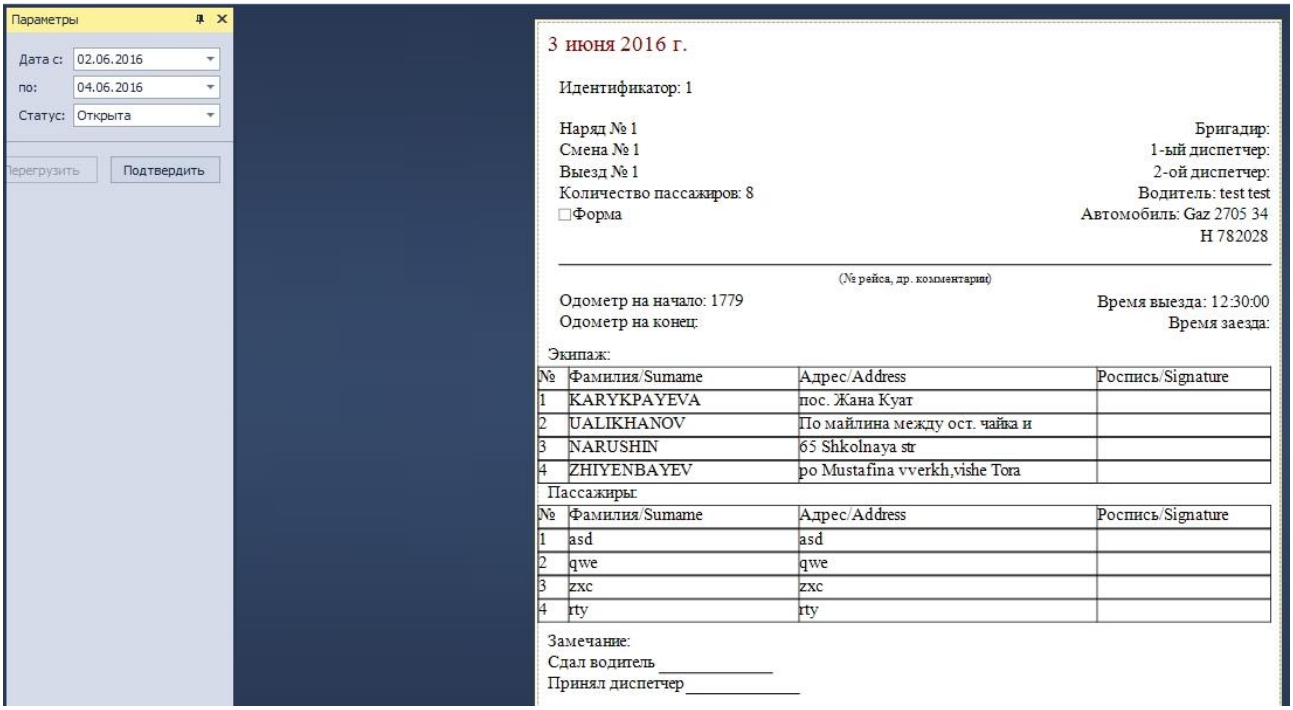


Рисунок 3.18 – Отчет по наряду

4 Безопасность жизнедеятельности

4.1 Анализ условий труда

С развитием научно-технического прогресса у людей важную роль играет возможность безопасного исполнения людьми своих должностных обязанностей. В связи с этим была создана и развивается наука о безопасности труда и жизнедеятельности человека.

Безопасность жизнедеятельности (БЖД) - это комплекс мероприятий, направленных на обеспечение безопасности условий труда человека в среде обитания, заботе его здоровья, разработку методов и средств защиты путем уменьшения влияния вредных и опасных факторов до допустимых значений, выработку мер по ограничению ущерба в устранении последствий чрезвычайных ситуаций.

Цель и содержание БЖД:

- поиск и изучение факторов окружающей среды, негативно влияющих на здоровье человека;
- уменьшения влияния этих факторов до допустимых пределов или исключение их, если это возможно;
- устранения последствий чрезвычайных ситуаций.
- Описание рабочего места:
- рабочее помещение находится на 4 этаже;
- помещение представляет собой комнату, размер которой 6x12x3 (ширина-длина-высота);
- в комнате 4 штук светильников, и в каждом по 4 люминесцентных ламп;
- окон в помещении два;
- созданию благоприятных условий для зрительного восприятия;
- для защиты от избыточной яркости с окон могут быть применены
- жалюзи;
- в помещении сидят 6 человека. Работают в дневную смену. Режим работы с 8:00 до 17:00.

Схема помещения приведена на рисунке 4.1.

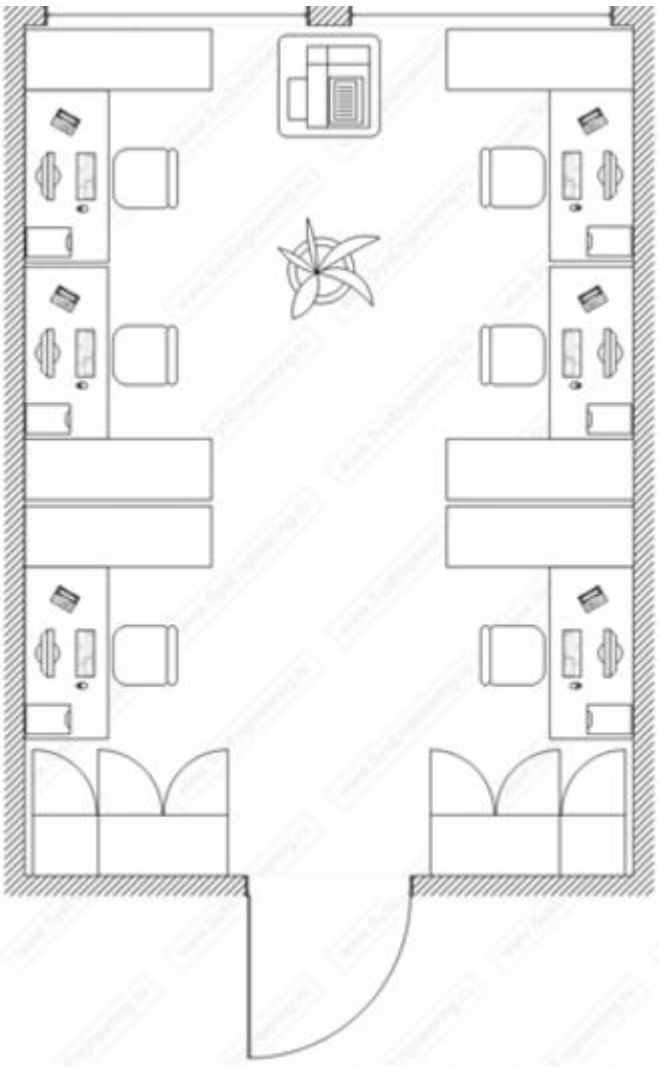


Рисунок 4.1 – План помещения

Условия труда пользователя, работающего с персональным компьютером, определяются:

- особенностями организации рабочего места;
- условиями производственной среды (освещением, микроклиматом, шумом, электромагнитными и электростатическими полями, визуальными эргономическими параметрами дисплея и т. д.);
- характеристиками информационного взаимодействия человека и персональных электронно-вычислительных машин.

При выполнении работ на персональном компьютере (ПК) согласно ГОСТу 12.0.003-74 ССБТ. Опасные и вредные производственные факторы. Классификация" могут иметь место следующие факторы:

- повышенная температура поверхностей ПК;
- повышенная или пониженная температура воздуха рабочей зоны;
- выделение в воздух рабочей зоны ряда химических веществ;
- повышенная или пониженная влажность воздуха;

- повышенный или пониженный уровень отрицательных и положительных аэроионов;
- повышенное значение напряжения в электрической цепи, замыкание;
- повышенный уровень статического электричества;
- повышенный уровень электромагнитных излучений;
- отсутствие или недостаток естественного света;
- недостаточная искусственная освещенность рабочей зоны;
- повышенная яркость света;
- монотонность трудового процесса;
- нервно-эмоциональные перегрузки.

Мебель, часто используемая разработчиком: это рабочий стол и кресло. Основное рабочее время разработчик проводит в сидячем положении, которое практически не вызывает утомления.

Одним из основных факторов, влияющих на работоспособность, предотвращения заболеваний и утомлений сотрудника является освещенность. Утомляемость органов зрения зависит от ряда причин:

- недостаточность освещенности;
- чрезмерная освещенность;
- неправильное направление света.

Недостаточность освещения является причиной напряжения органов зрения, ослабляется внимание, и появляется усталость и утомленность. Чрезмерная освещенность ослепляет, раздражает и режет глаза.

Микроклимат может пагубно влиять на состояние организма. Например, высокие температуры вызывают быструю утомляемость, перегрев организма, а также повышенное потоотделение, в то время как низкие температуры способствуют местному или общему переохлаждению организма, к тому же могут стать причиной различных заболеваний.

В комнате, где есть компьютеры, особо важен некоторые параметры микроклимата. При санитарных нормах СН-245-71 при котором значение параметров микроклимата благоприятны для создание комфортной среды обитания работника. Эти нормы устанавливаются в зависимости от сезона, характера трудового процесса и характер производственных объектов приведены в таблице 5.1.

Объем кабинета, в которых размещены работники, программисты составляют не меньше, чем 19.5 $м^3$/человека с максимальным количеством одновременных работников в смену. Поставка норм свежего воздуха в кабинет, где размещены компьютеры, приведены в табл. 4.2.

Таблица 4.1 - Параметры микроклимата для кабинета, где установлен компьютер

Сезон года	Параметр микроклимата	Величина
Холодный	Температура воздуха в кабинете	22…24 °С

Холодный	Относительная влажность	40…60%

Окончание таблицы 4.1

Сезон года	Параметр микроклимата	Величина
Холодный	Скорость движения воздуха	до 0,1 м/с
Теплый	Температура воздуха в кабинете	23…25°С
Теплый	Относительная влажность	40…60%
Теплый	Скорость движения воздуха	0,1…0,2 м/с
Примечание – составлено автором на основе источника [1]		

Таблица 4.2 – Нормы подачи свежего воздуха в помещения, где расположены компьютеры

Характеристика кабинета	Объемный расход подаваемого в кабинет свежего воздуха, $м^3$ на одного человека в час
Объем до 20$м^3$ на человека	Не менее 30
20…40$м^3$ на человека	Не менее 20
Более 40$м^3$ на человека	Естественная вентиляция
Примечание – составлено автором на основе источника [2]	

Обеспечить комфортные условия организации работы в зависимости от сезона используют организационные методы (рациональной, чередование труда и отдыха) и оборудование (вентиляция, кондиционирование, системы отопления).

Правильно разработанные и выполненные промышленное освещение улучшает условия зрительной работы, снижает утомляемость, повышает производительность труда, благотворно влияют на производственную среду, оказывая положительное психологическое воздействие на работника, повышает безопасность и снижает риск травм.

Недостаток света вызывает зрительное напряжение, снижает внимание, приводит к преждевременному наступлению усталости. Чрезмерно яркое освещение вызывает бликов, тем самым оказывая раздражение и боль в глазах. Искажение направление света на рабочем месте могут создать резкие тени, отражения, дезориентировать работника. Все эти факторы могут привести к аварии или профилактическому заболеванию, поэтому очень важен правильный расчет параметров освещения. Есть три типа освещения - естественное, искусственное и смешанное (искусственное и естественное вместе).

Неблагоприятными факторами для профессиональной деятельности разработчика, негативно влияющими на зрение, являются:

- пониженный уровень освещенности, приводящий к перенапряжению глаз и, как следствие к быстрому утомлению.

- чрезмерно высокая освещенность также является причиной быстрой утомляемости, приводя к раздражению и рези в глазах;

- неправильное направление света способствует появлению резких теней или сильных бликов, заметно быстрее утомляющих глаза.

Для освещения помещений и рабочих мест с ПК должно применяться естественное, искусственное и совмещенное освещение.

Рабочие места с ПК по отношению к световым проемам должны располагаться так, чтобы естественный свет падал сбоку, преимущественно слева.

Искусственное освещение в помещениях, в которых эксплуатируются ПК, должно осуществляться системой общего равномерного освещения. Хорошо проникающий в помещение естественный свет оказывает благоприятное воздействие на психику человека, вызывая положительные эмоции, обеспечивая хорошие гигиенические условия работы. За счет естественного освещения стимулируется обмен веществ, кровообращение, дыхание, что в свою очередь, обеспечивает высокую производительность труда.

Согласно СНиП II-4-79 в вычислительных центрах помещение должны применяться в комбинированной системе освещения.

При проведении визуальной работы категории высокой точности (минимальный размер объекта дискриминации на 0,3 – 0,5 мм). Коэффициент естественного освещения (КЕО) не должна быть ниже 1,5%, в то время как средняя точность зрительных работ КЕО должен составлять не ниже 1.0%. Вместо источников искусственного освещения широко применяют люминесцентные лампы типа ЛБ или ДРЛ, которые объединившись в светильники должны быть помещены над рабочей поверхности равномерно.

Требования освещения света в помещениях, где компьютеры устанавливаются следующим образом: зрительная работа при высокой точности составляет 300 лак освещения, при комбинированной 750 лак; соответственно при работе средней класса точности требуется от 200 до 300 лак.

Основным гигиеническим требованием является равномерное освещение всего поле зрения работающего. Другими словами, степень освещенности помещения и яркость экрана компьютера должны быть примерно равными, потому что яркий свет в периферийной зоне повышает интенсивность напряжения глаз и, как следствие, приводит к их усталости.

Освещение должно быть равномерным во всей комнате, особенно в рабочей зоне. Одинаково плохи и темный угол за шкафом, и непрерывно освещаемое солнцем пространство возле окна.

Таблица 4.3 - Требования к освещенности помещения

Нормы освещенности	
Экрана	от 100 до 250 лак

Стола	от 300 до 500 лак
Яркость экрана	35 кд/м2, не менее
П р и м е ч а н и е – составлено автором на основе источника [3]	

Шум, воздействуя на организм человека, снижает условие труда его работы. Работая долгое время при воздействии шума программист ощущает на себе раздражительность, головные боли, головокружение, частичную потерю памяти, утомляемость, потерю аппетита, и боли в ушах. Целый ряд таких нарушении систем человеческого организма способствует к неблагоприятному эмоциональному состоянию человека до стрессового. Заострение внимания под влиянием сильного шума у работника наблюдается снижение, нарушение физиологических функций, появляется усталость из-за затрачивания энергии и умственных нагрузок, и ухудшение голос переключения. Все это сказывается на работоспособности программиста и на его производительность, качество и безопасность работы.

В таблице 4.4 указаны в зависимости от категории тяжести труда и напряженности, приближенные к уровню звука, являются безопасными по отношению сохранения здоровья и работы труда программиста.

Таблица 4.4 - Предельные уровни звука, дБ, на рабочих местах

Категория напряженности труда	Низкая	Средняя	Высокая	Очень высокая
Мало напряженный	80	80	75	75
Умеренно напряженный	70	70	65	65
Напряженный	60	60	-	-
Очень напряженный	50	50	-	-
П р и м е ч а н и е – составлено автором на основе источника [4]				

На рабочем месте программиста уровень шума видеоматериала не должны быть выше – 50 дуба, и в вычислительных центрах обработки информации на компьютере – 65 дуба. Компьютеры, выложенные противочумным материалом для понижения уровня шума стены и потолка в комнате. В вычислительны центрах степень вибрации можно понизить путем установки специального оборудования на виброизоляторах.

В настоящее время ученых считают, что краткосрочное и долгосрочное оказание всех видов излучений от экрана монитора не является опасным для здоровья персонала, обслуживающего вычислительную технику. Следует заметить, что взаимоисключающие данные о вреде опасности, связанных с излучением от экранов монитора, работающих на компьютерах, так и нет, и исследований в этой области продолжается.

В таблице 4.5 показаны допустимые значения параметров неионизирующих электромагнитных излучений от экрана компьютера.

Максимальный уровень рентгеновского излучения на рабочем месте оператора персонального компьютера, как правило, не превышает 10 км Бар/ч,

и интенсивность ультрафиолетового и инфракрасного излучений от экрана монитора в диапазоне от 10 до 100 м Вт/м^2.

Таблица 4.5 - Возможные значения параметров неионизирующих электромагнитных излучений

Наименование параметра	Возможные значения
На расстоянии 50см от грани монитора	10 В/м
На расстоянии 50см от грани монитора	0,3 А/м
Для пользователей и детей дошкольных учреждений и учащихся в средних специальных и высших учебных заведениях	15 кВ/м
П р и м е ч а н и е – составлено автором на основе источника [5]	

Для уменьшения влияния этих видов излучения рекомендуется использовать мониторы с ограниченным излучением (MPR-II, TCO-92, TCO-99), установка щитов, а также сохранение регламентированных режимах труда и отдыха.

Проектирование рабочих мест, оснащенных видеотерминалами, является одним из важнейших вопросов эргономичный дизайн в области компьютерных наук.

4.2 Организация рабочего места.

Стол и кресло являются главными элементами рабочего места программиста. Основное положением это положение сидя.

Минимальную усталость вызывает у работника во время рабочих заседании положение сидя. Рациональная планировка рабочего места обеспечивает четкий порядок и последовательность размещения объектов, ручные инструменты и документацию. То, что требуется для выполнения работы, чаще всего находятся в зоне легкой досягаемости рабочего пространства.

Используемые при работе устройства расположить в пределах зоны досягаемости моторного поля (рисунок 5.2), а средства отображения информации - с учетом зон отображения информации.

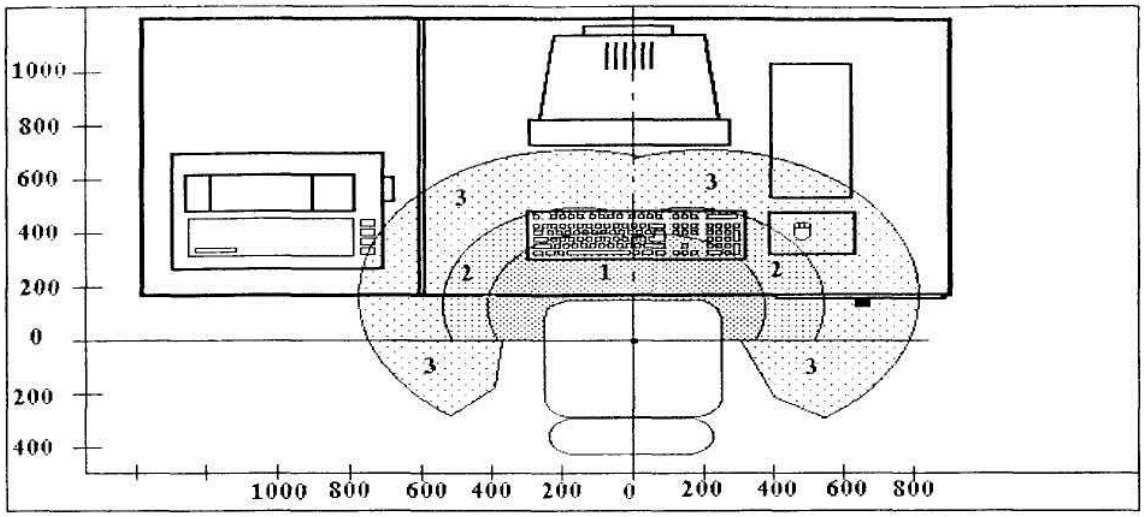


Рисунок 4.2 – Зона досягаемости моторного поля

- 1-я зона для размещения наиболее важных и очень часто используемых органов управления (оптимальная зона моторного поля) – сектор 60°;

- 2-я - зона для размещения часто используемых органов управления (зона легкой досягаемости моторного поля) – сектор 120°;

- 3-я- зона для размещения редко используемых органов управления (зона досягаемости моторного поля.

Часто используемые средства информации располагают под углом ±15° от нормальной линии взгляда, идущей на 15° ниже горизонтальной линии взгляда.

При пространственной организации рабочего места учитываются антропометрические данные, выбор рациональной рабочей поверхности, физиологически рациональной рабочей позы, оргтехоснастка, защита от блёскости.

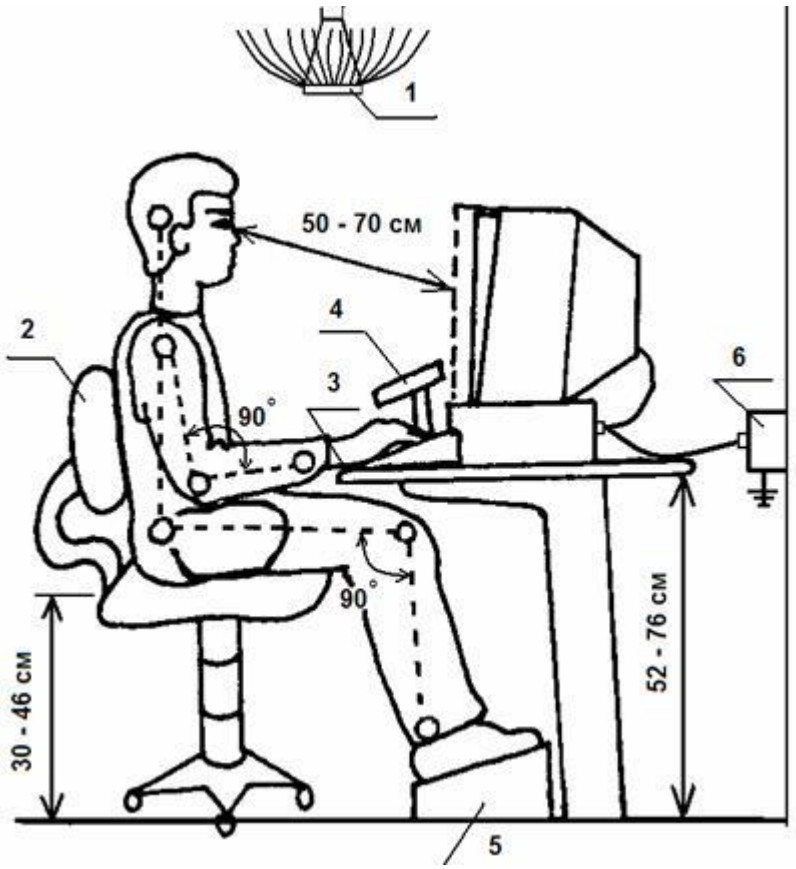


Рисунок 4.3 – Рекомендуемая организация рабочего места

Расположение основных и периферийных составляющих персонального компьютера на рабочем столе программиста:

1. Аппарат аэроионной профилактики;
2. Рабочее кресло;
3. Подставка под кисти рук;
4. Подставка под документы;
5. Регулируемая по высоте подставка для ног;
6. Заземленная панель питания.

Для удобной работы программиста поверхность стола должен удовлетворять следующим требованиям:

а) высота стола должна представляться таким образом, чтобы учитывая возможность, сидеть свободно в удобном расположении, не было необходимостью основываться на подлокотники;

б) в нижней части раздела должны быть рассчитаны места для того, чтобы программисту удобно было расположиться, и не было нужды поджимать ноги;

в) в поле зрения работающего грань, на которой он работает должна обладать таким свойствам, чтобы было возможным предотвратить появление бликов;

г) стол должен быть сконструирован так, чтобы включать в себя несколько выдвижных ящиков (не менее 3-х, где можно будит хранить документы, списки и канцтовары);

Высота грани, в которой располагается клавиатура, которая не должна превышать - 650 мм.

Большое внимание уделяется для характеристики стола и стула, работающего. Таким образом, оптимальная высота сиденья над полом должна располагаться в пределах 420 - 550 мм. Сиденья должны быть мягкими с закругленными передними краями и угол наклона спинки должен регулироваться.

Необходимо обеспечить при организации расположение различных документы на поверхности стола со стороны видео терминала, а также со стороны между монитора и клавиатуры. Кроме того, в тех случаях, когда устройство отображения видео плохого качества изображения, а именно видны мелькания. Расстояние от глаз до монитора размера должно быть около 700 мм, когда расстояние от глаз до документа составляет 300 - 450 мм. Расстояние пользователя до экрана от глаз, как правило при высоких качествах изображения, клавиатура и документы могут быть равны.

Размещение экрана определяется:

- расстояние подсчитывается от 0,6 до 0,7 м;
- угол чтения просмотра зрения направления в 20 градусов ниже горизонтальной и на экране перпендикулярно к этому направлению.

Предусмотрена возможность управление экраном:

- 3 см в высоту;
- по склону от минус 10 до плюс 20 °С относительно вертикали;
- по левой и правой стороне.

Огромное внимание оказывает правильная рабочая поза пользователя. Когда неудобно сидеть на работе, то могут появиться боли в мышцах, суставах и сухожилиях. Условия к рабочей позе видео терминала пользователя должна удовлетворять следующие требования:

- при наклоне головы более чем на 20 градусов;
- расслабленные плечи;
- локти расположены под углом 80 - 100 градусов;
- предплечья и руки расположены в горизонтальном положении.

Причина из-за неправильной осанки пользователя обусловлены следующими факторами: отсутствие хорошей подставки для документов, высокое расположение клавиатуры, документы расставлены ниже, для того чтобы было куда расположить руки и ноги.

Для того чтобы преодолеть эти недостатки предоставляют общие рекомендации: использование мобильной клавиатуры; должны предоставить специальные инструменты для регулировки высоты стола, клавиатуры, монитора и подставки для рук.

Размеры символов являются необходимым для продуктивной и качественной работы на компьютере, расположение их плотностей, контрастность и яркость отношение символов и фона монитора. Если расстояние от глаз до дисплея оператора составляет 60 - 80 см, высота надписи должна быть не менее 3 мм, оптимальным решением в отношении ширина и высота знака с указанием 3:4, а расстояние между метками – от 15 до 20% их высоты. Соотношение яркости фона экрана и знаков характеризуется от 1:2 до 1:15.

Во время компьютерных работ медики рекомендуют установить монитор на расстоянии 50 - 60 см от глаз. Эксперты считают, что в верхней части видео должна быть на уровне глаз или ниже. Когда человек смотрит прямо, то его глаза открыты намного шире, чем, когда он смотрит вниз. Кроме того, если установлен экран высокого льна и программист смотрит с широко открытыми глазами, тем самым ухудшается функция моргания.

Оказание благоприятных условий труда, правильный эстетический внешний вид рабочие места в промышленности имеет огромное значение, поскольку это облегчает производительность работы.

Неоднократно отмечалось, что при работе с вычислительной машиной, играет очень важную роль надлежащее соблюдение работы и отдыха. В противном случае у сотрудников отмечали жалобы: значительную напряженность в зрительном аппарате, боли в голове, расстройство и нарушение сна, боли шей, рук и в позвоночной области, что вызывало недовольство работой.

Информация о регламентированных перерывах представлены в таблице 4.6 необходимые при работе на компьютере, в зависимости от продолжительности рабочей смены, типов и категорий работы с ВДТ (видеотерминала) и ПК (в соответствии с СанПиН 2.2.2 542-96 "Гигиенические требования к видео-дисплейному терминалу, персональным электронно-вычислительным машинами и работ").

Таблица 4.6 – Время регламентированных перерывов при работе на компьютере

Категория работы с ВДТ или ПК	Уровень нагрузки за рабочую смену при видах работы с ВДТ			Общее время регламентированных перерывов	
	Группа А, количество знаков	Группа Б, количество знаков	Группа В, часов	При 8-часовой смене	При 12-часовой смене
1	20000, не более	15000, не более	2, не более	30	70
2	40000, не более	30000, не более	4, не более	50	90
3	60000, не	40000, не	6, не более	70	120

	более	более			

Перерывы введены в соответствии с конкретными медико-санитарными правилами и нормами. Время перерывов увеличивается на 30% при несоответствии фактических условий труда санитарным правилам и нормам. Все виды работ, относящие к работам с использованием вычислительной машиной, разделены на три группы в соответствии с СанПиН 2.2.2 546-96. Группа «А» включает в себя работу на чтение информации с экрана ВДТ или ПК с преждевременным запросом. Ввод информации включает в себя группа «Б». Творческая работа режима диалога с компьютером относится к группе «В».

Удобная мягкая мебель, расположение аквариумов, живых цветов, либо специально выделенного помещения с газоном хорошо скажутся при работе во время перерывов. Также можно заняться легкой гимнастикой, что эффективно отразиться на работе программиста.

4.3 Расчет искусственного освещения

Освещенность рабочей поверхности, создаваемая светильниками общего освещения в системе комбинированного, должна составлять не менее 10% нормируемой для комбинированного освещения при тех источниках света, которые применяются для местного освещении. При этом освещенность должна быть не менее 400 лак при люминесцентных лампах. В таблице 4.7 приведены нормы освещенности при искусственном освещении.

Таблица 4.7 - Нормы освещенности при искусственном освещении

Характеристика зрительной работы	Наименьший или эквивалентный размер объекта различения, мм.	Разряд зрительной работы	Искусственном освещение
			Освещенность, лак
			При комбинированном освещении.
Средней точности	Свыше 0.5 до 1.0	IV	400

Расчет освещенности рабочего места сводится к выбору системы освещения, определению необходимого числа светильников, их типа и размещения. Процесс работы программиста в таких условиях, когда естественное освещение недостаточно или отсутствует. Исходя из этого, рассчитаем параметры искусственного освещения.

Искусственное освещение выполняется посредством электрических источников света двух видов: ламп накаливания и люминесцентных ламп. Будем использовать люминесцентные лампы, которые, по сравнению с лампами накаливания, имеют существенные преимущества:

- по спектральному составу света они близки к дневному,
- естественному свету.
- обладают более высоким КПД (в 1,5-2 раза выше, чем КПД ламп накаливания).
- обладают повышенной светоотдачей (в 3-4 раза выше, чем у ламп
- накаливания).
- более длительный срок службы.

Расчет освещения производится для комнаты, длина которой 9 м, ширина - 6 м. Найдем значение освещенности данного помещения и сравним его с нормированным значением, для того, чтобы определить достаточно ли текущего искусственного освещения для работы в помещении.

Вычислим высоту подвеса светильника над рабочей поверхностью

$$H = h - h_p - h_c \quad (4.1)$$

где h_c – расстояние от светильника до перекрытия h_c = 0,05 м;

h_p – высота рабочей поверхности над полом, h_p = *0,8 м;*

h = высота помещения, h = 3 м;

H = 3 – 0,8 – 0,05 = 2,15 м.

Наиболее выгодное расстояние между светильниками определяется по формуле:

$$L = \lambda \cdot H \quad (4.2)$$

где λ = 1,2 ÷ 1,4

L = 1,3 · 2,15 = 2,795 м.

Определяем индекс помещения по формуле:

$$i = \frac{S}{H \cdot (A + B)} \quad (4.3)$$

где S – площадь помещения, S = 45 м^2;

H – расчетная высота подвеса, H=2,15 м;

A – ширина помещения, A = 6 м;

B – длина помещения, B = 9 м.

Подставив значения получим:

$$i = \frac{45}{2{,}15 \cdot (6+9)} = 1{,}495$$

Коэффициенты отражения от потолка, стен и пола равны:

$P_{пот}$ =*50%*

$P_{ст}$ = *10%*

$P_{пол}$ = *30%*

Зная индекс помещения i находим коэффициент использования η = 40%;

$K_з$– коэффициент запаса:

$K_з$ = 1,2

Формула для расчета освещенности:

$$E = \frac{N \cdot Фл \cdot \eta \cdot т}{S \cdot Кз \cdot z} \quad (4.4)$$

Для освещения выбираем люминесцентные лампы типа ЛД-65, световой поток которых $Ф_л$ =3750 Лк.

Подставим все значения в формулу (4.4) и получим:

$$E = \frac{7 \cdot 3750 \cdot 2 \cdot 0{,}4}{45 \cdot 1{,}2 \cdot 1{,}1} = 388{,}89 \text{ Лк}$$

Нормируемая освещенность составляет 300 Лк, а освещенность в помещении 294,61 Лк. Значит искусственное освещение в помещении в пределах нормы. Значит искусственное освещение в пределах нормы.

Так как L равен наиболее выгодному расстоянию между светильниками.

l – расстояние от крайних светильников или рядов светильников до стены:

$$l = 0{,}3 \div 0{,}5 \, L \qquad (4.5)$$

$l = 0{,}3 \cdot 2{,}795 = 0{,}8385$ м.

Предлагается установить в первой комнате 3 светильника, 2 в ряд, так как длина комнаты 6 метров, а ширина 4 метра. А во второй комнате 4 светильника (рисунок 4.4).

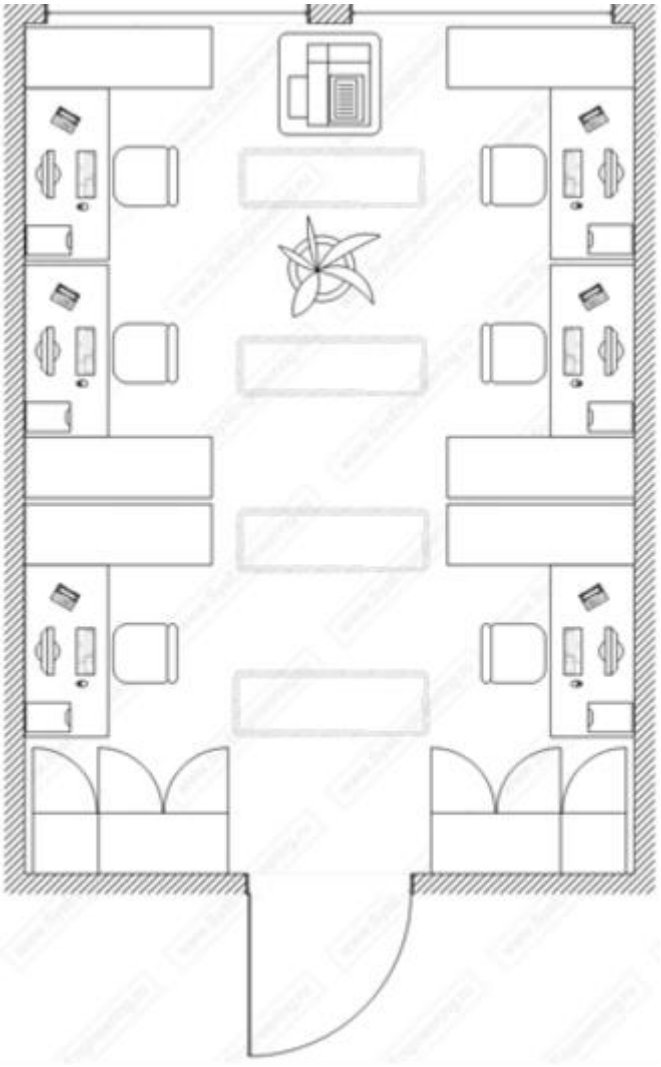


Рисунок 4.4 – Схема расположения светильников

5 Технико-экономическое обоснование

В этом разделе происходит рассмотрение экономической части реализации данного проекта. Показывает затраты:

- Финансовые;
- Трудовые;
- Временные.

5.1 Этапы реализации проекта

Проектирование и разработка приложения состоит из 5 этапов:

- постановка задач и анализ текущего положения;
- ознакомления с Фреймворками XAF и EF;
- составление функциональной спецификации;
- проектирование модели базы данных;
- разработка Windows - приложения;
- наполнение контентом;
- тестирование приложения;
- настройка оборудования и перенос на производственный сервер;
- запуск проекта.

Приведен график реализации проекта (см. таблицу 4.1).

Таблица 5.1 - Этапы реализации проекта

Перечень работ		Неделя от начала работ							
		1-2	2-3	3	4-7	8	9 -10	11	12
1 этап	Постановка задач								
	Анализ текущего положения								
	Выбор среды разработки								
2 этап	Изучение «туториалов» фреймворков								
3 этап	Составление функциональной спецификации								
4 этап	Проектирование базы данных								
5 этап	Разработка приложения								

6 этап	Наполнение контентом								
7 этап	Тестирование приложения								

Окончание таблицы 5.1

Перечень работ		Неделя от начала работ							
		1-2	2-3	3	4-7	8	9 -10	11	12
8 этап	Отладка приложения								
9 этап	Запуск проекта								

5.2 Трудовые ресурсы, используемые в работе

В работе над данным проектом задействован программист-разработчик.

Программист-разработчик – специалист, занимающийся написание программ на языке программирования, проектирование базы данных, сопровождение и инструктаж персонала сопровождения;

Количество сотрудников, задействованных в проекте, и их месячная заработная плата (см. таблицу 5.2).

Таблица 5.2 – Данные о сотрудниках

Должность	Количество	Заработная плата в месяц
Программист-разработчик	1	140000
Итого	1	140000

5.3 Техническое оборудование, использованное при разработке

Характеристики оборудования, используемого в работе, а также его стоимость (см. таблицу 5.3).

Таблица 5.3 – Перечень оборудования

Название оборудования	Характеристики	Количество	Стоимость за единицу
Acer Aspire TC-705	Intel Core i7 4770, 12 Gb DDR3, 2000 Gb HDD, Radeon R7 240	1	295500
Итого		1	295500

5.4 Программное обеспечение, используемое в работе

При разработке приложения используется следующее программное обеспечение:

- OS Windows 10 Professional – операционная система;
- Visual Studio 2015 Professional – среда для разработки;
- DevExpress – инструмент для быстрой разработки.

Программное обеспечение, использованное в работе и соответствующая ему стоимость (см. таблицу 5.4).

Таблица 5.4 – Перечень программного обеспечения

Наименование	Количество копий	Стоимость за единицу
Windows 10 Professional	1	48000
Visual Studio 2015 Professional	1	160000
DevExpress (Trial)	1	Бесплатно (30 дней)
Итого		208000

5.5 Расчет расходов на создание и реализацию проекта

Разработка Windows-приложения включает в себя кроме интеллектуальных усилий, финансовые затраты. Требует большого количества интеллектуальных затрат сотрудников, выполняющих работу, а также необходимых технических средств для ее реализации. Все это требует финансовых вложений, на основе которых высчитывается конечная стоимость проекта. Затраты на разработку данного приложения вычисляются по формуле:

$$С = ФОТ + С_н + Ф + Э + Р_{пр} + Н \quad (5.1)$$

где ФОТ – фонд оплаты труда;

$С_н$ – социальный налог;

А – амортизационные отчисления;

Э – затраты на электроэнергию;

$Р_{пр}$ – прочие расходы;

Н – накладные расходы.

5.5.1 Расходы на оплату труда разработчика

Основные затраты на реализацию программного продукта ложатся на выплату заработной платы работникам, которые рассчитываются по формуле:

$$ФОТ = З_{осн} + З_{доп} \quad (5.2)$$

где $З_{осн}$ – основная заработная плата;

$З_{доп}$ – дополнительная плата.

Для расчета затрат на основную заработную плату используются данные о среднедневном заработке и фактическом времени работы каждого сотрудника.

Среднедневной заработок:

$$Д_{ср} = \frac{ЗП_м}{Д_р} \quad (5.3)$$

где $ЗП_м$ – ежемесячный размер заработной платы;

$Д_р$– количество рабочих дней в месяце (21 день).

Программист - разработчик:

$Д_{ср} = \frac{140000}{21} = 6667$ тенге в день.

Один час работы расчитывается по формуле:

$$Н = \frac{Д_{ср}}{Ч_р} \quad (5.4)$$

где $Д_{ср}$ – средний дневной заработок работника;

$Ч_р$ – количество часов рабочего дня.

Программист - разработчик:

$Н = \frac{6667}{9} = 741$ тенге в час.

Длительность цикла в днях по каждому виду работ определяется по формуле:

$$t_n = \frac{T}{q_n * z * K} \quad (5.5)$$

где Т – трудоемкость этапа;

q_n– количество исполнителей по этапу;

z – продолжительность рабочего дня, z= 9 часов;

К= 952 К – коэффициент выполнения норм времени, К = 1,1.

Полученную величину t_n округляем в большую сторону до целых дней.

$t_1 = \frac{16}{1*8*1,1} \approx 2$ дня – Программист, постановка задач;

$t_2 = \frac{16}{1*8*1,1} \approx 2$ дня – анализ текущего положения;

$t_3 = \frac{8}{1*8*1,1} \approx 1$ дня – выбор среды разработки;

$t_4 = \frac{80}{1*8*1,1} \approx 9$ дней – изучение «туториалов» фреймворков;

$t_5 = \frac{16}{1*8*1,1} \approx 2$ дня – составление функциональной спецификации;

$t_6 = \frac{24}{1*8*1,1} \approx 3$ дня – проектирование базы данных;

$t_7 = \frac{168}{1*8*1,1} \approx 19$ дня – разработка приложения;

$t_8 = \frac{32}{1*8*1,1} \approx 4$ дня – наполнение контентом;

$t_9 = \frac{96}{1*8*1,1} \approx 11$ дней – тестирование приложения;

$t_{10} = \frac{32}{1*8*1,1} \approx 4$ дня – отладка приложения;

$t_{11} = \frac{48}{1*8*1,1} \approx 8$ дня – запуск проекта.

Сводные результаты расчета затрат на основную заработную плату работников, задействованных в разработке ПП (см. таблицу 5.5).

Таблица 5.5 – Сводные результаты расчета затрат на основную заработную плату.

№	Наименование работ	Исполнитель	Трудоемкость		Длительность цикла, день	Заработная плата за час работы, тенге	Сумма заработной плата, тенге
			Нормы – час	% от общей трудоемкости			
1	Постановка задач	Программист	16	3,0	2	741	13340
	Анализ текущего положения	Программист	16	3,0	2	741	13340
	Выбор среды разработки	Программист	8	1,5	1	741	6670
2	Изучение «туториалов» фреймворков	Программист	80	14,9	9	741	66690
3	Составление функциональной спецификации	Программист	16	3,0	2	741	13340
4	Проектирование базы данных	Программист	24	4,5	3	741	20010
5	Разработка приложения	Программист	168	31,3	19	741	146720
6	Наполнение контентом	Программист	32	6,0	4	741	26680
7	Тестирование приложения	Программист	96	17,9	11	741	80030
8	Отладка приложения	Программист	32	6,0	4	741	26680
9	Запуск проекта	Программист	48	9,0	8	741	40015

	Итого		536	100	65		453515

Таким образом, согласно произведенным расчётам основная заработная плата составляет 453515 тенге.

Дополнительная заработная плата рассчитывается по формуле:

$$З_{доп} = З_{осн} * 0,1 \quad (5.6)$$

Рассчитаем дополнительную заработную плату:

$З_{доп} = 453515 * 0,1 = 45351,5$ тенге

Фонд оплаты труда, согласно произведенным расчетам и формуле 5.2 составит:

$ФОТ = 453515 + 45351,5 = 498866,5$ тенге

5.5.2 Расходы по социальному налогу

Социальный налог составляет 11% от дохода сотрудника и рассчитывается по формуле:

$$С_н = (ФОТ - ПО) * 0,11 \quad (5.7)$$

где ПО – пенсионные отчисления.

ПО – составляют 10% от ФОТ и социальным налогом не облагаются и рассчитываются по формуле:

$$ПО = ФОТ * 0,1 \quad (5.8)$$

$ПО = 498866,5 * 0,1 = 49886,65$ тенге.

Таким образом социальный налог составит:

$С_н = (498866,5 - 49886,65) * 0,11 = 49387,8$ тенге

5.5.3 Расходы на амортизационные отчисления

Амортизационные отчисления рассчитываются по формуле:

$$A_j = \frac{H_A * С_{ПЕР} * N}{100 * 12 * n} \quad (5.9)$$

где H_A – норма амортизации;

$С_{пер}$ – первоначальная стоимость оборудования;

N – количество дней на выполнение работ;

n – количество рабочих дней в месяце.

Следовательно, амортизационные отчисления по используемому оборудованию составит:

$A_1 = \frac{40*295500*65}{100*12*21} = 30488,1$ тенге

Амортизационные отчисления по используемому программному обеспечению систовит:

$A_2 = \frac{40*208000*65}{100*12*21} = 21460,32$ тенге

Суммарные затраты на амортизацию рассчитываются по формуле:

$$A = A_1 + A_2 \quad (5.10)$$

Амортизационные отчисления составляют:

$A = 30488{,}1 + 214460{,}32 = 51948{,}42$ тенге

5.5.4 Расход на электроэнергию

Так как в процессе производства используется электрооборудование необходимо рассчитать затраты на электроэнергию. Затраты на электроэнергию для производственных нужд включают в себя расходы электроэнергии на оборудование и дополнительные нужды, рассчитываются по формуле:

$$Э = З_{эл.эн.обор.} + З_{доп.нуж.} \quad (5.11)$$

где $З_{эл.эн.обор.}$ – затраты на электроэнергию оборудования;

$З_{доп.нуж.}$ – затраты электроэнергии на дополнительные нужды.

Расходы электроэнергии на оборудование рассчитываются по формуле:

$$З_{эл.эн.обор.} = W * T * S * K_{исп} \quad (5.12)$$

где W – потребляемая мощность, Вт;

T – количество часов работы оборудования;

S – стоимость киловатт-часа электроэнергии (1кВтч = 21 тенге);

Кисп - коэффициент использования (Кисп = 0,9).

Максимальная потребляемая мощность питания «Acer Aspire TC-705 i7» находится в промежутке от 220 до 300 Вт.

Время высчитывается на основе количества рабочих дней и рабочих часов в день.

Сумма затрат на электроэнергию основного оборудования составит:

$З_{эл.эн.обор.} = ((0{,}22 + 0{,}3)/2) * 65 * 9 * 21 * 0{,}9 = 2874{,}69$ тенге

Затраты на дополнительные нужды берутся по укрупненному показателю в размере 5% от затрат на оборудование, и рассчитываются по формуле:

$$З_{доп.нуж.} = 0{,}05 * З_{эл.эн.обор.} \quad (5.13)$$

Затраты электроэнергии на дополнительные нужды:

$З_{доп.нуж.} = 0{,}05 * 2874{,}69 = 143{,}735$ тенге

Суммарные затраты на электроэнергию составляют:

$Э = 2874{,}69 + 143{,}735 = 3018{,}42$ тенге

5.5.5 Прочие расходы

Так как в процессе разработки проекта нужен офис и доступ в интернет, рассчитаем прочие расходы. Прочие расходы на разработку приложения, рассчитываются по формуле:

$$Р_{пр} = Р_{ар.пом.} + Р_{инт} \quad (5.14)$$

где $Р_{ар.пом.}$ – общая сумма затрат на аренду помещения;

$Р_{инт}$ – общая сумма затрат за интернет.

Месячная стоимость аренды в городе за 1 м^2 составило 2090 тенге, что соответствует средним показателем в городе. Площадь офиса составило 10 м^2.

Затраты на аренду офиса рассчитываются по формуле:

$$Р_{ар.пом.} = S_{ар.пом.} * S_{кв.м.} * N_{мес} \quad (5.15)$$

где $S_{ар.пом.}$ – площадь арендуемого помещения;

$S_{кв.м.}$ – стоимость аренды за 1 $м^2$ в месяц;

$N_{мес}$ – общая сумма затрат за интернет.

Рабочее помещения бралось в аренду на 3 месяца. Затраты на аренду офиса составляют:

$$P_{ар.пом.} = 15 * 2090 * 3 = 94050$$ тенге

Размер ежемесячной оплаты по тарифному плану «Интернет дома – навсегда!» от провайдера «Билайн» составила – 3500 тенге. Затраты на интернет рассчитываем по формуле:

$$P_{инт} = S_{инт.мес.} * N_{мес} \quad (5.16)$$

где $S_{инт.мес.}$ – стоимость ежемесячной оплаты за интернет.

Затраты за интернет составляют:

$$P_{инт} = 3500 * 3 = 10500$$ тенге

Прочие расходы составляют:

$$P_{пр} = 10500 + 94050 = 104550$$ тенге

5.5.6 Накладные расходы

Накладные расходы рассчитываются как 50% от всех затрат и определяются по формуле:

$$H_р = (ФОТ + C_н + А + Э + P_{пр}) * 0,5 \quad (5.17)$$

Накладный расходы:

$$H_р = (498866,5 + 49387,8 + 51948,2 + 3018,42 + 104550) * 0,5 =$$
$$= 353491,7$$ тенге

Таким образом, себестоимость разработки ПП составит:

$$C = 498866,5 + 49387,8 + 51948,42 + 3018,42 + 104550 + 353491,7 =$$
$$= 1008989$$ тенге

Сводные результаты расчета стоимости разработки приложения «KC-Drive» с применением кроссплатформенных технологий представлены в таблице (см. таблицу 5.6) и на рисунке (см. рисунок 5.1).

Таблица 5.7 – Результирующая таблица себестоимость

Наименование расходов	Сумма, тенге	В процентах от общей суммы, %
ФОТ	498866,5	49,44
Социальный налог	49387,8	4,89
Амортизационные отчисления	51948,42	5,15
Расходы на электроэнергию	3018,42	0,30
Прочие расходы	104550	10,36
Накладные расходы	301216,8	29,85
Итого	1008989	

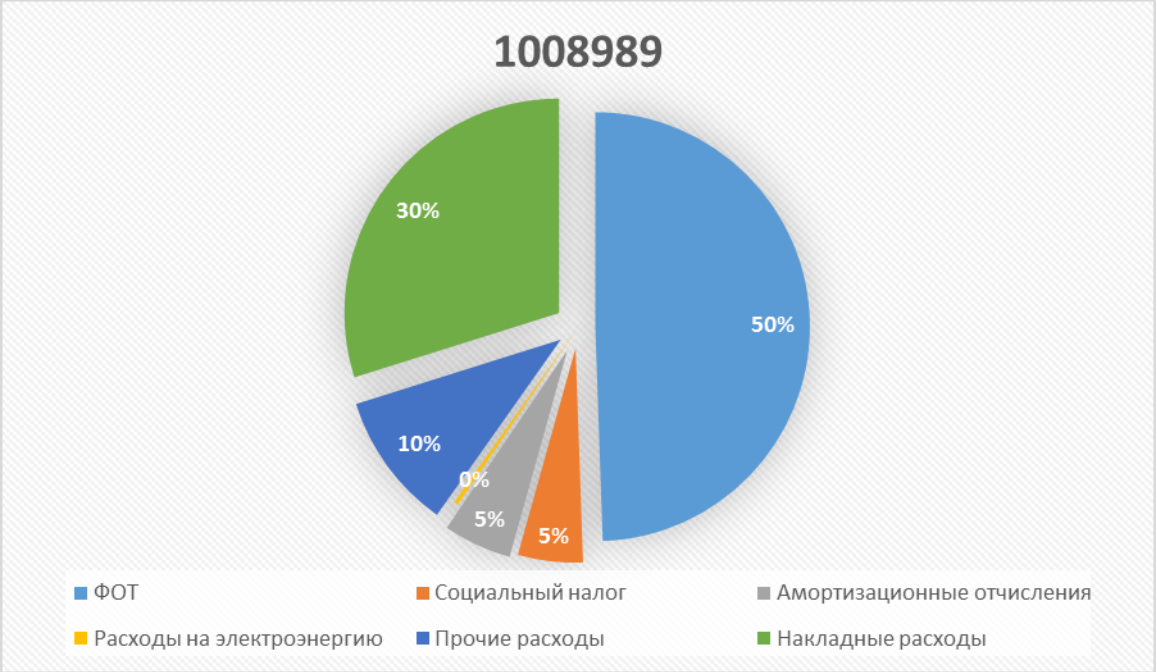


Рисунок 5.1 – Структура расходов на разработку приложения

5.6 Цена реализации проекта

Цена проекта складывается из себестоимости и желаемого чистого дохода. Минимальная цена программного продукта рассчитывается по следующей формуле:

$$Ц_п = С + П \quad (5.18)$$

где С – себестоимость разработки приложения;

П – чистый доход от программного продукта.

Для определения начальной цены используется желаемый уровень рентабельности. Для данной отрасли он составляет 40%, и рассчитывается по формуле:

$$Ц_п = С * (1 + Р/100) \quad (5.19)$$

где Р - рентабельность.

Тогда по формуле 4.16:

$Ц_п = 1008989 * (1 + 40/100) = 1412585$ тенге

Далее определяется цена реализации с учётом налога на добавленную стоимость (НДС), рассчитывается по формуле:

$$Ц_п = Ц_п + НДС \quad (5.20)$$

где НДС – налог на добавленную стоимость.

Ставка НДС на 2016 год, согласно пункту 1 статье 268 «Ставки налога на добавленную стоимость» Налогового кодекса РК составляет 12 %.

В итоге получаем цену реализации проекта:

$Ц_п = 1412585 + (1412585 * 0{,}12) = 1582095$ тенге

Заключение

Стоимость разработки приложения «KC-Drive» составила 1499500 тенге, в которую заложены все возможные затраты при разработки программного продукта.

По анализу расходов на реализацию проекта основной статьей расходов является затраты на оплату труда, более 50%.

Стоимость разработки Windows - приложения включает большие интеллектуальные и финансовые затраты. Расчёты показали, что производство и реализация данного программного продукта должны реализовывать внутри компании, для заказа данного продукта уйдут большие деньги.

Анализируя результаты внедрения приложения "KC-Drive" по обеспечению доставки экипажа на рейс, или пассажиров в гостиницу:

- обеспечило повышение эффективности использования ресурсов и качества работы
- упростило систему управления транспортными средствами и расширило функциональные ее возможности
- Оптимизировало и автоматизировало систему управления транспортными средствами
- Минимизировало ошибки и сбои за счет устранения «человеческого фактора»
- Система значительно облегчило работу всем пользователям

Список литературы

1. Методические рекомендации "Оценка теплового состояния человека с целью обоснования гигиенических требований к микроклимату рабочих мест и мерам профилактики охлаждения и перегревания" N 5168-90 от 05.03.90. В сб.: Гигиенические основы профилактики неблагоприятного воздействия производственного микроклимата на организм человека. В.43, М. 1991, с.192-211.

2. ГОСТ 12.1.005-88 "Общие санитарно-гигиенические требования к воздуху рабочей зоны".

3. СНиП РК 2.04. -05-2002

4. СН 2.2.4/2.1.8.562-96.

5. СанПиН 2.2.2.542-96

Приложение А

Листинг программного обеспечения:

```
using System;
using System.Data.Entity;
using System.Data.Common;
using DevExpress.ExpressApp.EF.Updating;
using DevExpress.Persistent.BaseImpl.EF;

namespace KC_Drive.Module.BusinessObjects
{
   public class KC_DriveDbContext : DbContext
   {
      public KC_DriveDbContext(String connectionString)
         : base(connectionString)
      {
      }
      public KC_DriveDbContext(DbConnection connection)
         : base(connection, false)
      {
      }

      public KC_DriveDbContext()
         : base("name=ConnectionString")
      {
      }
      public DbSet<ModuleInfo> ModulesInfo { get; set; }
      public DbSet<Role> Roles { get; set; }
      public DbSet<TypePermissionObject> TypePermissionObjects { get; set; }
      //public DbSet<User> Users { get; set; }
      public DbSet<KC_User> Users { get; set; }
      public DbSet<Event> Events { get; set; }
      public DbSet<Resource> Resources { get; set; }
      public DbSet<Analysis> Analysis { get; set; }
      public DbSet<ReportDataV2> ReportDataV2 { get; set; }
      public DbSet<ModelDifference> ModelDifferences { get; set; }
      public DbSet<ModelDifferenceAspect> ModelDifferenceAspects { get; set; }
      public DbSet<Car> Cars { get; set; }
      public DbSet<Repair> Repairs { get; set; }
      public DbSet<FuelFlow> FuelFlows { get; set; }
      public DbSet<Driver> Drivers { get; set; }
      public DbSet<Squad> Squads { get; set; }
      public DbSet<Passenger> Passengers { get; set; }
      public DbSet<RepairName> RepairNames { get; set; }
      public DbSet<RepairType> RepairTypes { get; set; }
      public DbSet<Location> Locations { get; set; }
      public DbSet<Shift> Shifts { get; set; }
   }
```

```
}
using DevExpress.ExpressApp.ConditionalAppearance;
using DevExpress.ExpressApp.DC;
using DevExpress.ExpressApp.Model;
using DevExpress.ExpressApp.SystemModule;
using DevExpress.Persistent.Base;
using DevExpress.Persistent.Validation;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.ComponentModel.DataAnnotations;

namespace KC_Drive.Module.BusinessObjects
{
   [DefaultClassOptions]
   [NavigationItem("Car")]
   [ListViewFilter("All Cars", "")]
   [ListViewFilter("Free", "Status = 'Free'")]
   [ListViewFilter("On the road", "Status = 'OnTheRoad'")]
   [ListViewFilter("On the repair", "Status = 'OnTheRepair'")]
   [Appearance("Car_City", AppearanceItemType = "ViewItem", TargetItems = "*",
      Criteria = "LP100 > 30 || (LP100 > 0 && LP100 < 12)", Context = "ListView")]
   public class Car
   {
      public Car()
      {
         Repairs = new List<Repair>();
      }
      [Browsable(false)]
      public Int32 CarID { get; protected set; }
      [RuleRequiredField("Car_Brand", DefaultContexts.Save, "Please enter the brand of car!")]
      [StringLength(35)]
      public string Brand { get; set; }
      [VisibleInListView(false)]
      [VisibleInLookupListView(false)]
      [RuleValueComparison("Car_CostCenter",                    DefaultContexts.Save,
ValueComparisonType.GreaterThan, 0)]
      public int CostCenter { get; set; }
      [VisibleInListView(false)]
      [VisibleInLookupListView(false)]
      [RuleRequiredField("Car_Engine", DefaultContexts.Save, "Please enter the car engine!")]
      [StringLength(35)]
      public string Engine { get; set; }
      [VisibleInListView(false)]
      [VisibleInLookupListView(false)]
      [StringLength(35)]
      public string Chassis { get; set; }
      [VisibleInListView(false)]
      [VisibleInLookupListView(false)]
      [RuleRequiredField("Car_Body", DefaultContexts.Save, "Please enter the car body!")]
      [StringLength(35)]
```

```
        public string Body { get; set; }
        [VisibleInListView(false)]
        [VisibleInLookupListView(false)]
        [RuleValueComparison("Car_Year", DefaultContexts.Save,
ValueComparisonType.GreaterThan, 0)]
        public int Year { get; set; }
        [VisibleInListView(false)]
        [VisibleInLookupListView(false)]
        [RuleValueComparison("Car_Mileage", DefaultContexts.Save,
ValueComparisonType.GreaterThan, 0)]
        public int Mileage { get; set; }
        [VisibleInListView(false)]
        [VisibleInLookupListView(false)]
        [ModelDefault("EditMask", "F")]
        [ModelDefault("DisplayFormat", "{0:F}")]
        [RuleRequiredField("Please enter the mileage date", DefaultContexts.Save)]
        public DateTime MileageDate { get; set; }
        [RuleRequiredField("Car_RegistrationNumber", DefaultContexts.Save, "Please enter the car
registration  number!")]
        [StringLength(35)]
        public string RegistrationNumber { get; set; }
        [VisibleInListView(false)]
        [VisibleInLookupListView(false)]
        [RuleRequiredField("Car_Color", DefaultContexts.Save, "Please enter the car color!")]
        [StringLength(35)]
        public string Color { get; set; }
        [VisibleInListView(false)]
        [VisibleInLookupListView(false)]
        [RuleRequiredField("Please enter your date of inspection", DefaultContexts.Save)]
        public DateTime Inspection { get; set; }
        [VisibleInListView(false)]
        [VisibleInLookupListView(false)]
        [RuleRequiredField("Please enter your date of insurance", DefaultContexts.Save)]
        public DateTime Insurance { get; set; }
        [VisibleInListView(false)]
        [VisibleInLookupListView(false)]
        [DisplayName("Certificat D'Immatriculation")]
        [RuleRequiredField("Car_NumberOfRegistrationCertificate", DefaultContexts.Save, "Please
enter the number of registration certificate car!")]
        [StringLength(35)]
        public string CertificatImmatriculation { get; set; }
        [VisibleInListView(false)]
        [VisibleInLookupListView(false)]
        [RuleRequiredField("Car_IssuedBy", DefaultContexts.Save, "Please enter the issued by of
registration certificate!")]
        [StringLength(35)]
        public string IssuedBy { get; set; }
        [VisibleInListView(false)]
        [VisibleInLookupListView(false)]
        [RuleValueComparison("Car_IssueDate", DefaultContexts.Save,
ValueComparisonType.GreaterThan, "01/01/1900", SkipNullOrEmptyValues = false)]
```

```
        public DateTime DateIssued { get; set; }
        [VisibleInListView(false)]
        [VisibleInLookupListView(false)]
        [RuleRequiredField("Car_Accounting", DefaultContexts.Save, "Please enter the car accounting!")]
        [StringLength(35)]
        public string Accounting { get; set; }
        [VisibleInListView(false)]
        [VisibleInLookupListView(false)]
        [RuleRequiredField("Car_Location", DefaultContexts.Save, "Please enter the car location!")]
        public virtual Location Location { get; set; }
        [VisibleInListView(false)]
        [VisibleInLookupListView(false)]
        [CaptionsForBoolValues("Yes", "No")]
        public bool TemporaryAccounting { get; set; }
        [VisibleInListView(false)]
        [VisibleInLookupListView(false)]
        [CaptionsForBoolValues("Diesel", "Petrol")]
        public bool FuelTank { get; set; }
        [VisibleInListView(false)]
        [VisibleInLookupListView(false)]
        [Appearance("ServiceHoursEnable", Enabled = false, Criteria = "Type != 'SpecialVehicle'", Context = "DetailView")]
        public Nullable<int> ServiceHours { get; set; }
        [VisibleInListView(false)]
        [VisibleInLookupListView(false)]
        [ModelDefault("EditMask", "F2")]
        [ModelDefault("DisplayFormat", "{0:F2}")]
        [RuleValueComparison("ReportFuel_WinterNorm", DefaultContexts.Save, ValueComparisonType.GreaterThan, 0)]
        public decimal WinterNorm { get; set; }
        [VisibleInListView(false)]
        [VisibleInLookupListView(false)]
        [ModelDefault("EditMask", "F2")]
        [ModelDefault("DisplayFormat", "{0:F2}")]
        [RuleValueComparison("ReportFuel_SummerNorm", DefaultContexts.Save, ValueComparisonType.GreaterThan, 0)]
        public decimal SummerNorm { get; set; }
        public CarType Type { get; set; }
        [VisibleInLookupListView(false)]
        public Status Status { get; set; }
        [Aggregated]
        public virtual IList<Repair> Repairs { get; set; }
        public override string ToString()
        {
            return Brand + " " + RegistrationNumber;
        }
    }
    public enum Status
    {
        [Display(Name = "Free")]
```

```
        Free,
        [Display(Name = "On the road")]
        OnTheRoad,
        [Display(Name = "On the repair")]
        OnTheRepair
    }

    public enum CarType
    {
        Car,
        Bus,
        Minibus,
        [Display(Name = "Special vehicle")]
        SpecialVehicle
    }
}
```