МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

ЗВІТ
З ПЕРЕДАТЕСТАЦІЙНОЇ ПРАКТИКИ СПЕЦІАЛІСТІВ
Місце проходження практики «НВП "ХАРТРОН-АРКОС"»
у період з "06" лютого 2017 р. по "01" квітня 2017 р.
Тема індивідуального завдання:

Система для планування та управління бюджетом підприємства

Студ. ІПЗс-16-2 Філіпов О.О.

Керівник практики Проф. Дудар З.В.

Робота захищена з оцінкою
_____
Дата «__»_____2017

Комісія:
Проф. Дудар З.В. _____

Доц. Кузнєцов Ю.О. _____

Ст. викл. Трощило О.С. _____

Харків, 2017

РЕФЕРАТ/ABSTRACT

Пояснювальна записка: 25 стор., 6 рис., 13 дж.

Метою роботи є проектування системи корпораційного планування та управління бюджетом підприємства.

У результаті був проведений аналіз предметної області та спроектована система фінансового планування.

ФІНАНСИ, ФІНАНСОВА СИСТЕМА, БЮДЖЕТУВАННЯ, UML, EF, MVC, C#, КОНТРОЛЕР, СЕРВЕР, REST, MS SQL.

Explanatory note 25, p. 6 Fig. 14 joules.

The aim is to design a system collegiate planning and budget management company.

As a result of an analysis of the subject area and designed system of financial planning.

FINANCE, FINANCIAL SYSTEMS, BUDGETING, UML, EF, MVC, C #, CONTROLLER, SERVER, REST, MS SQL.

ЗМІСТ

ВВЕДЕНИЕ

Планирование бюджета является очень важным фактором в любом предприятии, будь то большая корпорация или маленькая фирма. Естественно в большинстве предприятий в штате сотрудников присутствуют бухгалтеры, занимающиеся бухгалтерским учетом и работающие со сложными системами для его ведения. Но далеко не всегда бухгалтеры принимают решения о распределении бюджета в компании. Часто этим занимается отдельно лицо, этим лицом может быть менеджер или даже владелец компании. Для такого человека нет необходимости разбираться в огромном количестве кредитов и дебетов. Ему было бы гораздо проще видеть всю финансовую картину целиком, возможно, в графическом виде, и манипулировать финансами используя удобный и интуитивно понятный интерфейс небольшого приложения. Также возможны случаи, когда решения о распределении финансов может принимать два или более человек, поэтому большим плюсом была бы возможность управления финансами предприятия одновременно нескольким лицам, синхронизируя свои действия.

Решением данной проблемы является разработка системы, которая позволит управлять финансами, не требуя специальных навыков. Главное в системе – ее понятность и простота использования. Система позволит пользователю добавлять единоразовую и периодическую прибыль, также определить периодические затраты (такие как выплата зарплат, затраты на обслуживание офиса и т.д.). Кроме периодических затрат присутствуют единоразовые затраты (к примеру организация корпоративного торжества и прочие). Все эти данные пользователь определяет самостоятельно. Система позволит просматривать план бюджета в виде различных диаграмм, таблиц и списков, что упрощает понимание картины в целом, а простой интерфейс позволит безо всяких трудностей вносить изменения. Для удобства использования все финансовые изменения можно объединять в группы. Также одной из главных фич является возможность пользователя создавать собственные шаблоны финансовых операций (например разделить указанную сумму среди всех сотрудников в виде премии)[1].

Система будет взаимодействовать с пользователем через веб-сайт, что делает ее кроссплатформенной и предоставляет возможность пользоваться ее с любого устройства через сеть интернет.

Система подлежит к использованию в малых предприятиях, небольших частных фирмах или в филиалах больших предприятий, но также не исключается использование системы для планирования домашнего бюджета.

1 ИНФОРМАЦИЯ О ПРЕДПРИЯТИИ

1.1 Научно-техническая и производственная деятельность нпп хартрон-аркос лтд

НАУЧНО-ПРОИЗВОДСТВЕННОЕ ПРЕДПРИЯТИЕ (НПП) ХАРТРОН-АРКОС ЛТД (ОБЩЕСТВО С ОГРАНИЧЕННОЙ ОТВЕТСТВЕННОСТЬЮ) образовано в 1992 году как дочернее предприятие ПАО «ХАРТРОН». Основной его задачей является создание систем управления для различных объектов ракетно-космической техники.

По тематике разработок НПП ХАРТРОН-АРКОС ЛТД является преемником образованного в апреле 1959 года ОКБ-692 (а/я 67), в последствие КБ Электроприборостроения, НПО «Электроприбор».

НПП ХАРТРОН-АРКОС ЛТД разработаны и внедрены в эксплуатацию системы управления ряда ракет-носителей (РН): «Днепр», «Рокот», «Стрела», «Циклон»; космических аппаратов (КА): «Аркон», «Купон», транспортных модулей орбитального комплекса «Мир», блока «Заря» международной космической станции «Альфа», «Коронас-И», «Коронас-Ф», «EgyptSat-1», МС-2-8 и др.

Одновременно выполнялись работы по созданию автоматизированных систем управления (АСУ) технологическими процессами (ТП) для нужд объектов атомной энергетики и железнодорожного транспорта.

Опыт работы НПП ХАРТРОН-АРКОС ЛТД востребован странами дальнего зарубежья. В частности, выполнены контракты с рядом проектных и исследовательских институтов Италии, Китая, США.

В настоящее время в НПП ХАРТРОН-АРКОС ЛТД разрабатываются системы управления ряда ракет-носителей, а также украинского микроспутника системы «Сич-2», КА «Микросат», ведутся работы по сопровождению и модернизации систем управления эксплуатируемых КА. Создается программно-математическое обеспечение системы контроля ядерной подкретичной установки (ЯПУ), создаваемой по международному проекту в Харьковском физико-техническом институте.

В НПП ХАРТРОН-АРКОС ЛТД трудится более 700 инженерно-технических работников, в числе которых более 20 сотрудников, имеют научные степени, 9 человек удостоены почетных званий Лауреатов премии Правительства Российской Федерации и Государственных премий Украины.

Организационно НПП ХАРТРОН-АРКОС ЛТД находится в подчинении Государственного космического агентства Украины (ГКАУ).

1.2 Структура пао «хартрон» и нпп хартрон-аркос лтд

ПАО «ХАРТРОН» имеет следующую структурно-организационную схему («холдинговую» структуру):

1. Комплекс научно-производственных предприятий:
   - 1.1 «ХАРТРОН-АРКОС»;
   - 1.2 «ХАРТРОН-ЮКОМ»;
   - 1.3 «ХАРТРОН-ИНКОР»;
   - 1.4 «ХАРТРОН-ОРБИТА»;
   - 1.5 «ХАРТРОН-СИГМА»;
   - 1.6 «ХАРТРОН-ЭКСПРЕСС».
2. Завод «ХАРТРОН-ПЛАНТ»:
   - 2.1 комплексы;
     - 2.1.1 монтажно-сборочный;
     - 2.1.2 производственно-механический;
   - 2.2 службы;
     - 2.2.1 главного механика;
     - 2.2.2 главного электрика;
     - 2.2.3 планирования организации труда и заработной платы;
     - 2.2.4 АСУП;
     - 2.2.5 отдел технического контроля;
     - 2.2.6 сбыта;
   - 2.3 спец. конструкторское технологическое бюро.
3. Службы:
   - 3.1 «ХАРТРОН-ЦЕНТР»;
   - 3.2 группа референтов;
   - 3.3 по кадрам;
   - 3.4 режимно-секретный;
   - 3.5 охраны труда и техники безопасности;
   - 3.6 экономики и управления;
   - 3.7 маркетинга;
   - 3.8 коммерческий;
   - 3.9 обеспечения качества;
   - 3.10 по социальным вопросам;
   - 3.11 юридический.

4. Фирмы:

4.1 «ХАРТРОН-ТРАНС»;

4.2 «ХАРТРОН-ЭЛЕКТРОСВЯЗЬ».

5. Представительство заказчика.

На рис. 1 приведена типовая структурно-организационная схема научно-производственного предприятия ПАО «ХАРТРОН» на примере НПП ХАРТРОН-АРКОС ЛТД.

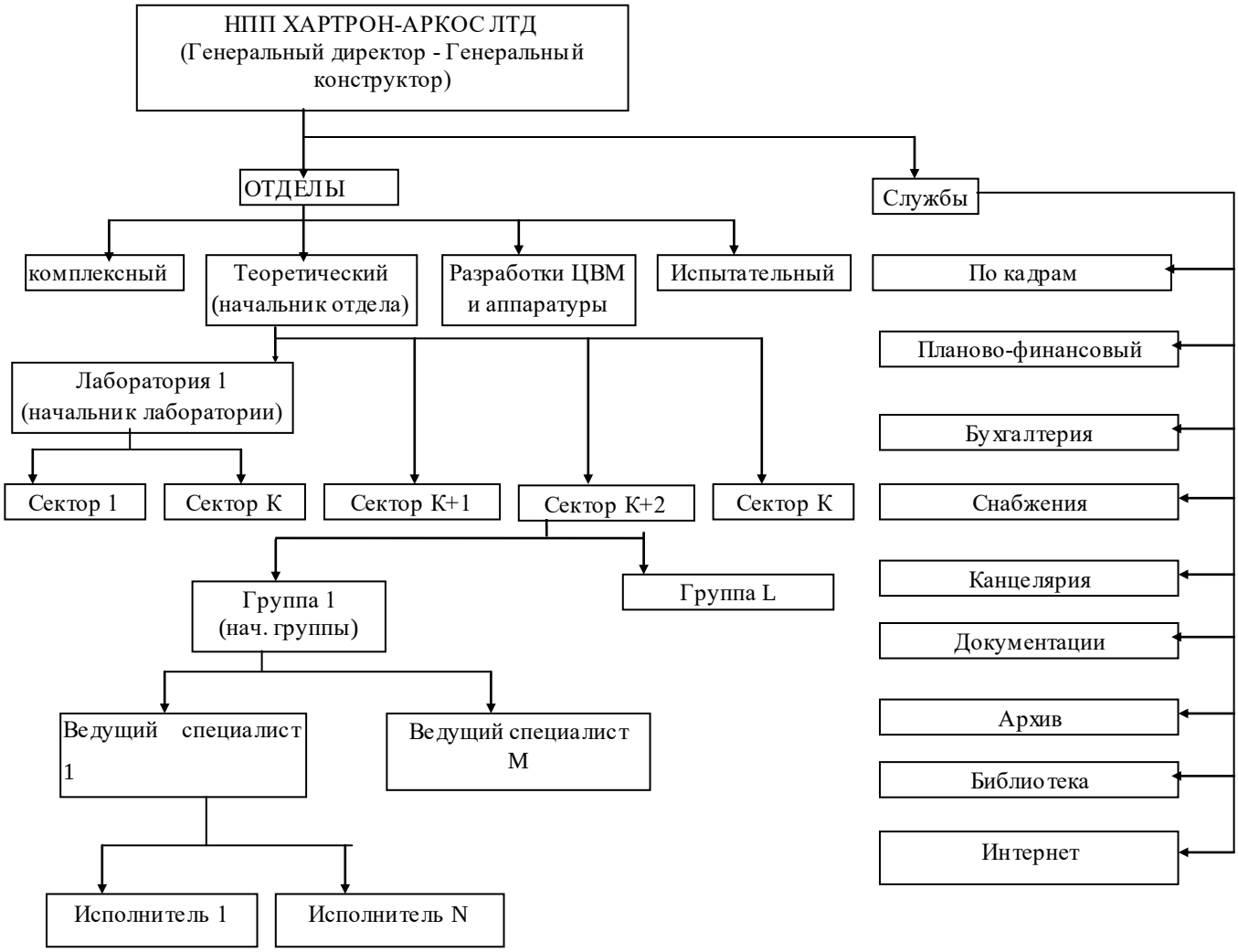

Рисунок 1 - Структура НПП ХАРТРОН-АРКОС ЛТД ПАО «ХАРТРОН»

2. АНАЛИЗ ПРОБЛЕМНОЙ ОБЛАСТИ И ПОСТАНОВКА ЗАДАЧИ

2.1 Теоретические основы

Данная система является частным случаем планировщика бюджета со своими фичами.

Планировщик бюджета – приложение, позволяющее вести учет финансовых средств, и управлять их распределением.

Основным назначением данной системы является управление бюджетом небольших предприятий. Система позволяет избавиться от постоянной путаницы с массой счетов и значительно упрощает расчёт финансов предприятия даже для ничего не смыслящего в финансах человека.

Большие и сложные системы бухгалтерского учета здесь не рассматриваются так как они не являются конкурирующими продуктами. Предприятиям с многочисленными финансовыми оборотами просто не обойтись без систем бухгалтерского учета. Но данная система предназначена не для бухгалтеров, а для управляющих компаниями, для которых в системах бухгалтерского учета будет присутствовать много лишней информации, что затруднит распределение финансов для них. Поэтому данная система может отлично работать в параллельно с программными средствами для бухгалтерского учета.

2.2 Анализ аналогов

В данный момент уже существует масса подобных сервисов и приложений. Среди них очень много систем и приложений, предназначенных для управления домашним бюджетом. Либо наоборот функционал и общая концепция системы сильно приближается к концепции систем бухгалтерского учета. Ниже описаны несколько конкурирующих продуктов:

Money Care - финансовый планировщик для IOS и Andriod от компании StudioMoob. Приложение для дотошных счетоводов и просто больших любителей денег, готовых, как Скрудж Макдак, пересчитывать свои доходы каждую свободную минуту. Если вы не боитесь пожертвовать своим временем и удобствами в угоду точности учёта расходов, то это приложение определённо займёт должное место в вашем телефоне. При всей своей неказистости и внешней недружелюбности, Money Care обладает богатым функционалом, позволяющим учитывать массу разных мелочей: смотреть графики своих затрат, делить расходы на несколько человек и так далее[2].

Рисунок 2.1 – Интерфейс приложения Money Care

К преимуществам данного приложения можно отнести возможность резервного копирования данных на почту, Dropbox или Google Drive а также экспорт данных в формате Exel.

Недостатками являются относительно сложный и неудобный интерфейс к примеру, для добавления поступлений и расходов: вбей название транзакции — подтверди, добавь сумму — подтверди, не забудь присвоить категорию — подтверди, а потом ещё раз подтверди всю операцию. Приложение создано только для двух мобильных платформ. Приложения является условно-бесплатным, бесплатно можно сделать до 90 записей, а для большего количества необходимо приобрести полную версию. Также отсутствует возможность планирования бюджета несколькими пользователями одновременно, с постоянной синхронизацией данных между ними.

Oracle Financial Analyzer OFA является самым многофункциональным инструментом моделирования бюджетов. Эта система позволяет применять формулы любой степени сложности, и проводить многомерный анализ данных в любом разрезе (например, анализ доходности компании за II квартал 2002 г. по каждому продукту с разбивкой по дилерам региона)[3].

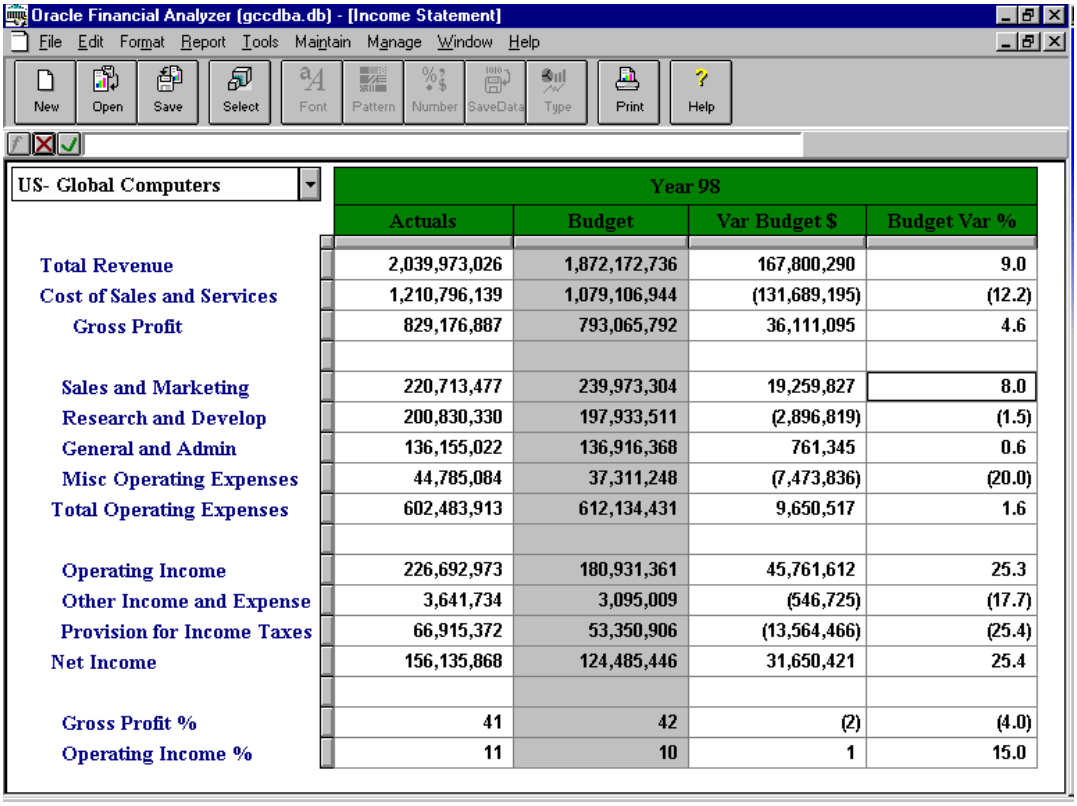

Рисунок 2.2 – Интерфейс приложения Oracle Financial Analyzer

В системе OFA могут работать сотрудники, находящиеся в различных городах, что очень важно для компаний с множеством филиалов. Правда, для этого необходимо установить в каждом филиале свой сервер, через который будут передаваться данные.

В системе есть гибкий контроль доступа к данным. Например, можно настроить программу таким образом, что руководитель будет видеть только агрегированные показатели по подразделениям, а подчиненный — только свои показатели и показатели в целом по компании. Можно также разделять финансовую модель на подмодели по подразделениям.

Однако у программы есть и недостатки. Интерфейс ее не очень нагляден, поэтому работа в OFA требует сложной и дорогостоящей настройки. Кроме того, система не полностью переведена на русский язык: практически весь раздел помощи доступен только в английском варианте. OFA имеет слишком обширный функционал, и использование ее обычным менеджером или владельцем малого предприятия будет запутанным, а также потребуется немало времени чтобы обучиться работать в системе.

В OFA не развиты средства документирования модели, то есть разработанную модель сложно описать с помощью встроенных инструментов, например, получить условную схему модели в виде рисунка или таблицы. Это серьезная проблема, которая встает особенно остро при смене персонала: если в компании нет аналитика, который занимается описанием всех

моделей, объяснить новому сотруднику подробности вашей бюджетной инфраструктуры будет трудно.

Наконец, в OFA полностью отсутствуют встроенные средства, позволяющие вести документооборот. Учитывая, что система рассчитана на большое количество пользователей, из-за этого сразу появляются определенные трудности: например, начав работать с бюджетом, вы не знаете, внес ли туда свои данные тот или иной отдел. Поэтому для организации коллективной работы необходимо использовать дополнительное программное обеспечение. Причем простыми MS Outlook или Lotus Notes тут не обойтись; придется ставить специализированную систему документооборота, например Documentum.

Красный директор

Программный продукт предназначен для использования в повседневной деятельности менеджерами всех уровней и финансовыми директорами.

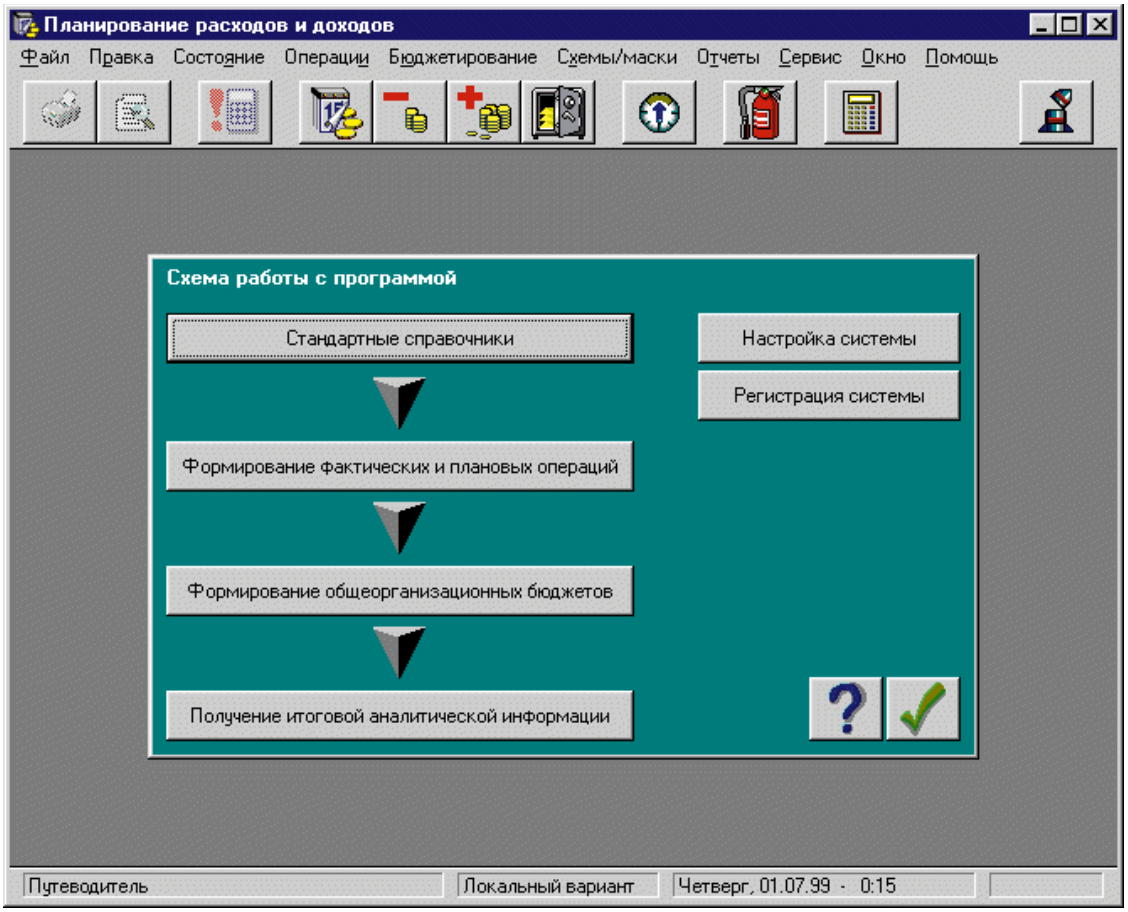

Рисунок 2.3 – Интерфейс приложения «Красный директор»

«Красный директор» — решение для небольших, возможно, для средних предприятий. Для ситуации, когда финансовый директор компании единолично строит бюджет нескольких подразделений, а схемы платежей просты и ведутся вручную, «Красный директор» вполне подойдет как дешевая альтернатива больших комплексов.

Система сделана как коробочный продукт, то есть не требует сложного внедрения, но и особой гибкостью не обладает — это готовое решение, а не инструмент.

Поскольку в «Красном директоре» не было необходимости создавать функционал для групповой работы, согласования бюджетов и многого другого, интерфейс достаточно прост и работу можно начинать сразу после установки. Этому существенно помогает и вынесение всех операций в одно окно с четко прорисованной рекомендованной последовательностью работы.

Бюджетирование, реализованное в программе «Красный директор», позволяет создавать бюджеты недельной (месячной, квартальной, годовой) периодичности для каждого отдельного подразделения организации, направления работы и т. п.

После ввода текущих осуществленных платежей/поступлений по трем видам источников денежных средств (расчетным счетам, кассам и неучтенным кассам) производится автоматический учет произведенных операций с точки зрения исполнения бюджета. Программа автоматически показывает отклонение доходных и расходных статей бюджета от плановых показателей в абсолютном выражении и в процентах.

Данные хранятся в простой файловой базе данных, оперировать с ними довольно сложно, интеграции с какими-либо системами управленческого учета нет, что, впрочем, вполне естественно для системы, ориентированной на небольшие компании.

У разрабатываемого проекта есть несколько преимуществ перед любым из вышеперечисленных для решения поставленных целей. В отличии от Money Care клиентским интерфейсом данной системы является веб-сайт, что делает ее кроссплатформенной. Также в Money Care отсутствует возможность планирования одного бюджета нескольким лицам, а доступный функционал нацелен больше на планирование домашнего бюджета, чем корпоративного. Oracle Financial Analyzer в свою очередь реализует весь необходимый функционал, но минусом в данном случае можно считать его «перенагруженность» другими функциями, которые просто не нужны обыкновенному руководителю небольшой компании или менеджеру. Другими словами использование Oracle Financial Analyzer позволит решить все задачи пользователя, но он будет задействовать только малый процент всего функционала, что не рентабельно. Не говоря об устаревшем и не удобном интерфейсе. Красный директор, в отличии от остальных конкурентов содержит только тот функционал, который в данном случае необходим. Но по сравнению с разрабатываемой системой большим его недостатком является отсутствие возможности многопользовательского планирования бюджета, а также возможности пользователя составлять шаблоны финансовых операций.

Также отмечу что разрабатываемая система, в отличии от всех вышеперечисленных является полностью бесплатной и платформонезависимой.

2.3 Постановка задачи

Целью работы является проектирование системы корпоративного планирования и управления бюджетом. Данная система должна превосходить аналоги по функциональности и в простоте использования. Используя ее пользователь получает возможность создавать счета, добавлять единоразовые и периодические доходы/расходы, и управлять распределением финансовых ресурсов исходя из данных в системе.

Таким образом для достижения поставленной цели система должна поддерживать следующие основные функции:

- регистрацию новых пользователей;
- регистрацию новых предприятий;
- добавление счетов;
- добавление единоразовых операций со счетами (прибыль, убыток, перевод между счетами);
- добавление периодических операций со счетами (прибыль, убыток, перевод между счетами);
- соединение счетов в группы;
- создание шаблонов финансовых операций.

В ходе выполнения работы необходимо выполнить следующие задания:

- провести анализ предметной области;
- провести концептуальное и uml-моделирование предметной области;
- определить архитектуру системы;
- определить используемые программные средства и архитектурные решения для разработки.

Для реализации программного продукта выла выбрана среда разработки Microsoft Visual Studio. Сервер, а также веб-сайт, с помощью которого выполняются все манипуляции с системой будет выполнен в виде ASP .NET MVC приложения с поддержкой API для последующего добавления мобильных клиентов. При выборе базы данных было решено выбрать Microsoft SQL Server. Все компоненты разрабатываемого приложения будут написаны на языке C# с использованием программной платформы .NET Framework.

3 ПЕРЕЧЕНЬ ТРЕБОВАНИЙ К ПРОГРАММНОЙ СИСТЕМЕ

Под требованиями программной системы понимают совокупность свойств, которые должна иметь реализуемая система[4]. Ниже описаны требования к данной программной системе.

3.1 Предназначение разработки

Система планирования корпоративного бюджета предназначена для управления финансовыми средствами в небольших предприятиях. Основная цель системы максимально упростить процесс управления финансами, избавляя пользователя в потребности приобретении специальных навыков использования крупных систем бухгалтерского учета или дополнительном финансовом образовании. Идея заключается в том, что после проведения некоторых конфигураций пользователь будет видеть всю финансовую картину его предприятия перед собой в виде диаграмм и списков и сможет распределить финансовые ресурсы предприятия в несколько кликов. Как говорилось ранее потенциальными пользователями системы являются менеджеры или владельцы небольших предприятий.

3.2 Требования к программному продукту

По первичным требованиям система должна представлять из себя сервер и веб-сайт. Для разработки сервера необходимо использовать ASP.NET MVC. Программный код должен быть написан на языке C#. Для упрощения процесса сопровождения продукта при разработке сервера использовать шаблоны проектирования Dependency Injection и Repository.

Серверная часть является вычислительным ядром системы и должна выполнять все ее основные функции. К основным функциям относится:

- регистрация пользователей системы;
- авторизация пользователей в системе;
- управление конфигурациями пользовательских аккаунтов;
- выдавать разрешение другим пользователям управлять распределением финансов вашего предприятия;

- добавление счетов;
- просмотр распределения средств между счетами и группами счетов в виде диаграмм/списков;
- добавление единоразовых операций со счетами (прибыль, убыток, перевод между счетами);
- добавление периодических операций со счетами (прибыль, убыток, перевод между счетами);
- соединение счетов в группы;
- просмотр истории финансовых операций;
- создание шаблонов финансовых операций.

Также сервер реализует работу веб-сайта, который является пользовательским интерфейсом всей системы. Именно с помощью веб-сайта пользователи системы будут получать доступ к ее функционалу. Сервер, а также веб-сайт должен быть написан на языках C#(Razor), HTML, JavaScript, с использованием технологий ASP .NET MVC. Веб-сайт должен обеспечивать доступ пользователя ко всему функционалу сервера.

В качестве системы управления базами данных необходимо использовать Microsoft SQL Server. Для связи с базой данных использовать Entity Framework.

3.3 Требования к клиенту

Данная система, как любая другая нуждается в обслуживании, поэтому ее пользователи должны соблюдать некоторые требования. К этим требованиям относятся:

- для использования системы необходимо быть зарегистрированным в ней;
- для использования системы необходимо подключение к сети интернет;
- в начале работы необходимо провести конфигурацию системы (создать счета, добавить опрации).

4 ОПИСАНИЕ ПРИНЯТЫХ ПРОЕКТНЫХ РЕШЕНИЙ

4.1 Концептуальное моделирование предметной области

Концептуальная модель — описание структуры проекта, выполненное с использованием таблиц, диаграмм, и других средств, понятных всем людям, работающим над проектом. Основным компонентом концептуальной модели является описание объектов предметной области и связей между ними.

Всех пользователей данной системы можно разбить на две категории:

а) пользователи-владельцы – те пользователи, которые создали компанию на своем аккаунте;

б) остальные пользователи – пользователи которым дали право редактировать бюджет другой компании.

У пользователей обеих категорий выделяются одинаковые требования к функционалу системы:

Потребности основного количества пользователей:

- зарегистрироваться в системе;
- авторизоваться в системе;
- изменить конфигурации своего аккаунта;
- добавить счет;
- просмотреть распределение средств на счетах и группах счетов в виде диаграмм/таблиц/списков;
- добавить единоразовые операции со счетами (прибыль, убыток, перевод между счетами);
- добавить периодические операции со счетами (прибыль, убыток, перевод между счетами);
- добавить группу;
- добавить счет в группу;
- просмотреть историю финансовых операций;
- создать шаблон финансовой операции.

Потребности пользователя-владельца включают в себя потребности обычного пользователя, а также расширяются возможностью выдавать разрешение другим пользователям управлять распределением финансов вашего предприятия.

Исходя из наведенных выше данных составим диаграмму прецедентов, изображенную на рисунке 4.1.

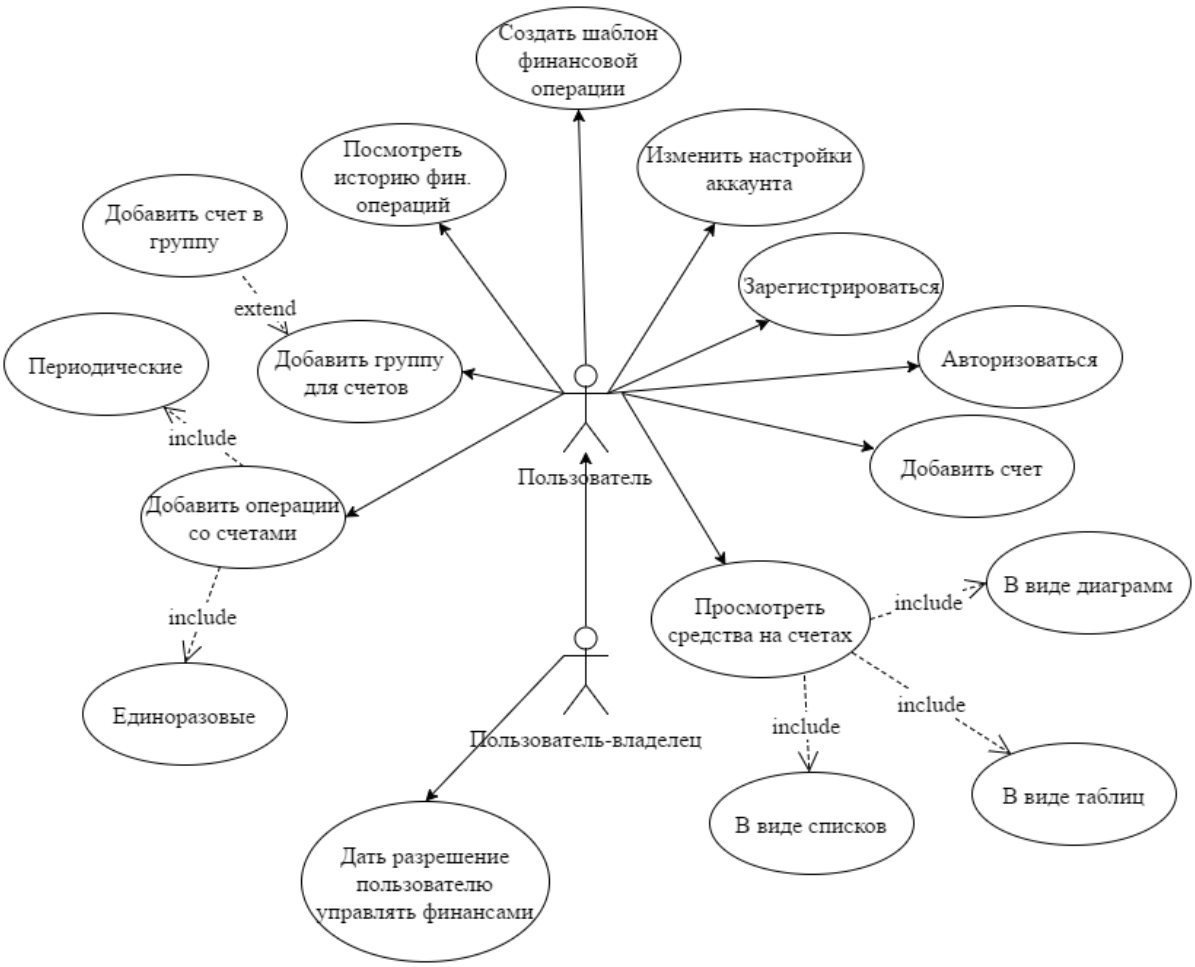

Рисунок 4.1 – Диаграмма прецедентов

Из диаграммы прецедентов видим, что пользователь-владелец имеет тот же функционал что и обычный пользователь (на диаграмме это отображено линией наследования), но обычный пользователь, в свою очередь не может дать разрешение другому пользователю управлять финансами.

Также хочется отметить что функция предоставления добавления группы в счет расширяет функцию добавления группы для счетов это значит, что пользователь не может добавить счет в еще не созданную группу.

4.2 Архитектура

Для реализации проекта было решено использовать трехуровневую архитектуру. Это решение было принято так как трёхуровневая архитектура обеспечивает, как правило, большую масштабируемость (за счёт горизонтальной масштабируемости сервера приложений и мультиплексирования соединений), боольшую конфигурируемость (за счёт изолированности уровней друг от друга).

Трёхуровневая архитектура (трёхзвенная архитектура, англ. three-tier) — архитектурная модель программного комплекса, предполагающая наличие в нём трёх компонентов: клиента, сервера приложений (к которому подключено клиентское приложение) и сервера баз данных (с которым работает сервер приложений)[6].

Как видно на рисунке 4.2 в данной системе клиентом выступают веб-сайт и эмулятор запирающего устройства, далее идет сервер приложений, который управляет работой всей системы, последним элементом является сервер базы данных.

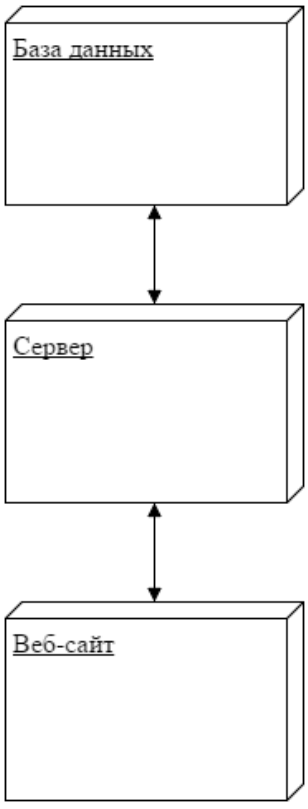

Рисунок 4.2 – Структура архитектуры проекта

Использование трехуровневой архитектуры отделяет часть UI от логической части, в также выводит сервер базы данных на новый уровень, что делает систему безопаснее и менее тесно связанной.

Как дополнительные архитектурные решения, в целях упрощения сопровождения и дальнейшей разработки продукта решено использовать следующие шаблоны проектирования:

Внедрение зависимости (англ. Dependency injection, DI) — процесс предоставления внешней зависимости программному компоненту. Является специфичной формой «инверсии управления» (англ. Inversion of control, IoC), когда она применяется к управлению зависимостями. В полном соответствии с принципом единой обязанности объект отдаёт заботу о построении требуемых ему зависимостей внешнему, специально предназначенному для этого общему механизму;

Репозиторий (англ. Repository) — посредничает между уровнями области определения и распределения данных (domain and data mapping layers), используя интерфейс, схожий с коллекциями для доступа к объектам области определения.

4.3 Проектирование базы данных

Для базы данных было решено использовать MS SQL Server так как все остальные элементы системы также будут реализованы средствами, разработки от Microsoft, что предоставит меньше трудностей при разработке. Также одной из причин является наличие опыта работы с данной системой.

Перед созданием базы данных следует определить основные сущности и их атрибуты[6]. Одной из основных сущностей системы является пользователь и ему соответствуют следующие атрибуты:

- уникальный идентификатор;
- адрес электронной почты;
- имя пользователя;
- пароль.

Не менее важной сущностью является компания. Пользователь создает компанию как имитатор того предприятия, для которого он будет вести расчёт финансов. Для компаний выделены такие основные атрибуты:

- универсальный идентификатор;
- название;
- краткое описание.

Еще одна сущность – счета. Счета в системе представляют из себя хранилища и источники финансовых средств. К примеру, для того, чтобы добавить в систему выплату

зарплаты сотруднику, пользователь создаст счет для этого сотрудника. Для счетов выделены такие атрибуты:

- универсальный идентификатор;
- название;
- количество средств на счету;
- описание.

Также выделена сущность групп, для объединения счетов. Это должна быть дополнительная таблица, содержащая имя группы.

Еще одна основная сущность – операция. Операция проводится над одним или двумя счетами и представляет из себя перенос средств между ними. В данной системе операция может быть единоразовой и периодической, за это отвечает атрибут частоты. Ниже список необходимых для нее атрибутов:

- универсальный идентификатор;
- название;
- количество переведенных средств;
- описание;
- дата операции;
- частота операции;
- счет отправитель;
- счет получатель.

Последней основной сущностью является история. Она должна содержать в себе данные о произведенных операциях. Сохраняться должны как операции, автоматически проведенные системой, так и операции, совершенные пользователем. Эта сущность должна иметь такие атрибуты:

- универсальный идентификатор;
- универсальный идентификатор пользователя;
- универсальный идентификатор операции;
- дата и время совершения операции.

Структура базы данных приведена на рисунке 4.3. Как видно созданные таблицы полностью соответствуют определенным ранее сущностям и их атрибутам.

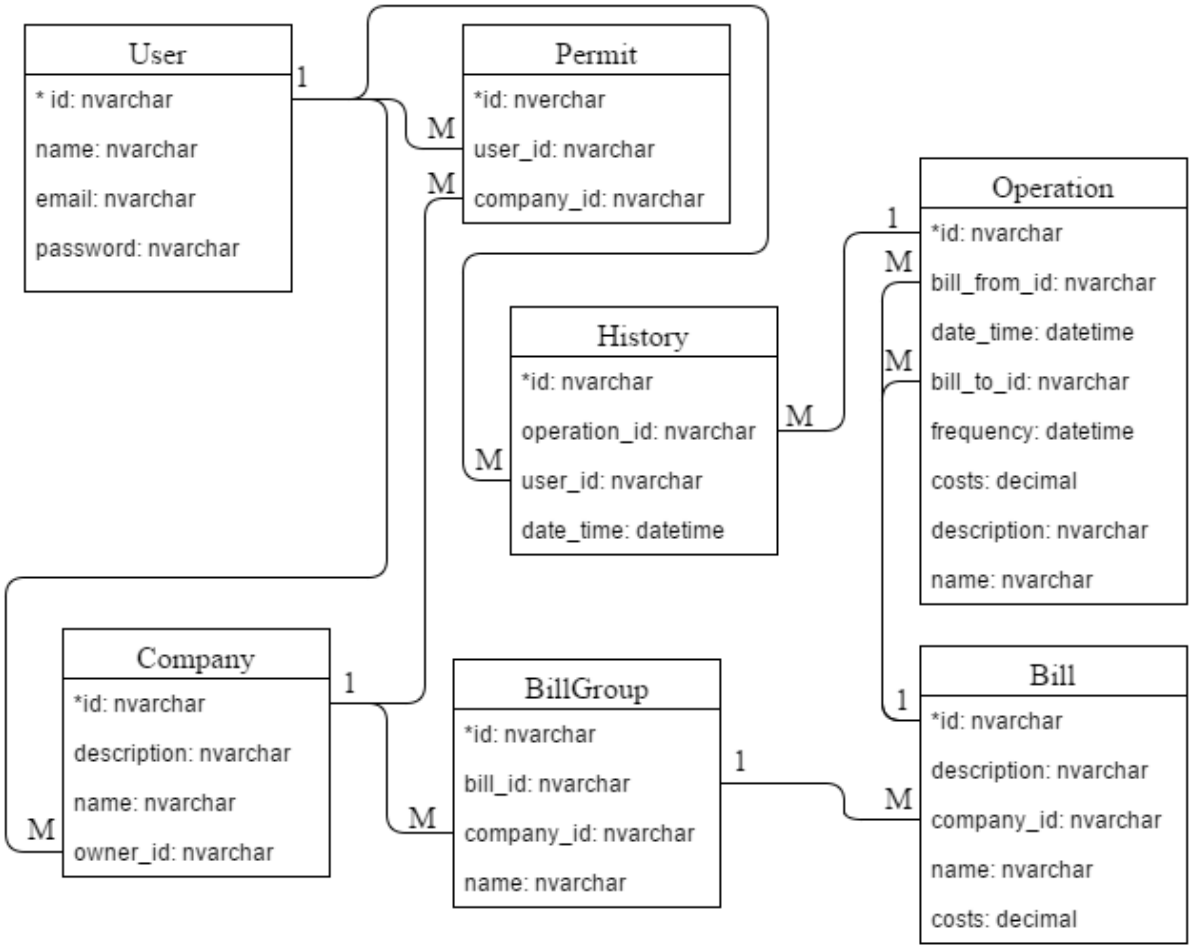

Рисунок 4.3 – Схема базы данных

В ходе проектирования базы данных была создана таблица пользователей (User), таблица компаний (Company), таблица для хранения будущих операций (Operation), таблица для ведения истории произведенных финансовых операций (History), таблица счетов (Bill) и таблица групп счетов (BillGroup). А также добавлена таблица, определяющая возможность пользователя управлять планированием бюджета компании (Permit).

4.4 Выбор технологий разработки

Система полностью реализована на языке C#, используя программную платформу. NET Framework. Для упрощения разработки системы были выбраны и использовались некоторые технологии описанные ниже.

– REST Services (сокр. от англ. Representational State Transfer — «передача репрезентативного состояния») — метод взаимодействия компонентов распределённого

приложения в сети Интернет, при котором вызов удаленной процедуры представляет собой обычный HTTP-запрос (обычно GET или POST; такой запрос называют REST-запрос), а необходимые данные передаются в качестве параметров запроса. Этот способ является альтернативой более сложным методам, таким как SOAP, CORBA и RPC. В широком смысле REST поддерживает концепцию построения распределённого приложения, при которой компоненты взаимодействуют наподобие взаимодействия клиентов и серверов во Всемирной паутине. Этот метод в данной программной системе используется для общения между сервером и клиентскими приложениями (веб-сайтом и эмулятором запирающего устройства) [7].

– Model-view-controller (MVC, «модель-представление-контроллер», «модель-вид-контроллер») — схема использования нескольких шаблонов проектирования, с помощью которых модель приложения, пользовательский интерфейс и взаимодействие с пользователем разделены на три отдельных компонента таким образом, чтобы модификация одного из компонентов оказывала минимальное воздействие на остальные. Данная схема проектирования часто используется для построения архитектурного каркаса, когда переходят от теории к реализации в конкретной предметной области. Основная часть программной системы, сервер – написана с использованием шаблона MVC.

– Microsoft SQL Server — система управления реляционными базами данных (РСУБД), разработанная корпорацией Microsoft. Основной используемый язык запросов — Transact-SQL, создан совместно Microsoft и Sybase. Transact-SQL является реализацией стандарта ANSI/ISO по структурированному языку запросов (SQL) с расширениями. Используется для работы с базами данных размером от персональных до крупных баз данных масштаба предприятия; конкурирует с другими СУБД в этом сегменте рынка.

– ADO.NET Entity Framework (EF) — объектно-ориентированная технология доступа к данным, является object-relational mapping (ORM) решением для .NET Framework от Microsoft. Предоставляет возможность взаимодействия с объектами как посредством LINQ в виде LINQ to Entities, так и с использованием Entity SQL. Для облегчения построения web-решений используется как ADO.NET Data Services (Astoria), так и связка из Windows Communication Foundation и Windows Presentation Foundation. Entity SQL представляет собой язык, подобный языку SQL, который позволяет выполнять запросы к концептуальным моделям в Entity Framework. LINQ to Entitis – это альтернативный интерфейс LINQ API, используемый для обращения к базе данных. Для доступа к данным используется только этот фреймворк.

– Json.NET — фреймворк от NewtonSoft, который уменьшает затраты производительности и сил на серриализацию и десерриализацию объектов. Используется так как при общении между сервером и клиентами данные передаются в формате Json.

– Bootstrap 3 — фреймворк для разработки адаптивных и мобильных web-проектов. С помощю него создан веб-интерфейс системы.

– FusionCharts Free — бесплатный open source компонент для рисования графиков (для десктопных приложений и веб-приложений). С возможностями 22 популярных типов графиков — Column, Line, Pie, Bar, Area, Stacked, Candlestick и Funnel Chart, которые выглядят одинаково во всех браузерах; анимация и интерактив; работает с PHP, ASP.NET, JSP, ColdFusion, Python, RoR, HTML и даже с PowerPoint презентациями; взаимодействие с любыми базами данных; поддержка AJAX.

ВЫВОДЫ

В ходе прохождения практики была спроектирована программная система, которая должна стать удобной и простой системой планирования и расчёта финансов на небольших предприятиях.

В начале работы был проведен анализ предметной области, а также анализ систем-аналогов, которые уже присутствуют на рынке, и успешно реализуют свою продукцию.

Также были определены требования к программному продукту и к потенциальным пользователям.

На этапе проектирования было выполнено концептуальное моделирование предметной области и определена архитектура проекта. Также была спроектирована и разработана модель базы данных и определены технологии, используемые при разработке программного продукта.

Следующим этапом – будет этап разработки, на котором будет реализована данная система в виде сервера и веб-сайта, используя средства и технологии, определенные на этапе проектирования системы.

ПЕРЕЧЕНЬ ССЫЛОК

1. Вільна енциклопедія Википедия [Електронний ресурс] / Wikipedia, the free encyclopedia - Forefront TMG 2010: - Режим доступу: https://en.wikipedia.org/. - Загл. с екрана.

2. Money Care - Bills monitor [Електронний ресурс] Money Care - Bills monitor - Режим доступу: https://itunes.apple.com/ru/app/money-care-bills-monitor/id420669776?mt=8. - Загл. с екрана.

3. Oracle Financial Analyzer [Електронний ресурс] / Oracle Financial Analyzer: - Режим доступу: http://www.oracle.com/technetwork/documentation/ofa-097121.html. - Загл. с екрана.

4. Буч, Г. Язык UML. Руководство пользователя [Текст] / Г. Буч – М.: ДМК Пресс, 2-е издание, 2006. – 248 с.

5. Грекул, В. И. Проектирование информационных систем[Текст] / Н. Л. Коровкина – Изд-во: ИНТУИТ, 2005. – 240с.

6. Гарсия-Молина, Г. Системы баз данных. Полный курс. [Текст]: Пер. с англ./ Г.Гарсия-Молина, Дж. Ульман, Дж. Уидом. – Изд-во: Вильямс, 2003. – 1088 с.

7. Кренке, Г. Теорія й практика побудови баз даних [Текст] / Г. Кренке. - Спб.: Питер, 2001. - 858с.

8. Рихтер, Дж. CLR via C#. Программирование на платформе Microsoft.NET Framework 4.5 на языке C# [Текст]/ Дж. Рихтер. – Изд-во: Питер, 2013. – 896 с.

9. Троелсен Э. C# и платформа .NET. Библиотека программиста [Текст]/ Э. Троелсен. – Изд-во: Питер, 2005. – 796 с.

10. Фримен, А. ASP.NET MVC 5 с примерами на C# 5.0 для профессионалов [Текст]/ Адам Фримен, 5-е издание. – Изд-во: Вильямс, 2014. – 736 с.

11. Мартин, Р. Чистый код: создание, анализ и рефакторинг. Библиотека программиста [Текст]/ Р. Мартин. – Изд-во: Вильямс, 2015. – 464 с.

12. Metanit.com: Сайт о программировании, про создание сайтов и IT-технологии [Електронний ресурс] / metanit.com, 2012-2014. Режим доступу: http://metanit.com/index.php. - Загл. с екрана.

13. Котляров, В.П. Основы тестирования программного обеспечения [Текст]/ В. Котляров. – Изд-во: Бином, 2009. – 288 с.