АВТОНОМНАЯ НЕКОММЕРЧЕСКАЯ ОРГАНИЗАЦИЯ
ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
ОКТЯБРЬСКИЙ ЭКОНОМИЧЕСКИЙ ТЕХНИКУМ

ОТЧЕТ

по производственной практике

ПМ.03. Участие в интеграции программных модулей

Выполнил
студент группы 4ПР1-13 Л.З. Каримов
Принял
преподаватель А.Ю. Рамазанова

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

История ОЗНА началась в начале 1950-х годов, в период послевоенного восстановления народного хозяйства и бурного развития нефтяной промышленности СССР. В марте 1953 года в г. Октябрьском (Башкирия) был построен ремонтно-механический завод, ставший основой Компании. Его продукция была востребована на нефтепромыслах республики, где шла интенсивная добыча черного золота.

В январе 1958 года в Октябрьском построен завод по производству приборов и средств автоматизации и диспетчеризации «Нефтеавтоматика». Эти два предприятия уверенно заняли положение лидеров в своей отрасли.

В 1950-1960 гг. оборудование ОЗНА поставлялось преимущественно нефтяникам Башкирии, показывавшим самый значительный рост нефтедобычи в стране, за что республика была удостоена почётного наименования «второе Баку».

С 1970-х гг. Компания начала серийные поставки блочных кустовых и нефтеперекачивающих насосных станций, блоков дозирования реагентов, замерных установок и другого нефтепромыслового оборудования. В число заказчиков продукции ОЗНА, помимо отечественных нефтяников, вошли предприятия стран Совета экономической взаимопомощи (СЭВ): Болгарии, Румынии, Югославии.

Трансформации 1990-х годов потребовали новых подходов к организации деятельности: 1990 год - создано арендное предприятие (АП) «ОЗАО и П»; 1991 год - внедрена блочная система управления производством; 1992 год - принято решение о приватизации путем акционирования; 1993 год - на базе АП «ОЗАО и П» образовано «Акционерное общество открытого типа «ОЗНА». 12 июля 1996 года создано ОАО «Акционерная компания ОЗНА».

Главной целью производственной (по профилю специальности) практики

является закрепление и совершенствование приобретенных в процессе обучения профессиональных умений обучающихся по изучаемой специальности, развитие общих и профессиональных компетенций, освоение современных производственных процессов, адаптация обучающихся к конкретным условиям деятельности организация различных организационно-правовых форм.

В результате прохождения производственной (по профилю специальности) практики в рамках профессионального модуля обучающийся должен приобрести практический опыт работы:

- с проектной и технической документацией на уровне взаимодействия компонент программного обеспечения;
- выполнения интеграции модулей в программную среду;
- выполнения отладки программного продукта с использованием специализированных программных средств;
- разработки текстовых наборов и текстовых сценариев;
- проведения инспектирования компонент программного продукта на предмет соответствия стандартам кодирования.

1. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

1.1 Жизненный цикл программного продукта

Жизненный цикл программного продукта - период времени, который начинается с момента принятия решения о необходимости создания программного продукта и заканчивается в момент его полного изъятия из эксплуатации.

Стандарт ГОСТ 34.601-90 предусматривает следующие стадии и этапы создания автоматизированной системы:

1. Формирование требований к автоматизированной системе

1.1. Обследование объекта и обоснование необходимости создания автоматизированной системы

1.2. Формирование требований пользователя к автоматизированной системе

.3. Оформление отчета о выполнении работ и заявки на разработку автоматизированной системы

2. Разработка концепции автоматизированной системы

2.1. Изучение объекта

2.2. Проведение необходимых научно-исследовательских работ

.3. Разработка вариантов концепции автоматизированной системы и выбор варианта концепции автоматизированной системы, удовлетворяющего требованиям пользователей

.4. Оформление отчета о проделанной работе

3. Техническое задание

3.1. Разработка и утверждение технического задания на создание автоматизированной системы

4. Эскизный проект

4.1. Разработка предварительных проектных решений по системе и её частям

4.2. Разработка документации на автоматизированную систему и её части

5. Технический проект

5.1. Разработка проектных решений по системе и её частям

5.2. Разработка документации на автоматизированную систему и её части

.3. Разработка и оформление документации на поставку комплектующих изделий

.4. Разработка заданий на проектирование в смежных частях проекта

6. Рабочая документация

6.1. Разработка рабочей документации на автоматизированную систему и её части

6.2. Разработка и адаптация программ

7. Ввод в действие

7.1. Подготовка объекта автоматизации

7.2. Подготовка персонала

.3. Комплектация автоматизированной системы поставляемыми изделиями (программными и техническими средствами, программно-техническими комплексами, информационными изделиями)

.4. Строительно-монтажные работы

.5. Пусконаладочные работы

.6. Проведение предварительных испытаний

.7. Проведение опытной эксплуатации

.8. Проведение приёмочных испытаний

8. Сопровождение автоматизированной системы

8.1. Выполнение работ в соответствии с гарантийными обязательствами

8.2. Послегарантийное обслуживание

Эскизный, технический проекты и рабочая документация - это последовательное построение все более точных проектных решений. Допускается исключать стадию «Эскизный проект» и отдельные этапы работ на всех стадиях, объединять стадии «Технический проект» и «Рабочая документация» в «Технорабочий проект», параллельно выполнять различные этапы и работы, включать дополнительные.

1.2 Основные модели процесса разработки программного обеспечения

Модель кодирования и устранения ошибок.

Совершенно простая модель, характерная для студентов ВУЗов. Именно по этой модели большинство студентов разрабатывают лабораторные работы.

Данная модель имеет следующий алгоритм:

- постановка задачи;
- выполнение;
- проверка результата;
- при необходимости переход к первому пункту.

Модель ужасно устаревшая и характерна для 1960-1970 гг., поэтому преимуществ перед следующими моделями практически не имеет, а недостатки на лицо.

Каскадная модель жизненного цикла программного обеспечения представлена на рисунке 1.

Рисунок 1 Каскадная модель жизненного цикла программного обеспечения

Преимущества:

- последовательное выполнение этапов проекта в строгом фиксированном порядке;
- позволяет оценивать качество продукта на каждом этапе.

Недостатки:

- отсутствие обратных связей между этапами;
- не соответствует реальным условиям разработки программного продукта.

Каскадная модель с промежуточным контролем (водоворот).

Данная модель является почти эквивалентной по алгоритму предыдущей модели, однако при этом имеет обратные связи с каждым этапом жизненного цикла, при этом порождает очень весомый недостаток: 10-ти кратное увеличение затрат на разработку.модель (разработка через тестирование).

Данная модель имеет более приближенный к современным методам алгоритм, однако все еще имеет ряд недостатков. Является одной из основных практик экстремального программирования. Изображение представлено на рисунке 2.

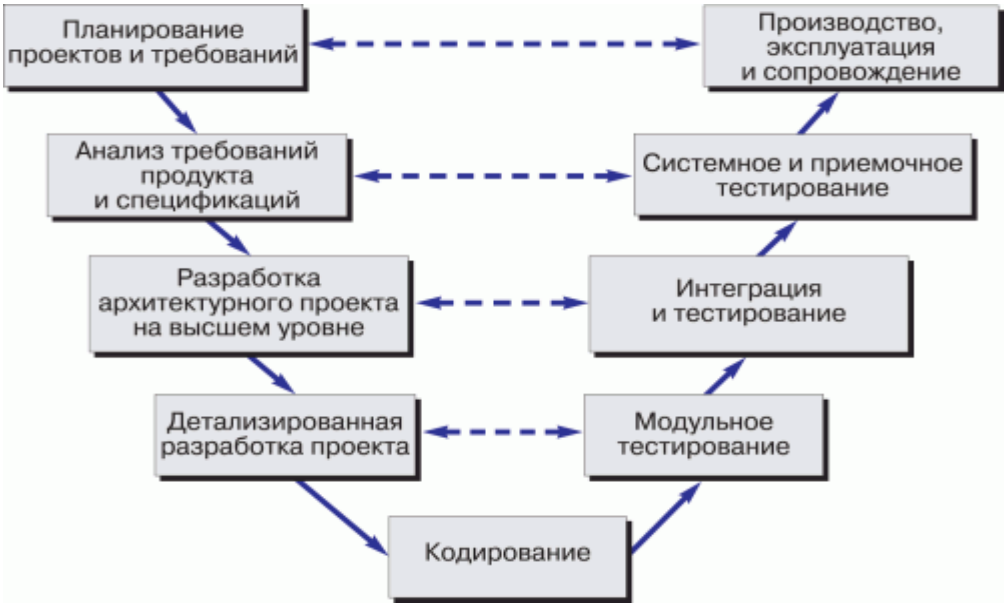

Рисунок 2 V модель

Модель на основе разработки прототипа.

Прототипирование используется на ранних стадиях жизненного цикла программного обеспечения:

- Прояснить не ясные требования;
- Выбрать одно из ряда концептуальных решений;
- Проанализировать осуществимость проекта.

Классификация протопипов:

- Горизонтальные и вертикальные;
- Одноразовые и эволюционные;
- бумажные и раскадровки.

Горизонтальные прототипы - моделирует исключительно UI не затрагивая логику обработки и базу данных.

Вертикальные прототипы - проверка архитектурных решений.

Одноразовые прототипы - для быстрой разработки.

Эволюционные прототипы - первое приближение эволюционной системы.

Спиральная модель жизненного цикла программного обеспечения.

Спиральная модель представляет собой процесс разработки программного обеспечения, сочетающий в себе как проектирование, так и постадийное прототипирование с целью сочетания преимуществ восходящей и нисходящей концепции.

Преимущества:

Быстрое получение результата

Повышение конкурентоспособности

Изменяющиеся требования - не проблема

Недостатки:

Отсутствие регламентации стадий

Изображение модели представлено на рисунке 3.

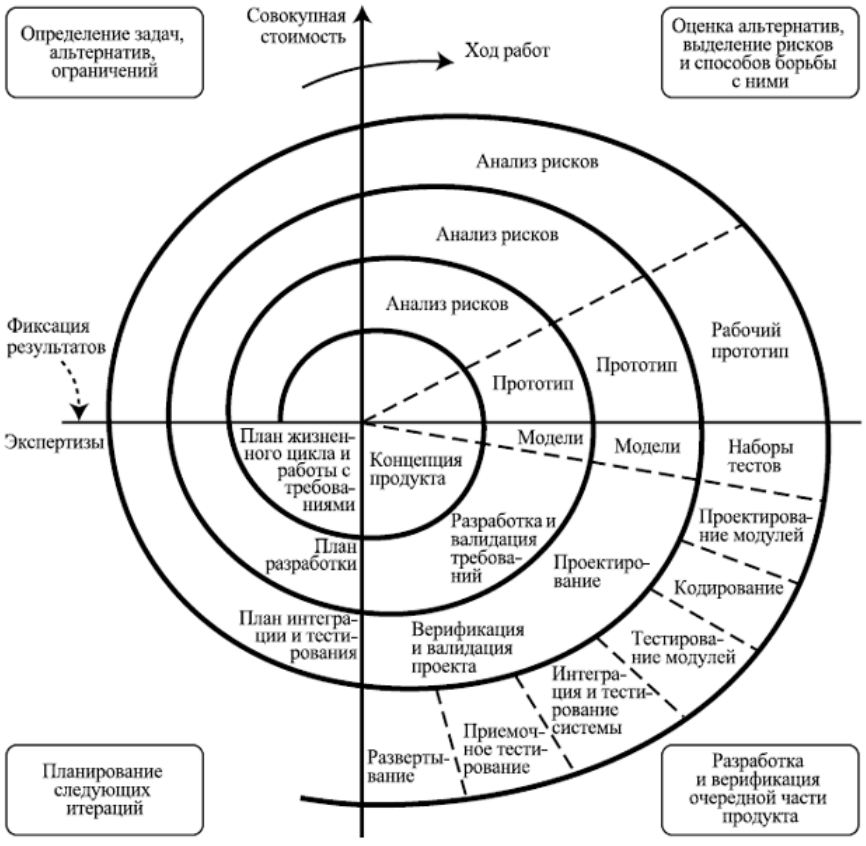

Рисунок 3 Спиральная модель жизненного цикла

1.3 Организация процесса разработки программного обеспечения

Maturity Model - модель зрелости возможностей (модель полноты потенциала) создания программного обеспечения: эволюционная модель развития способности компании разрабатывать программное обеспечение.

В ноябре 1986 года американский институт Software Engineering Institute (SEI) совместно с Mitre Corporation начали разработку обзора зрелости процессов разработки программного обеспечения, который был предназначен для помощи в улучшении их внутренних процессов.

Разработка такого обзора была вызвана запросом американского федерального правительства на предоставление метода оценки субподрядчиков для разработки программного обеспечения. Реальная же проблема состояла в неспособности управлять большими проектами. Во многих компаниях проекты выполнялись со значительным опозданием и с превышением запланированного бюджета. Необходимо было найти решение данной проблемы.

В сентябре 1987 года SEI выпустил краткий обзор процессов разработки программного обеспечения с описанием их уровней зрелости, а также опросник, предназначавшийся для выявления областей в компании, в которых были необходимы улучшения. Однако, большинство компаний рассматривало данный опросник в качестве готовой модели, вследствие чего через 4 года вопросник был преобразован в реальную модель, Capability Maturity Model for Software (CMM). Первая версия CMM (Version 1.0), вышедшая в 1991 году, в 1992 году была пересмотрена участниками рабочей встречи, в которой принимали участие около 200 специалистов в области программного обеспечения, и членами общества разработчиков.

Использование модели на практике выявило неоднозначность в подходах к достижению более высоких уровней организации процессов разработки программного обеспечения. Поэтому к 2002 году разрабатываются рекомендации по улучшению процесса разработки, которые получают название

CMMI (Capability Maturity Model Integration).

.4 Проектирование и разработка программного обеспечения

Проектирование программного обеспечения - процесс создания проекта программного обеспечения, а также дисциплина, изучающая методы проектирования. Проектирование программного обеспечения является частным случаем проектирования продуктов и процессов.

Целью проектирования является определение внутренних свойств системы и детализации её внешних (видимых) свойств на основе выданных заказчиком требований к программному обеспечению (исходные условия задачи). Эти требования подвергаются анализу.

Первоначально программа рассматривается как чёрный ящик. Ход процесса проектирования и его результаты зависят не только от состава требований, но и выбранной модели процесса, опыта проектировщика.

Модель предметной области накладывает ограничения на бизнес-логику и структуры данных.

В зависимости от класса создаваемого программного обеспечения, процесс проектирования может обеспечиваться как «ручным» проектированием, так и различными средствами его автоматизации. В процессе проектирования программного обеспечения для выражения его характеристик используются различные нотации - блок-схемы, ER-диаграммы, UML-диаграммы, DFD-диаграммы, а также макеты.

Проектированию обычно подлежат:

- Архитектура программного обеспечения;
- Устройство компонентов программного обеспечения;
- Пользовательские интерфейсы.

В российской практике проектирование ведется поэтапно в соответствии

со стадиями, регламентированными ГОСТ 2.103-68:

- техническое задание(по ГОСТ 2.103-68 к стадиям разработки не относится);
- техническое предложение;
- эскизный проект;
- технический проект;
- рабочий проект.

На каждом из этапов формируется свой комплект документов, называемый проектом (проектной документацией).

В зарубежной практике регламентирующими документами, например, являются Software Architecture Document, Software Design Document.

Разработка программного обеспечения (англ. software development) - деятельность по созданию нового программного обеспечения.

Разработка программного обеспечения как инженерная дисциплина является составной частью (областью) программной инженерии, наряду с дисциплинами, отвечающими за функционирование и сопровождение программных продуктов.

1.5 Интеграция системы

Интеграция информационных систем - это процесс установки связей между информационными системами предприятий и организаций для получения единого информационного пространства и организации поддержки сквозных бизнес-процессов предприятий и организаций.

Задача интеграции информационных и учетных систем состоит из двух взаимосвязанных частей: интеграция приложений и интеграция данных. Без интеграции данных невозможно провести интеграцию приложений.

Интеграция данных - процесс компоновки информации из различных

информационных систем предприятий и организаций, установки ее однозначного соответствия в разных системах, синхронизация одинаковых информационных объектов в различных информационных систем.

Решая задачу интеграции данных, компания должна провести унификацию и стандартизацию нормативно-справочной информации. Нормативно-справочная информация - условно-постоянная составляющая общей корпоративной информации. Эта информация используется при регламентации деятельности компании, она обеспечивает «сшивку» данных, сопровождающих бизнес-процессы компании. Другими словами, нормативно-справочная информация - это набор справочников, словарей, классификаторов, стандартов, регламентов, используемых в деятельности компании. Нормативно-справочная информация является ядром информационного пространства компании. Наличие однозначной, структурированной, стандартизированной Нормативно-справочной информации, управление которой ведется в соответствии с продуманными правилами и алгоритмами, - базис, обязательное условие создания эффективных интеграционных решений.

Интеграция приложений - процесс организации и настройки взаимодействия информационных систем. Для многих крупных компаний наилучшим выбором становится создание композитного приложения с максимальным сохранением существующего программного обеспечения и технологий, т.е. реализация интеграции информационных систем с помощью сервисной шины предприятия (Enterprise Service Bus). Интеграция приложений с использованием Enterprise Service Bus - действенный инструмент для создания единого информационного пространства и организации надежного информационного обмена между всеми автоматизированными системами учета и управления в компании.

.6 Среды разработки приложений

Интегрированная среда разработки, ИСР/IDE (англ. Integrated development environment) - комплекс программных средств, используемый программистами для разработки программного обеспечения.

Первые IDE были созданы для работы через консоль или терминал, которые сами по себе были новинкой: до того программы создавались на бумаге, вводились в машину с помощью предварительно подготовленных бумажных носителей (перфокарт, перфолент) и т. д.BASIC был первым языком, который был создан с IDE, и был также первым, который был разработан для использования в консоли или терминале. Эта IDE (часть Dartmouth Time Sharing System) управлялась при помощи команд, поэтому существенно отличалась от более поздних, управляемых с помощью меню и горячих клавиш, и тем более графических IDE, распространённых в XXI веке. Однако она позволяла редактировать исходный код, управлять файлами, компилировать, отлаживать и выполнять программы способом, принципиально подобным современным IDE.I - продукт от Softlab Munich, был первой в мире интегрированной средой разработки для программного обеспечения в 1975 г.[2] и, возможно, мировым лидером в этой рыночной нише в течение 1970-х и 1980-х годов. Он был установлен у 22000 программистов во всем мире. До 1989 года 6000 копий было установлено в Федеративной Республике Германия. Ныне Maestro I принадлежит истории и может быть найден разве что в Музее Информационной технологии в Арлингтоне.

Одной из первых IDE с возможностью подключения плагинов была Softbench.

Среда разработки включает в себя:

- текстовый редактор;
- компилятор и/или интерпретатор;
- средства автоматизации сборки;

- отладчик.

Иногда содержит также средства для интеграции с системами управления версиями и разнообразные инструменты для упрощения конструирования графического интерфейса пользователя. Многие современные среды разработки также включают браузер классов, инспектор объектов и диаграмму иерархии классов - для использования при объектно-ориентированной разработке программного обеспечения. IDE обычно предназначены для нескольких языков программирования - такие как IntelliJ IDEA, NetBeans, Eclipse, Qt Creator, Geany, Embarcadero RAD Studio, Code::Blocks, Xcode или Microsoft Visual Studio, но есть и IDE для одного определённого языка программирования - как, например, Visual Basic, Delphi, Dev-C++.

Частный случай IDE - среды визуальной разработки, которые включают в себя возможность визуального редактирования интерфейса программы.

.7 Язык SQL

SQL (Structured Query Language - Структурированный язык запросов) - язык управления базами данных для реляционных баз данных. Сам по себе SQL не является Тьюринг-полным языком программирования, но его стандарт позволяет создавать для него процедурные расширения, которые расширяют его функциональность до полноценного языка программирования.

Язык был создан в 1970х годах под названием “SEQUEL” для системы управления базами данных System R. Позднее он был переименован в “SQL” во избежание конфликта торговых марок. В 1979 году SQL был впервые опубликован в виде коммерческого продукта Oracle V2.

Первый официальный стандарт языка был принят ANSI в 1986 году и ISO - в 1987. С тех пор были созданы еще несколько версий стандарта, некоторые из них повторяли предыдущие с незначительными вариациями, другие принимали

новые существенные черты.

Несмотря на существование стандартов, большинство распространенных реализаций SQL отличаются так сильно, что код редко может быть перенесен из одной системы управления базами данных в другую без внесения существенных изменений. Это объясняется большим объемом и сложностью стандарта, а также нехваткой в нем спецификаций в некоторых важных областях реализации.создавался как простой стандартизированный способ извлечения и управления данными, содержащимися в реляционной базе данных. Позднее он стал сложнее, чем задумывался, и превратился в инструмент разработчика, а не конечного пользователя. В настоящее время SQL (по большей части в реализации Oracle) остается самым популярным из языков управления базами данных, хотя и существует ряд альтернатив.состоит из четырех отдельных частей:

1. Язык определения данных (DDL) используется для определения структур данных, хранящихся в базе данных. Операторы DDL позволяют создавать, изменять и удалять отдельные объекты в базе данных. Допустимые типы объектов зависят от используемой системы управления базами данных и обычно включают базы данных, пользователей, таблицы и ряд более мелких вспомогательных объектов, например, роли и индексы.

Основными его командами являются:

- CREATE DATABASE (создать базу данных);
- CREATE TABLE (создать таблицу);
- CREATE VIEW (создать виртуальную таблицу);
- CREATE INDEX (создать индекс);
- CREATE TRIGGER (создать триггер);
- CREATE PROCEDURE (создать сохраненную процедуру);
- ALTER DATABASE (модифицировать базу данных);
- ALTER TABLE (модифицировать таблицу);

- ALTER VIEW (модифицировать виртуальную таблицу);
- ALTER INDEX (модифицировать индекс);
- ALTER TRIGGER (модифицировать триггер);
- ALTER PROCEDURE (модифицировать сохраненную процедуру);
- DROP DATABASE (удалить базу данных);
- DROP TABLE (удалить таблицу);
- DROP VIEW (удалить виртуальную таблицу);
- DROP INDEX (удалить индекс);
- DROP TRIGGER (удалить триггер);
- DROP PROCEDURE (удалить сохраненную процедуру.

2. Язык манипуляции данными (DML) используется для извлечения и изменения данных в базе данных. Операторы DML позволяют извлекать, вставлять, изменять и удалять данные в таблицах. Иногда операторы select извлечения данных не рассматриваются как часть DML, поскольку они не изменяют состояние данных. Все операторы DML носят декларативный характер.

Основными его командами являются:

- SELECT (выбрать);
- INSERT (вставить);
- UPDATE (обновить);
- DELETE (удалить).

3. Язык определения доступа к данным (DCL) используется для контроля доступа к данным в базе данных. Операторы DCL применяются к привилегиям и позволяют выдавать и отбирать права на применение определенных операторов DDL и DML к определенным объектам базы данных.

Основными его командами являются:

- GRANT (дать права)
- REVOKE (забрать права)

4. Язык управления транзакциями (TCL) используется для контроля обработки транзакций в базе данных. Обычно операторы TCL включают commit для подтверждения изменений, сделанных в ходе транзакции, rollback для их отмены и savepoint для разбиения транзакции на несколько меньших частей.

SQL реализует декларативную парадигму программирования: каждый оператор описывает только необходимое действие, а система управления базами данных принимает решение о том, как его выполнить, т.е. планирует элементарные операции, необходимые для выполнения действия и выполняет их. Тем не менее, для эффективного использования возможностей SQL разработчику необходимо понимать то, как система управления базами данных анализирует каждый оператор и создает его план выполнения.

1.8 Защита информации в базах данных

В настоящее время объём информации в мире настолько велик, что самым оптимальным методом работы с ней является база данных. База данных - это представленная в объективной форме совокупность материалов, систематизированных так, чтобы эти материалы могли быть найдены и обработаны с помощью компьютера. Её защита является одной из самых сложных задач на сегодняшний день.

Угрозы потери конфиденциальной информации стали обычным явлением, и если в системе защиты есть недостатки, то ценные данные могут оказаться в руках третьих лиц. Каждый сбой работы базы данных может парализовать работу целых корпораций, фирм, что приведет к весомым материальным потерям.

Методы защиты баз данных в различных системах управления базами данных условно делятся на две группы (анализ современных фирм Borland и Microsoft): основные и дополнительные.

К основным средствам защиты относится:

- защита паролем;
- шифрование;
- разделение прав доступа к объектам базы данных;
- защита полей и записей таблиц базы данных.

Защита паролем - это самый простой способ защиты базы данных от несанкционированного доступа.

Пароли устанавливаются пользователями или администраторами. Их учет и хранение выполняется системой управления базой данных. Пароли хранятся в специальных файлах системы управления базы данных в шифрованном виде. После ввода пароля пользователю предоставляется доступ к требуемой информации.

Несмотря на простоту парольной защиты, у неё имеется ряд недостатков. Во-первых, пароль уязвим, особенно если он не шифруется при хранении в системе управления базой данных. Во-вторых, пользователю надо запоминать или записать пароль, а при небрежном отношении к записям пароль может стать достоянием других.

Более мощным средством защиты данных является шифрование. Шифрование - это процесс перевода информации по определенному алгоритму в вид непригодный для чтения, в целях защиты от несанкционированного просмотра или использования. Важной особенностью любого алгоритма шифрования является использование ключа, который утверждает выбор конкретного метода кодирования из всех возможных. В основном применяется для защиты уязвимых данных.

Шифрование обеспечивает три состояния безопасности информации:

- конфиденциальность;
- целостность;
- идентифицируемость.

В целях контроля использования основных ресурсов системы управления базы данных во многих системах имеются средства установления прав доступа к объектам базы данных. Права доступа определяют возможные действия над объектами. Владелец объекта, а также администратор базы данных имеют все права. Остальные пользователи имеют те права и уровни доступа к объектам, которыми их наделили.

Разрешение на доступ к конкретным объектам базы данных сохраняется в файле рабочей группы.

Файл рабочей группы содержит данные о пользователях группы и считывается во время запуска. Файл содержит следующую информацию: имена учетных записей пользователей, пароли пользователей, имена групп, в которые входят пользователи.

К дополнительным средствам защиты баз данных можно отнести следующие средства:

- встроенные средства контроля значений данных в соответствии с типами:
- повышение достоверности вводимых данных:
- обеспечения целостности связей таблиц:
- организации совместного использования объектов базы данных в сети.

Описанные выше методы и способы являются основополагающими, однако их использование не гарантирует полной сохранности данных. Для повышения уровня безопасности информации в базе данных рекомендуется использование комплексных мер.

1.9 Стандартизация защищенности программ

Общие критерии оценки защищённости информационных технологий,

Общие критерии, ОК (англ. Common Criteria for Information Technology Security Evaluation, Common Criteria, CC) - российский и международный стандарт по компьютерной безопасности. В отличие от стандарта FIPS 140, Common Criteria не приводит списка требований по безопасности или списка особенностей, которые должен содержать продукт. Вместо этого он описывает инфраструктуру (framework), в которой потребители компьютерной системы могут описать требования, разработчики могут заявить о свойствах безопасности продуктов, а эксперты по безопасности определить, удовлетворяет ли продукт заявлениям. Таким образом, Common Criteria позволяет обеспечить условия, в которых процесс описания, разработки и проверки продукта будет произведён с необходимой скрупулёзностью.

Стандарт содержит два основных вида требований безопасности: функциональные, предъявляемые к функциям безопасности и реализующим их механизмам, и требования доверия, предъявляемые к технологии и процессу разработки и эксплуатации.

Функциональные требования сгруппированы на основе выполняемой ими роли или обслуживаемой цели безопасности, всего 11 функциональных классов (в трёх группах), 66 семейств, 135 компонентов.

Первая группа определяет элементарные сервисы безопасности:

- FAU - аудит, безопасность (требования к сервису, протоколирование и аудит);
- FIA - идентификация и аутентификация;
- FRU - использование ресурсов (для обеспечения отказоустойчивости).

Вторая группа описывает производные сервисы, реализованные на базе элементарных:

- FCO - связь (безопасность коммуникаций отправитель-получатель);
- FPR - приватность;

- FDP - защита данных пользователя;
- FPT - защита функций безопасности объекта оценки.

Третья группа классов связана с инфраструктурой объекта оценки:

- FCS - криптографическая поддержка (обслуживает управление криптоключами и крипто-операциями);
- FMT - управление безопасностью;
- FTA - доступ к объекту оценки (управление сеансами работы пользователей);
- FTP - доверенный маршрут/канал;

Требования гарантии безопасности (доверия) - требования, предъявляемые к технологии и процессу разработки и эксплуатации объекта оценки. Разделены на 10 классов, 44 семейства, 93 компонента, которые охватывают различные этапы жизненного цикла.

Первая группа содержит классы требований, предшествующих разработке и оценке объекта:

- APE - оценка профиля защиты;
- ASE - оценка задания по безопасности.

Вторая группа связана с этапами жизненного цикла объекта аттестации:

- ADV - разработка, проектирование объекта;
- ALC - поддержка жизненного цикла;
- ACM - управление конфигурацией;
- AGD - руководство администратора и пользователя;
- ATE - тестирование;
- AVA - оценка уязвимостей;
- ADO - требования к поставке и эксплуатации;
- AMA - поддержка доверия-требования, применяется после сертификации объекта на соответствие общим критериям.

1.10 Сертификация и порядок её проведения

Основной целью сертификации технологий проектирования и производства систем и программных средств является защита интересов пользователей, государственных и ведомственных интересов на основе контроля качества продукции, обеспечения их высоких потребительских свойств, повышения эффективности затрат в сфере их производства, эксплуатации и сопровождения, повышения объективности оценок характеристик и обеспечения конкурентоспособности конечного продукта. Проведение сертификации систем качества предприятия обычно планируется для достижения одной или нескольких целей:

- определения соответствия или несоответствия элементов системы качества установленным требованиям производства;
- определения эффективности внедренной системы качества предприятия с точки зрения соответствия поставленным целям для
- обеспечения качества продукции;
- обеспечения возможности предприятию улучшить свою систему качества;
- определения соответствия системы качества производства регламентирующим требованиям.

Обязательная сертификация необходима для программных продуктов и их производства, выполняющих особо ответственные функции, в которых недостаточное качество, ошибки или отказы могут нанести большой ущерб или опасны для жизни и здоровья людей. Этот ущерб может определяться степенью безопасности применения комплексов программ в авиации, для управления в космосе, в атомной энергетике, в военных системах; или большими экономическими потерями в результате низкого качества функционирования ПС в системах государственного управления, в финансовых, банковских,

транспортных системах. В подобных системах обязательная сертификация программных продуктов способствует значительному снижению риска заказчика и повышению безопасности функционирования программного продукта у потребителя до необходимого уровня. В этих случаях разработчики и поставщики программного продукта обязаны подвергать свои изделия сертификации на соответствие требованиям качества и безопасности для получения разрешения компетентных органов на их реальную эксплуатацию и применение по прямому назначению.

Добровольная сертификация применяется с целью повышения конкурентоспособности продукции, расширения сферы ее использования и получения дополнительных экономических преимуществ. Экономическими целями сертификации могут быть большие тиражи изделий при производстве, большая длительность жизненного цикла с множеством версий, снижение налогов за высокое качество, увеличение прибыли разработчиков и поставщиков программного продукта, сокращение рекламаций пользователей. Результаты сертификации должны оправдывать затраты на ее проведение вследствие получения пользователями продукции более высокого и гарантированного качества при некотором повышении ее стоимости. Таким сертификационным испытаниям подвергаются компоненты операционных систем и пакеты прикладных программ широкого применения, повышение гарантии качества, которое выгодно как для поставщиков, так и для пользователей программного продукта. В этих случаях разработчики и поставщики добровольно предоставляют программные средства для сертификации с учетом экономических оценок выгодности ее проведения для их продуктов.

Порядок проведения сертификации в России установлен постановлением Госстандарта России от 21 сентября 1994 г. № 15 по отношению к обязательной сертификации (в том числе и импортируемой продукции), но может

применяться и при добровольной сертификации. Для систем сертификации однородной продукции с учетом ее особенностей допускается разработка соответствующего порядка.

. Этап заявки на сертификацию заключается в выборе заявителем органа по сертификации, способного провести оценку соответствия интересующего его объекта. Это определяется областью аккредитации органа по сертификации.

. Этап оценки соответствия имеет особенности в зависимости от объекта сертификации. Применительно к продукции он состоит из отбора и идентификации образцов изделий и их испытаний. Образцы должны быть такими же, как и продукция, поставляемая потребителю. Образцы выбираются случайным образом по установленным правилам из готовой продукции.

Этап оценки системы качества на предприятии начинается с подготовки в органе по сертификации. При подготовке к проверке и оценке системы качества выполняют следующие работы:

- составляют программу проверки;
- распределяют обязанности между членами комиссии в соответствии с программой проверки;
- подготавливают рабочие документы;
- согласуют программы проверки с проверяемой организацией.

Составление акта, где указываются результаты проверки, выводы и рекомендации комиссии, проводят по окончании работ по оценке соответствия.

На этом этап практической оценки соответствия при сертификации систем качества заканчивается.

. Этап анализа практической оценки соответствия объекта сертификации установленным требованиям заключается в рассмотрении результатов испытаний, экзамена или проверки системы качества в органе по сертификации.

При сертификации систем качества анализ результатов оценки соответствия проводится на основании акта о проверке. Выводы по акту

сводятся к одному из трех вариантов:

) система полностью соответствует заявленному стандарту;

) система в целом соответствует стандарту, но обнаружены отдельные малозначительные несоответствия по элементам системы качества;

) система содержит значительные несоответствия.

. Решение по сертификации сопровождается выдачей сертификата соответствия заявителю или отказом в нем. При положительных результатах испытаний (проверок), предусмотренных схемой сертификации, и экспертизы представленных документов орган по сертификации оформляет сертификат соответствия, регистрирует его и выдает лицензию на право применения знака соответствия. Этим знаком маркируется продукция или документация на услуги, прошедшая сертификацию..

. Инспекционный контроль за сертифицированным объектом проводится органом, выдавшим сертификат, если это предусмотрено схемой сертификации. Он проводится в течение всего срока действия сертификата, обычно один раз в год в форме периодических проверок. В комиссии органа по сертификации при инспекционном контроле могут участвовать специалисты территориальных органов Ростех-регулирования, представители обществ потребителей и других заинтересованных организаций. Внеплановые проверки осуществляются при наличии информации о претензиях к качеству продукции и услуг, а также при существенных изменениях в конструкции сертифицированного изделия, технологии оказания услуг или организационной структуре предприятия, влияющих на процессы системы качества.

1.11 Подготовка к эксплуатации

Все работы данного этапа уже проводятся на территории Заказчика и включают в себя установку и настройку всех компонентов системы в

информационной среде Заказчика, проведение предварительного тестирования, разработку пользовательской документации, обучение пользователей, загрузку исходных данных, проведение испытаний системы в соответствии с программой и методикой испытаний и прочие подготовительные работы.

К моменту окончания всех подготовительных работ должен быть разработан и утвержден регламент эксплуатации системы. Регламент, в частности, должен определять пользователей и их роли в системе, в соответствии с их должностными обязанностями.

Сопровождение (поддержка) программного обеспечения - процесс улучшения, оптимизации и устранения дефектов программного обеспечения после передачи в эксплуатацию. Сопровождение программного обеспечения - это одна из фаз жизненного цикла программного обеспечения, следующая за фазой передачи программного обеспечения в эксплуатацию. В ходе сопровождения в программу вносятся изменения, с тем, чтобы исправить обнаруженные в процессе использования дефекты и недоработки, а также для добавления новой функциональности, с целью повысить удобство использования и применимость программного обеспечения.

2. ПРАКТИЧЕСКАЯ ЧАСТЬ

2.1 Техническое задание

2.1.1 Основание для разработки

Основанием для разработки является задание на производственную практику ПМ.03. Участие в интеграции программных модулей от 30 января 2017 года.

Наименование работы: Автоматизированная информационная система “Учет работы ОЗНА”.

2.1.2 Назначение разработки

Автоматизированная информационная система “Учёт работы «АК ОЗНА»” предназначена для сбора сведений о выполненной работе за определённый промежуток времени и ведения статистики. Пользователями программы выступают бухгалтера компании, отдел учёта, отдел приема и оформления заявок. Выполнение заявок осуществляется на основании договоров о сотрудничестве, в которых оговариваются условия и стоимость работ. Менеджер ведёт учёт полученных заявок, где указывается номер по порядку, срок выполнения, наименование заявки или поломки, фамилия заказчика.

2.1.3 Требования к программе

2.1.3.1 Требования к функциональным характеристикам

Автоматизированная информационная система “Учет работы ОЗНА” должна обеспечивать выполнение функций:

- ввод, хранение, поиск и обработку информации по производству;
- ввод, хранение, поиск и обработку информации по заказам на

предприятии;

- ввод, хранение, поиск и обработку информации по работникам, должностям на предприятии;

2.1.3.2 Требования к надежности

Разрабатываемое программное обеспечение должно иметь:

- возможность самовосстановления после сбоев (отключения электропитания, сбои в операционной системе и т.д.);
- ограничение несанкционированного доступа к данным;
- возможность резервного копирования информационной базы.

Предусмотреть контроль вводимой информации и блокировку некорректных действий пользователя при работе с системой.

2.1.3.3 Требования к составу и параметрам технических средств

Системные требования для работы программного продукта должны быть следующими:

- тактовая частота процессора 1.2 ГГц;
- объём оперативной памяти 64 Мб;
- объём свободного дискового пространства 50 Мб;
- разрешение монитора 1024 x 768;
- наличие устройства чтения компакт-дисков.

2.1.3.4 Требования к информационной и программной совместимости

Программа должна работать в операционных системах Windows XP и выше. Все формируемые отчёты должны иметь возможность экспортирования в редактор электронных таблиц MS Office Access 2003 и выше.

.1.3.5 Требования к транспортированию и хранению

Программа поставляется на лазерном носителе информации. Программная документация поставляется в электронном и печатном виде.

.1.3.6 Специальные требования

Программное обеспечения должно иметь дружественный интерфейс,

рассчитанный на пользователя средний квалификации (с точки зрения компьютерной грамотности).

Ввиду объемности проекта задачи предполагается решать поэтапно. При этом модули программного обеспечения ПО, сделанные в разное время, должны быть совместимы друг с другом, поэтому документация на принятое эксплуатационное ПО должна содержать полную информацию, необходимую для работы с ним программистов. Язык программирования определяется выбором исполнителя, при этом должен обеспечивать возможность поддержки программного обеспечения с пакетом MS Office 2003 и выше.

2.1.3.7 Требования к программной документации

В ходе разработки программы должны быть подготовлены: код программы, описание программы, руководство пользователя, технико-экономическое обоснование.

Экономический эффект от внедрения автоматизированной информационной системы “Учет работы ОЗНА” ожидается за счёт сокращения времени на выполняемые менеджерами операции, исключения ошибок при формировании отчётов, увеличения времени на анализ хозяйственной деятельности и т.д.

2.2 Описание программы

2.2.1 Общие сведения о программе

Программа “Учет работы ОЗНА” предназначена для хранения, изменения, добавления информации об изделиях на предприятии.

Исходными данными является информация о заказчиках, заявках, о специалистах ОЗНА с которым пользователь будет осуществлять работу.

Ввод и редактирование данных в программе происходит путем перехода на нужную таблицу после выбора таблицы в меню с названием “Таблицы”.

Просмотр данных соответствующих запросов происходит путем выбора нужного запроса в меню с названием “Запросы”.

2.2.2 Функциональное назначение

Функциональным назначением программы “Учет работы ОЗНА” является:

- добавление данных;
- редактирование данных;
- удаление данных;
- вывод данных;
- сохранение данных;
- переход на следующую/предыдущую строку;
- переход в начало/конец списка;
- просмотр различных запросов.

.2.3 Описание логической структуры

Работы программы начинается с запуска “Учет работы ОЗНА.exe”. Осуществляется переход на первую форм.

На первой форме расположены:

- список - таблица;
- выбор таблиц и выбор запросов - меню;
- Выход - кнопка;
- “Добавить”, “Редактировать”, “Удалить”, “В начало”, “В конец”, “Вверх”, “Вниз” - кнопки

Макет первой формы представлен на рисунке 5.

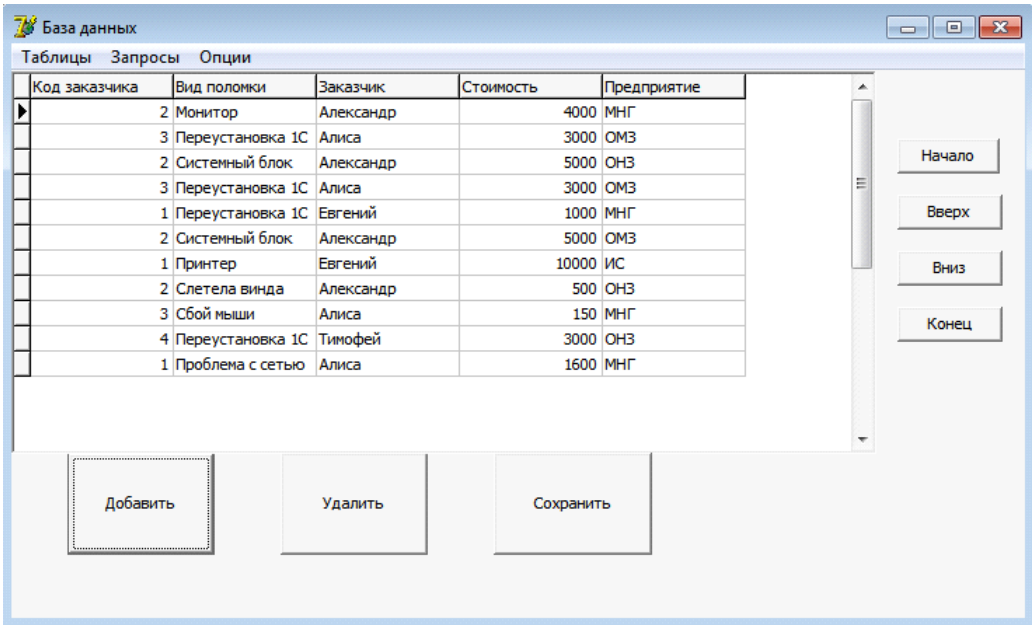

Рисунок 5 Макет первой формы

При нажатии в меню на кнопку “Выход” происходит полное закрытие программы. Для этого используется процедура:

TForm1.N10Click(Sender: TObject);.Close;;

Для открытия одной из таблиц необходимо выбрать в меню с названием “Таблицы”, после этого в списке отобразится выбранная таблица с подстроенным размером колонок. Для этого используется процедура:

TForm1.N2Click(Sender: TObject);:=0;.SQL.Clear;.SQL.Add('SELECT * FROM Заявки');.Active:=True;.Open;i := 0 to DBGrid1.Columns.Count - 1DBGrid1.Columns.Items[i].Width := 101;;TForm1.N3Click(Sender: TObject);:=1;.SQL.Clear;.SQL.Add('SELECT * FROM Заказчики);.Active:=True;.Open;i := 0 to DBGrid1.Columns.Count - 1DBGrid1.Columns.Items[i].Width := 87;;TForm1.N7Click(Sender: TObject);:=2;.SQL.Clear;.SQL.Add('SELECT * FROM [Исполнение заявок]');.Active:=True;.Open;i := 0 to DBGrid1.Columns.Count - 1DBGrid1.Columns.Items[i].Width := 114;;TForm1.N4Click(Sender:

TObject);:=3;.SQL.Clear;.SQL.Add('SELECT * FROM [Специалисты ОЗНА]');.Active:=True;.Open;i := 0 to DBGrid1.Columns.Count - 1DBGrid1.Columns.Items[i].Width := 114;;TForm1.N5Click(Sender: TObject);:=4;.SQL.Clear;.SQL.Add('SELECT * FROM [Список предприятий]');.Active:=True;.Open;i := 0 to DBGrid1.Columns.Count - 1DBGrid1.Columns.Items[i].Width := 146;;

Для выбора одного из заданных запросов необходимо выбрать один из запросов в меню “Запросы”, после этого в списке отобразится выбранный запрос. Для этого используется процедура:

procedure TForm1.N12Click(Sender: TObject);:=5;.SQL.Clear;.SQL.Add('SELECT * FROM Заказ');.Active:=True;.Open;i := 0 to DBGrid1.Columns.Count - 1DBGrid1.Columns.Items[i].Width := 100;;TForm1.N13Click(Sender: TObject);:=6;.SQL.Clear;.SQL.Add('SELECT * FROM Предприятия');.Active:=True;.Open;i := 0 to DBGrid1.Columns.Count - 1DBGrid1.Columns.Items[i].Width := 100;;

На третьей форме расположены:

- Добавить - кнопка;
- выпадающий список для ввода информации.

Макет третьей формы представлен на рисунке 6.

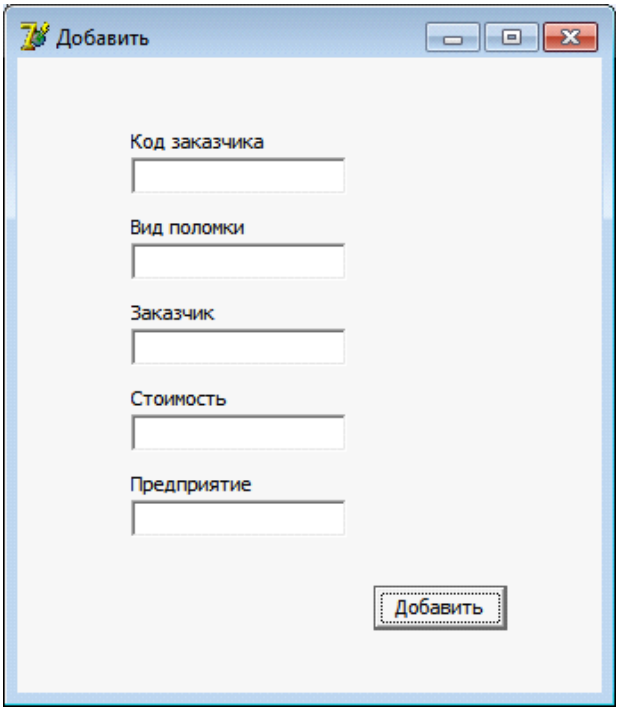

Рисунок 6 Макет третьей формы

При активации формы в выпадающий список добавляются все места работы/должности исходя из того, какой запрос был выбран, для выбора одного из них и вывода сотрудников производства по данным критериям. Для этого используется процедура:

TForm3.FormActivate(Sender: TObject);x,i:integer;:=Form1.ADOQuery1.FieldCount;i:=0 to x-1 do(Components[i]).Caption:=Form1.ADOQuery1.Fields[i].FieldName; (Components[i]).Visible:=true;(Components[i+6]).Visible:=true;;

end;

При нажатии на кнопку “Добавить” третья форма закроется, после этого на первой форме в списке отобразится информация по выбранным критериям. Для этого используется процедура:

procedure TForm3.Button1Click(Sender: TObject);

var i:integer;

begin.ADOQuery1.insert;ed of

:.ADOQuery1.FieldByName(Код заказчика).AsString:=Edit1.Text;.ADOQuery1.FieldByName('Вид поломки).AsString:=Edit2.Text;.ADOQuery1.FieldByName('Заказчик').AsString:=Edit3.Text;.ADOQuery1.FieldByName('Стоимость').AsString:=Edit4.Text;.ADOQuery1.FieldByName('Предприятие').AsString:=Edit5.Text;;

:.ADOQuery1.FieldByName('Код заказчика').AsString:=Edit1.Text;.ADOQuery1.FieldByName('Имя заказчика').AsString:=Edit2.Text;.ADOQuery1.FieldByName('Фамилия заказчика').AsString:=Edit3.Text;.ADOQuery1.FieldByName('Должность').AsString:=Edit4.Text;.ADOQuery1.FieldByName('Отдел').AsString:=Edit5.Text;.ADOQuery1.FieldByName('Номер телефона').AsString:=Edit5.Text;;

:.ADOQuery1.FieldByName('Код специалиста').AsString:=Edit1.Text;.ADOQuery1.FieldByName('Имя специалиста').AsString:=Edit2.Text;.ADOQuery1.FieldByName('Наименование заявки').AsString:=Edit3.Text;.ADOQuery1.FieldByName('Стоимость').AsString:=Edit4.Text;;

:.ADOQuery1.FieldByName('Код специалиста').AsString:=Edit1.Text;.ADOQuery1.FieldByName('Имя специалиста').AsString:=Edit2.Text;.ADOQuery1.FieldByName('Фамилия специалиста').AsString:=Edit3.Text;.ADOQuery1.FieldByName('Номер телефона').AsString:=Edit4.Text;;

:.ADOQuery1.FieldByName('Наименование предприятия').AsString:=Edit1.Text;.ADOQuery1.FieldByName('Адрес').AsString:=Edit2.Text;;;;.Close;.Clear; edit2.Clear; edit3.Clear;.Clear; edit5.Clear; edit6.Clear;;

При закрытии формы происходит очистка выпадающего списка. Для этого используется процедура:

TForm3.FormClose(Sender: TObject; var Action: TCloseAction);x,i:integer;:=Form1.ADOQuery1.FieldCount;i:=0 to x-1 do(Components[i]).Visible:=false;(Components[i+6]).Visible:=false;;;

2.3 Руководство оператора

2.3.1 Назначение программы

Программа “Учет работы ОЗНА” предназначена для ведения учёта производства, закупок изделий на предприятии. Программа выполняет следующее:

- позволяет вносить изменение информации о заказах, заявках, закупке изделий: добавлять, удалять, редактировать;

- выполнять поиск сотрудникам по их должности, либо месту их работы;
- защищает от несанкционированного входа в программу путём ввода пароля при запуске программы.

.3.2 Условия выполнения программы

Для правильной работы программы необходимы следующие аппаратные и программные средства:

- Компьютер с установленной операционной системой MS Windows XP и выше;
- тактовая частота процессора 1.2 ГГц;
- объём оперативной памяти 64 Мб;
- объём свободного дискового пространства 50 Мб;
- разрешение монитора 800 x 600 или лучше;
- клавиатура и мышь;
- наличие устройства чтения компакт-дисков.

2.3.3 Выполнение программы

Сначала нужно установить программу с помощью инсталляционного файла “Setup.exe” и следовать инструкциям, отображаемым в процессе установки.

Запуск программы “ Учет работы ОЗНА” осуществляется при помощи соответствующего ярлыка на рабочем столе.

После запуска программы будет произведен вход в программу и открытие главного окна для работы с данными. Изображение главного окна представлено на рисунке 8.

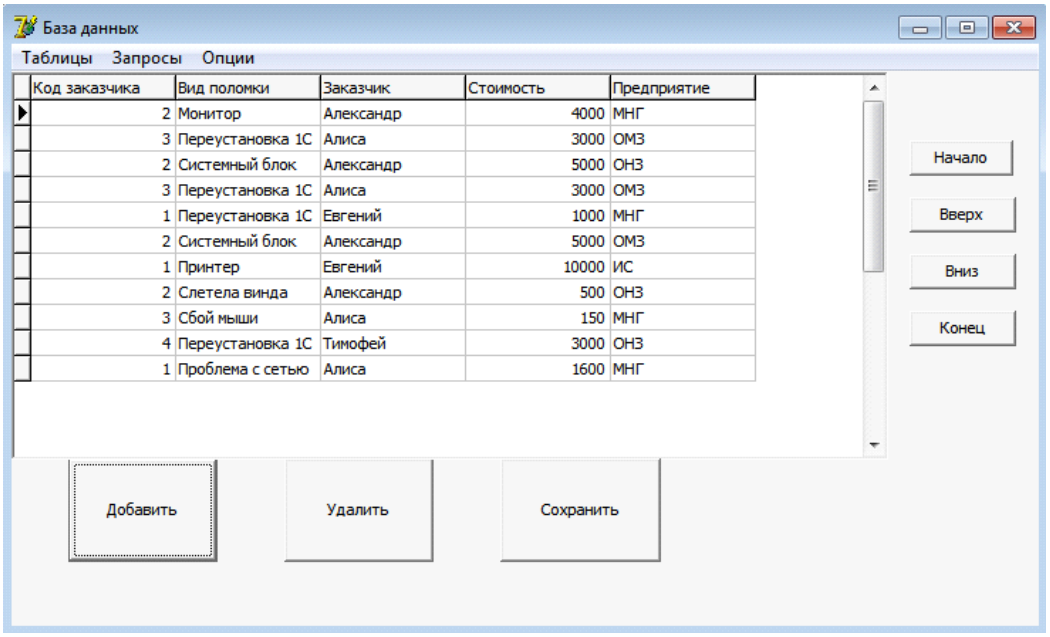

Рисунок 8 Главное окно программы

Для выхода из программы необходимо нажать в меню на кнопку “Выход”.

Для вывода данных необходимо в меню выбрать запрос, либо таблицу с соответствующими названиями “Таблицы”, ”Запросы”. Изображение представлено в рисунке 9.

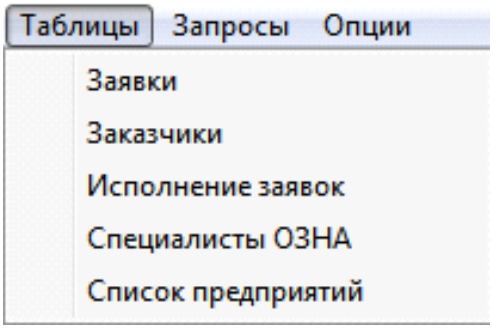

Рисунок 9 Выбор таблицы/запроса

Для добавления, удаления данных, перехода по записям, обновление данных используются кнопки по таблице. Изображение кнопок представлено на рисунке 10.

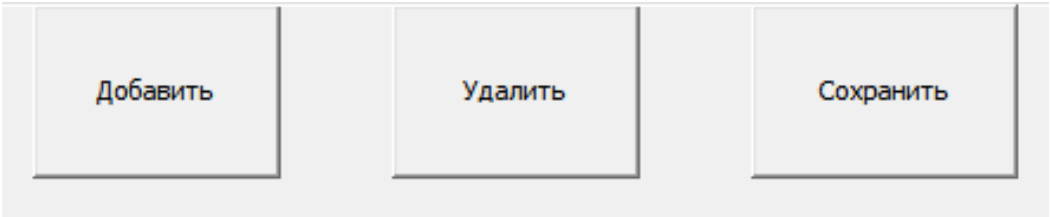

Рисунок 10 Навигатор по таблице

При нажатии на запрос “Добавить” откроется дополнительное окно с выпадающим списком для выбора критерия для поиска. После выбора необходимо нажать кнопку “Добавить”. Изображение окна поиска представлено на рисунке 11.

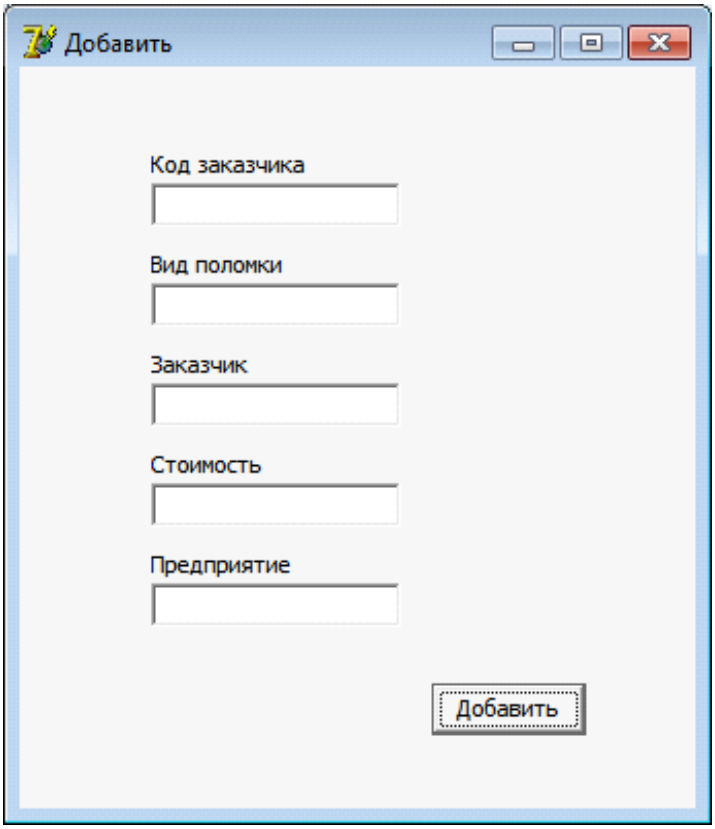

Рисунок 11 Окно поиска

.3.4 Сообщения оператору

Нет сообщений пользователю.

2.4 Сертификация

.4.1 Подготовка перечня документации для прохождения сертификации

Сертификация не предусмотрена, так как разработка программы велась в сотрудничестве с заказчиком и поэтому перечень документации не был необходим.

2.4.2 Проверка соответствия требованиям

Программа была разработана в соответствии с требованиями к программе, указанными в п.2.1.3 и п.2.1.4 и с требованиями ГОСТ 19.701 - 90 (ИСО 5807 - 85). Код программы был выполнен в соответствии со стандартом оформления Delphi.

.4.3 Подготовка к сертификационным испытаниям и их проведение

Основной целью сертификации программных продуктов, является контроль и удостоверение качества продукции, гарантирование её высоких потребительских свойств. Задача состоит в повышении эффективности затрат в сфере создания и применения ответственного, конечного программного продукта, а также улучшение объективности оценок его функций, характеристик и конкурентоспособности. Формальной целью сертификации является подготовка и принятие решения о целесообразности выдачи сертификата соответствия с учетом следующих факторов:

- полноты, точности и достоверности исходного технического задания и спецификаций требований, представленной в документации на программный продукт;
- достоверности и точности измерения и обобщения результатов сертификационных испытаний, получения адекватных показателей качества конечных продуктов и соответствия требованиям заказчика;
- методологии и качества интерпретации данных об объекте испытаний с учетом достоверности оценок, квалификации и объективности испытателей, заказчиков и пользователей.

В процессе испытаний все функции программы работают и удовлетворяют требованиям заказчика.

Тестирование программы прошло успешно. В ходе тестов было проверено:

- работоспособность на операционных системах Windows XP и Windows 8;
- возможность добавления, удаления, редактирования информации в базе данных;
- достоверность выполнения запросов с подсчётом и поиском.

2.4.4 Приемка и эксплуатация программного обеспечения

Приемка программного обеспечения предусматривает оценку результатов квалификационного тестирования программного обеспечения и системы и документирование результатов оценки, которые проводятся заказчиком с помощью разработчика. Разработчик выполняет окончательную передачу программного обеспечения заказчику в соответствии с договором, обеспечивая при этом необходимое обучение и поддержку.

Процесс эксплуатации охватывает действия и задачи оператора - организации, эксплуатирующей систему, и включает действия:

Управление конфигурацией позволяет организовать, систематически учитывать и контролировать внесение изменений в программное обеспечение на всех стадиях его жизненного цикла. Общие принципы и рекомендации по управлению конфигурацией программного обеспечения отражены в проекте стандарта ISO/IEC 12207-2: 1995 “Information Technology - Software Life Cycle Processes, Part 2, Configuration Management for Software”.

Процесс обеспечения качества обеспечивает соответствующие гарантии того, что программное обеспечение и процессы его жизненного цикла соответствуют заданным требованиям и утвержденным планам. Под качеством программного обеспечения понимается совокупность свойств, которые характеризуют способность программного обеспечения удовлетворять заданным требованиям.

.4.5 Разработка пользовательской документации

Пользовательская документация представляет собой руководство пользователя, которое описывает каждую функцию программы, а также шаги, которые нужно выполнять для использования этой функции. Также пользовательская документация должна предоставлять инструкции о том, что делать в случае возникновения проблем. Очень важно, чтобы документация не вводила в заблуждение и была актуальной. Руководство должно иметь чёткую структуру. Существует три подхода к организации пользовательской документации:

- вводное руководство, наиболее полезное для новых пользователей, последовательно проводит по ряду шагов, служащих для выполнения каких-либо типичных задач;
- тематический подход, при котором каждая глава руководства посвящена какой-то отдельной теме, больше подходит для совершенствующихся пользователей;
- подход, в котором команды или задачи организованы в виде алфавитного справочника - часто это хорошо воспринимается продвинутыми пользователями, хорошо знающими, что они ищут.

Жалобы пользователей обычно относятся к тому, что документация охватывает только один из этих подходов, и поэтому хорошо подходит лишь для одного класса пользователей.

Во многих случаях разработчики программного продукта ограничивают набор пользовательской документации справочной информацией о командах или пунктах меню и прочее.

Документация пользователя была составлена в соответствии с ГОСТ 19.101-77 Разработка программных документов.

2.4.6 Определение состава документации

Руководство по использованию программы “Учет работы ОЗНА” содержит следующие разделы:

- пункт “Назначение программы” содержит информацию о программе, назначениях, целях;
- пункт “Условия выполнения программы” содержит информацию об аппаратных и программных средствах необходимых для правильной работы программы;
- пункт “Выполнение программы” и пункт “Сообщения оператору” содержат подробную инструкцию, как правильно работать с программой.

.4.7 Подготовка руководства пользователя

Руководство пользователя выполнено в соответствии с п.2.1.3 и п.2.1.4, и ГОСТ 19.101-77. Документ содержит всю необходимую информацию, по которой пользователь сможет без особых усилий разобраться в возможностях программы, а также решать возникшие проблемы во время использования.

ЗАКЛЮЧЕНИЕ

Существует много различных причин перевода информации на компьютерную основу. Сейчас стоимость хранения информации в файлах ЭВМ дешевле, чем на бумаге. ЭВМ позволяет хранить, структурировать и извлекать информацию оптимальным для пользователя способом. Использование клиент-сервисных технологий позволяет сберечь значительные средства. А главное и время для получения необходимой информации, а также упрощает доступ и ведение, поскольку они основываются на комплексной обработке данных и централизация их хранения.

При выполнении отчёта по производственной была разработана программа “Учет работы ОЗНА” с использованием средств быстрой разработки приложения Delphi 7, а также документации к приложению в соответствии с требованиями ГОСТ и ЕСПД. Данная программа содержит сведения о закупке и производства изделий, информации об изделиях, сотрудниках на предприятии и может реализовать следующие функции:

- добавление данных;
- редактирование данных;
- удаление данных;
- вывод данных;
- сохранение данных;
- переход на следующую/предыдущую строку;
- переход в начало/конец списка;
- просмотр различных запросов.

В результате прохождения производственной (по профилю специальности) практики в рамках профессионального модуля был приобретен практический опыт работы:

- с проектной и технической документацией на уровне взаимодействия

компонент программного обеспечения;

- выполнения интеграции модулей в программную среду;
- выполнения отладки программного продукта с использованием специализированных программных средств;
- разработки текстовых наборов и текстовых сценариев;
- проведения инспектирования компонент программного продукта на предмет соответствия стандартам кодирования.

СПИСОК ЛИТЕРАТУРЫ

1. Нил Дж. Рубенкинг. Язык программирования Delphi для «чайников». Введение в Borland Delphi = Delphi for Dummies. - М.: Диалектика, 2013.
2. Хавьер Пашеку. Программирование в Borland Delphi для профессионалов = Delphi for.NET Developer’s Guide. - М.: Вильямс, 2012.
3. Осипов Д. Базы данных и Delphi. Теория и практика. - СПб.: БХВ-Петербург, 2014. - С. 752.
. А. Н. Вальвачев, К. А. Сурков, Д. А. Сурков, Ю. М. Четырько. Программирование на языке Delphi. Учебное пособие. - 2015.
. Когаловский М. Р. Энциклопедия технологий баз данных. - М.: Финансы и статистика, 2012. - 800 с.
. Кузнецов С. Д. Основы баз данных. М.: Интернет-университет информационных технологий; БИНОМ. Лаборатория знаний, 2016. - 484 с.
. Дейт К. Дж. Введение в системы баз данных = Introduction to Database Systems. - 8-е изд. - М.: Вильямс, 2015. - 1328 с.
. Аллен Тейлор. SQL для чайников, 8-е издание = SQL For Dummies, 8th edition. - М.: «Диалектика», 2014. - 416 с.
. Крис Фиайли. SQL: Руководство по изучению языка. - М.: Peachpit Press, 2013. - 456 с.
10. Джеймс Р. Грофф, Пол Н. Вайнберг, Эндрю Дж. Оппель. SQL: полное руководство, 3-е издание = SQL: The Complete Reference, Third Edition. - М.: «Вильямс», 2014. - 960 с.
11. Культин Н.Б. Основы программирования в Embarcadero Delphi, 2015.
. Рубанцев В. Delphi в примерах, играх и программах, 2012. -418 с.
. Культин Н. Основы программирования в Delphi XE. - СПБ.: БХВ-Петербург, 2013. - 416 с.
. Вирт Н. Алгоритмы + структуры данных = программы. - М.: Мир, 2015. -

С. 406.

. Вирт Н., Йенсен К. Паскаль. Руководство для пользователя и описание языка, 2012. - С. 151.

ПРИЛОЖЕНИЕ А

код программы

Первая форма:

```
unit Unit1;
interface
uses
Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
Dialogs, DB, ADODB, Menus, Grids, DBGrids, StdCtrls;
type
TForm1 = class(TForm)
MainMenu1: TMainMenu;: TButton;: TButton;: TDBGrid;: TDataSource;: TADOQuery;: TMenuItem;: TMenuItem;: TMenuItem;: TMenuItem;: TMenuItem;: TMenuItem;: TMenuItem;: TADOConnection;: TMenuItem;: TMenuItem;: TButton;: TButton;: TButton;: TButton;: TButton;: TMenuItem;: TMenuItem;: TMenuItem;N2Click(Sender: TObject);FormCreate(Sender: TObject);N3Click(Sender: TObject);N4Click(Sender: TObject);N5Click(Sender: TObject);Button2Click(Sender: TObject);Button1Click(Sender: TObject);N9Click(Sender: TObject);N7Click(Sender: TObject);N10Click(Sender: TObject);N8Click(Sender: TObject);Button3Click(Sender: TObject);Button4Click(Sender: TObject);Button5Click(Sender: TObject);Button6Click(Sender: TObject);Button7Click(Sender: TObject);N12Click(Sender: TObject);N13Click(Sender: TObject);
{ Private declarations }
{ Public declarations };: TForm1; i,ed:integer;Unit3, Unit2;
{$R *.dfm}TForm1.FormCreate(Sender: TObject);:=0;.SQL.Clear;.SQL.Add('SELECT * FROM
```

Заявки');.Active:=True;.Open;i := 0 to DBGrid1.Columns.Count - 1DBGrid1.Columns.Items[i].Width := 101;;TForm1.N2Click(Sender: TObject);:=0;.SQL.Clear;.SQL.Add('SELECT * FROM Заявки');.Active:=True;.Open;i := 0 to DBGrid1.Columns.Count - 1DBGrid1.Columns.Items[i].Width := 101;;TForm1.N3Click(Sender: TObject);:=1;.SQL.Clear;.SQL.Add('SELECT * FROM Заказчики');.Active:=True;.Open;i := 0 to DBGrid1.Columns.Count - 1DBGrid1.Columns.Items[i].Width := 87;;TForm1.N7Click(Sender: TObject);:=2;.SQL.Clear;.SQL.Add('SELECT * FROM [Исполнение заявок]');.Active:=True;.Open;i := 0 to DBGrid1.Columns.Count - 1DBGrid1.Columns.Items[i].Width := 114;;TForm1.N4Click(Sender: TObject);:=3;.SQL.Clear;.SQL.Add('SELECT * FROM [Специалисты ОЗНА]');.Active:=True;.Open;i := 0 to DBGrid1.Columns.Count - 1DBGrid1.Columns.Items[i].Width := 114;;TForm1.N5Click(Sender: TObject);:=4;.SQL.Clear;.SQL.Add('SELECT * FROM [Список предприятий]');.Active:=True;.Open;i := 0 to DBGrid1.Columns.Count - 1DBGrid1.Columns.Items[i].Width := 146;;TForm1.Button2Click(Sender: TObject);Application.MessageBox('Удалить?','Удалить',MB_YESNO)=IDYES then.Delete;;e:Exception do;;TForm1.Button1Click(Sender: TObject);.showmodal;;TForm1.N9Click(Sender: TObject);.show;;TForm1.N10Click(Sender: TObject);.Close;;TForm1.N8Click(Sender: TObject);:= (sender as TMenuItem).MenuIndex;id of

: ADOQuery1.SQL.Add('SELECT * FROM Заявки');

: ADOQuery1.SQL.Add('SELECT * FROM Заказчики');

: ADOQuery1.SQL.Add('SELECT * FROM [Исполнение заявок]');

: ADOQuery1.SQL.Add('SELECT * FROM [Специалисты ОЗНА]');

: ADOQuery1.SQL.Add('SELECT * FROM [Список предприятий]');

: ADOQuery1.SQL.Add('SELECT * FROM Заказ');

: ADOQuery1.SQL.Add('SELECT * FROM Предприятия');;.Active:=True;;TForm1.Button3Click(Sender: TObject);.First;;TForm1.Button4Click(Sender: TObject);.Last;;TForm1.Button5Click(Sender: TObject);.Next;;TForm1.Button6Click(Sender: TObject);.Prior;;TForm1.Button7Click(Sender: TObject);Application.MessageBox('Сохранить?','Сохранение',MB_YESNO)=IDYES then.Post;;e:Exception do; end;TForm1.N12Click(Sender: TObject);:=5;.SQL.Clear;.SQL.Add('SELECT * FROM Заказ');.Active:=True;.Open;i := 0 to DBGrid1.Columns.Count - 1DBGrid1.Columns.Items[i].Width := 100;;TForm1.N13Click(Sender: TObject);:=6;.SQL.Clear;.SQL.Add('SELECT * FROM Предприятия');.Active:=True;.Open;i := 0 to DBGrid1.Columns.Count - 1DBGrid1.Columns.Items[i].Width := 100;;.

Третья форма:Unit3;, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,, StdCtrls, DB, ADODB;= class(TForm): TEdit;: TEdit;: TEdit;: TEdit;: TEdit;: TLabel;: TLabel;: TLabel;: TLabel;: TLabel;: TEdit;: TLabel;: TButton;: TADOQuery;: TDataSource;: TADOConnection;FormActivate(Sender: TObject);FormClose(Sender: TObject; var Action: TCloseAction) ;Button1Click(Sender: TObject);

{ Private declarations }

{ Public declarations };: TForm3; x,id:integer;Unit1, Unit2;

{$R *.dfm}TForm3.FormActivate(Sender: TObject);x,i:integer;:=Form1.ADOQuery1.FieldCount;i:=0 to x-1 do(Components[i]).Caption:=Form1.ADOQuery1.Fields[i].FieldName; (Components[i]).Visible:=true; (Components[i+6]).Visible:=true;;;TForm3.FormClose(Sender: TObject; var Action:

TCloseAction);x,i:integer;:=Form1.ADOQuery1.FieldCount;i:=0 to x-1 do(Components[i]).Visible:=false;
(Components[i+6]).Visible:=false;;;TForm3.Button1Click(Sender: TObject);i:integer;.ADOQuery1.insert;ed of

:.ADOQuery1.FieldByName('Код заказчика').AsString:=Edit1.Text;.ADOQuery1.FieldByName('Вид поломки').AsString:=Edit2.Text;.ADOQuery1.FieldByName('Заказчик').AsString:=Edit3.Text;.ADOQuery1.FieldByName('Стоимость').AsString:=Edit4.Text;.ADOQuery1.FieldByName('Предприятие').AsString:=Edit5.Text;;

:.ADOQuery1.FieldByName('Код заказчика').AsString:=Edit1.Text;.ADOQuery1.FieldByName('Имя заказчика').AsString:=Edit2.Text;.ADOQuery1.FieldByName('Фамилия заказчика').AsString:=Edit3.Text;.ADOQuery1.FieldByName('Должность').AsString:=Edit4.Text;.ADOQuery1.FieldByName('Отдел').AsString:=Edit5.Text;.ADOQuery1.FieldByName('Номер телефона').AsString:=Edit5.Text;;

:.ADOQuery1.FieldByName('Код специалиста').AsString:=Edit1.Text;.ADOQuery1.FieldByName('Имя специалиста').AsString:=Edit2.Text;.ADOQuery1.FieldByName('Наименование заявки').AsString:=Edit3.Text;.ADOQuery1.FieldByName('Стоимость').AsString:=Edit4.Text;;

:.ADOQuery1.FieldByName('Код специалиста').AsString:=Edit1.Text;.ADOQuery1.FieldByName('Имя специалиста').AsString:=Edit2.Text;.ADOQuery1.FieldByName('Фамилия специалиста').AsString:=Edit3.Text;.ADOQuery1.FieldByName('Номер телефона').AsString:=Edit4.Text;;

:.ADOQuery1.FieldByName('Наименование предприятия').AsString:=Edit1.Text;.ADOQuery1.FieldByName('Адрес').AsString:=Edit2.Text;;;;.Close;.Clear; edit2.Clear; edit3.Clear;.Clear; edit5.Clear; edit6.Clear;;

end.