АННОТАЦИЯ

Настоящий документ содержит расчетно-пояснительную записку к дипломному проекту на тему «Система видео конференцсвязи на облачной платформе Windows Azure».

Программный продукт решает вопрос интерактивного взаимодействия двух и более абонентов, территориально удаленных друг от друга в режиме реального времени.

В данном дипломном проекте проанализировано несколько типов облачных платформ, предлагаемых различными вендорами. Обоснованы главные преимущества системы - это сокращение расходов, увеличение производительности труда и повышение эффективности восприятия и обмена информацией. Предложено высокомасштабируемое сервисное решение для видеоконференцсвязи на платформе Windows Azure.

Оглавление

[ТЕХНИЧЕСКОЕ ЗАДАНИЕ 5](#_Toc327707633)

[1 ИССЛЕДОВАТЕЛЬСКАЯ ЧАСТЬ 8](#_Toc327707634)

[1.1Понятие «облачные вычисления» 9](#_Toc327707635)

[1.2 Предпосылки распространения облачных вычислений 10](#_Toc327707636)

[1.2.1 Унификация и стандартизация 10](#_Toc327707637)

[1.2.2 Экономические преимущества 11](#_Toc327707638)

[1.2.3 Эффективность разработки 12](#_Toc327707639)

[1.3 Разновидность архитектур облачных вычислений 13](#_Toc327707640)

[1.3.1 Инфраструктура как сервис 14](#_Toc327707641)

[1.3.2 Платформа как сервис 16](#_Toc327707642)

[1.3.3 Программное обеспечение как сервис 17](#_Toc327707643)

[1.4 Рынок облачных решений 19](#_Toc327707644)

[1.5 Сравнительный анализ облачных решений с точки зрения модели предоставления сервисов 23](#_Toc327707645)

[2 КОНСТРУКТОРСКАЯ ЧАСТЬ 25](#_Toc327707646)

[2.1 Методология Microsoft Solution Framework 25](#_Toc327707647)

[2.1.1 Создание общей картины приложения 27](#_Toc327707648)

[2.1.2 Планирование 28](#_Toc327707649)

[2.1.3 Разработка 29](#_Toc327707650)

[2.1.4 Стабилизация 30](#_Toc327707651)

[2.1.5 Развертывание 31](#_Toc327707652)

[2.2 Механизмы обработки видео 32](#_Toc327707653)

[2.2.1 IIS Smooth Streaming 32](#_Toc327707655)

[2.2.2 FFMpeg 32](#_Toc327707656)

[2.2.3 Expresssion Encoder SDK 33](#_Toc327707657)

[3 ТЕХНОЛОГИЧЕСКАЯ ЧАСТЬ 34](#_Toc327707658)

[3.1 Хранение данных и доступ к ним в Windows Azure 34](#_Toc327707659)

[3.2 Учетная запись хранилища 34](#_Toc327707660)

[3.3 Хранилище бинарных данных BLOB Storage 34](#_Toc327707661)

[3.3.1 Интерфейс REST объектов Blob 36](#_Toc327707662)

[3.3.2 Blob как список блоков 37](#_Toc327707663)

[3.3.3 Абстракции данных блоков 39](#_Toc327707664)

[3.3.4 Сценарии загрузки блоков 39](#_Toc327707665)

[3.3.5 Windows Azure Queue 43](#_Toc327707666)

[3.3.6 Очередь 47](#_Toc327707667)

[3.3.7 Сообщения. 47](#_Toc327707668)

[3.3.8 Интерфейс REST очереди 48](#_Toc327707669)

[3.3.9 Сценарий поставщик-потребитель 50](#_Toc327707670)

[3.4 Table Storage 51](#_Toc327707671)

[3.4.1 Секционирование таблиц 54](#_Toc327707672)

[3.4.2 Масштабируемость таблицы. 55](#_Toc327707673)

[3.4.3 Транзакции над группами сущностей 55](#_Toc327707674)

[3.4.4 Расположение сущностей 56](#_Toc327707675)

[3.4.5 Программирование таблиц 56](#_Toc327707676)

[3.4.6 Описание класса сущности для таблицы 58](#_Toc327707677)

[3.4.7 Создание таблицы 59](#_Toc327707678)

[3.4.8 Использование REST API 60](#_Toc327707679)

[3.4.9 Параллельные обновления 61](#_Toc327707680)

[3.5 Технология «Windows Communication Foundation» 63](#_Toc327707681)

[4 ЭКОНОМИЧЕСКАЯ ЧАСТЬ 66](#_Toc327707682)

[4.1 Введение 66](#_Toc327707683)

[4.2 Основные этапы проекта разработки программного продукта 66](#_Toc327707684)

[4.3 Расчет трудоемкости разработки программного продукта 67](#_Toc327707685)

[4.3.1 Трудоемкость разработки технического задания 68](#_Toc327707686)

[4.3.2 Трудоемкость разработки эскизного проекта 69](#_Toc327707687)

[4.3.3 Трудоемкость разработки технического проекта 70](#_Toc327707688)

[4.3.4 Трудоемкость разработки рабочего проекта 71](#_Toc327707689)

[4.3.5 Трудоемкость выполнения стадии «Внедрение» 72](#_Toc327707690)

[4.4 Расчет затрат на реализацию программного продукта 73](#_Toc327707691)

[4.4.1 Расчет материальных затрат 73](#_Toc327707692)

[4.4.2 Расчет амортизационных отчислений 73](#_Toc327707693)

[4.4.3 Расчет заработной платы 73](#_Toc327707694)

[4.4.4 Расчет отчислений в социальные фонды 75](#_Toc327707695)

[4.4.5 Прочие расходы 76](#_Toc327707696)

[4.5 Определение цены программного продукта 76](#_Toc327707697)

[4.6 Выводы 77](#_Toc327707698)

[5 ЭКОЛОГИЧЕСКАЯ ЧАСТЬ 78](#_Toc327707699)

[5.1 Введение 78](#_Toc327707700)

[5.2 Анализ факторов воздействия среды на оператора ПЭВМ 78](#_Toc327707701)

[5.3 Общие положения организации рабочего места 79](#_Toc327707702)

[5.4 Обеспечение параметров микроклимата 81](#_Toc327707703)

[5.5 Эргономичность 83](#_Toc327707704)

[5.5.1 Оптимальное размещение оборудования 84](#_Toc327707705)

[5.5.2 Обеспечение электробезопасности 84](#_Toc327707706)

[5.5.3 Обеспечение допустимого уровня шума 85](#_Toc327707707)

[5.5.4 Обеспечение допустимых эргономических характеристики дисплеев 86](#_Toc327707708)

[5.5.5 Обеспечение пожаробезопасности 86](#_Toc327707709)

[5.6 Расчет системы освещения 87](#_Toc327707710)

[5.6.1 Выбор источников света 88](#_Toc327707711)

[5.6.2 Выбор системы освещения 89](#_Toc327707712)

[5.6.3 Выбор осветительных приборов 89](#_Toc327707713)

[5.6.4 Размещение осветительных приборов 89](#_Toc327707714)

[5.6.5 Выбор освещенности и коэффициента запаса 90](#_Toc327707715)

[5.6.6 Расчет системы искусственного освещения 91](#_Toc327707716)

[5.6.7 Утилизация элементов системы искусственного освещения 93](#_Toc327707717)

[5.7 Выводы 93](#_Toc327707718)

[ЗАКЛЮЧЕНИЕ 95](#_Toc327707719)

[СПИСОК ЛИТЕРАТУРЫ 96](#_Toc327707720)

ПРИЛОЖЕНИЕ 1. ГРАФИЧЕСКИЕ МАТЕРИАЛЫ

# ТЕХНИЧЕСКОЕ ЗАДАНИЕ

Разработать систему видеоконференцсвязи с использованием облачных вычислений, как приложение. Приложение предназначено для коммуникации и взаимодействия пользователей системы в режиме реального времени.

1. Основания для разработки.

Основанием для разработки является учебный план кафедры РК6 на 11-й семестр, утвержденный заведующим кафедрой.

1. Назначение разработки.

Система предназначена для внедрения и использования в учебных заведениях. С помощью программного продукта необходимо реализовать новый подход к общению студентов и абитуриентов с руководством ВУЗов. Разработка должна позволять людям с периферии иметь возможность получить ответы на интересующие вопросы непосредственно у представителей университета удалённо, без отрыва от места жительства.

1. Требования к программе или программному изделию.
   1. Требования к функциональным характеристикам.

Разрабатываемая система должна обладать следующими функциями:

а) работать под управлением ОС Windows 7/8;

б) для соединения и обмена данными использовать протокол TCP/IP;

в) интерфейс пользователя должен быть простым и доступным для понимания;

г) иметь гибкую систему настроек;

д) серверная часть должна хранить базу данных пользователей, имеющих доступ к системе и обеспечивать аутентификацию пользователей согласно имеющимся записям;

е) поддержка соединений серверной частью до 500000 пользователей одновременно;

ж) обеспечение серверной частью стабильным видеопотоком независимо от канала передачи данных на стороне пользователя.

* 1. Требования к надежности.

Надежность системы в целом зависит от надежности используемой операционной системы. Серверная часть должна обслуживать без сбоев одновременное подключение и работу до 500000 пользователей. Обе части должны без потерь передавать информацию по каналу связи между клиентом и сервером.

* 1. Условия эксплуатации.

Стандартные условия эксплуатации программных продуктов. Необходимые сотрудники для обслуживания серверной части системы – системный администратор для обслуживания собственно сервера (регистрация и удаление пользователей, добавление и настройка сохраненных записей сеансов связи).

* 1. Требования к составу и параметрам технических средств.

Для нормальной работы клиентской части необходимо:

а) наличие адаптера подключения к интернету;

б) ОС Windows 7/8, установленная на ПК;

в) наличие браузера с поддержкой Silverlight 4/5, например, Internet Explorer версий 6-10, Mozilla Firefox, Google Chrome.

* 1. Требования к информационной и программной совместимости.

Система должна использовать протокол передачи данных HTTP, TCP.

Для хранения информации требуется использование Windows Azure Table Storage, Windows Azure Blob Storage.

В качестве средства разработки требуется использовать интегрированную среду разработки Visual Studio 2010, включающую редактор исходных текстов, компилятор, компоновщик и отладчик.

* 1. Требования к маркировке и упаковке.

Не предъявляются.

* 1. Требования к транспортированию и хранению.

Не предъявляются.

* 1. Специальные требования.

Не предъявляются.

1. Требования к программной документации.

Программной документацией к разрабатываемой системе видеоконференцсвязи является расчётно-пояснительная записка.

1. Этапы разработки.

Разработка ПП разбивается на следующие этапы (стадии): техническое задание, эскизный проект, технический проект, рабочий проект, внедрение.

1. Порядок контроля и приемки.

Испытание представленной модели и контроль качества ее работы провести на базе компьютерного класса кафедры РК 6.

Во время испытаний проверить работу системы последующим позициям:

а) запуск серверной и клиентской частей;

б) соединение клиента(ов) с сервером;

в) проверка правильности соединения;

г) аутентификация пользователя на сервере;

д) проверка изменения состава зарегистрированных пользователей и групп;

е) подключение на сервере сообщений от клиентов;

ж) завершение сеанса связи.

# 1 ИССЛЕДОВАТЕЛЬСКАЯ ЧАСТЬ

Первые идеи об использовании вычислений как общедоступной услуге были предложены в 1960-х годах известным ученым в области информационных технологий, изобретателем языка Lisp, профессором MIT и Стэнфордского университета Джоном Маккарти. Появление первой технологии, близкой к современному пониманию термина облачные вычисления (англ. cloud computing), приписывается компании Salesforce.com, основанной в 1999 году. Именно тогда и появилось первое предложение нового вида business-to-business (сокр. b2b) продукта «Программное обеспечение как сервис» (англ. «Software as a Service», SaaS). Определенный успех Salesforce в этой области вызвал интерес у гигантов ИТ индустрии, которые быстро сообщили о своих исследованиях в области облачных технологий. И уже первое бизнес-решение под названием «Amazon Web Services» было запущено в 2005 году компанией Amazon.com. Следующим свою технологию постепенно ввела Google, начав с 2006 года b2b предложение SaaS сервисов под названием «Google Apps», а затем и модели предоставления платформы как сервиса (сокращённое англоязычное название PaaS) под названием «Google App Engine». И, наконец, свое предложение анонсировала компания Microsoft, презентовав ее на конференции PDC 2008 под названием «Azure Services Platform».

Сам факт высокой заинтересованности крупнейших игроков рынка ИТ демонстрирует определенный статус облачных вычислений как тренда 2009-2010 годов. Кроме того, с релизом Microsoft Azure Service Platform множество экспертов связывает новый виток развития веб-технологий и выход всей сферы облачных вычислений на новый уровень.

Возникновение англоязычного термина начало активно обсуждаться в 2008 году в одной из тематических интернет-конференций. В результате дискуссии выдвигались различные версии, по одной из которых термин облако был впервые использован главой компании Google Эриком Шмидтом в выступлении и получил распространение в средствах массовой информации. Другая популярная версия предполагает, что термин «облачные вычисления» стал широко употребляться в США с 2005 года после запуска компанией Amazon.com проекта Elastic Compute Cloud (Amazon EC2) и широко распространился в бизнесе, среди поставщиков информационных технологий и в научно-исследовательской среде.

Далее рассматривается понятие «облачные вычисления», их характеристики и модели предоставления функциональности, а в конце главы произведено сравнение облачных платформ, предлагаемых в настоящее время ведущими вендорами.

## Понятие «облачные вычисления»

Облачные вычисления – это программно-аппаратное обеспечение, доступное пользователю через Интернет (или локальную сеть) в виде сервиса. Этот сервис позволяет использовать удобный веб-интерфейс для удалённого доступа к выделенным ресурсам (вычислительным ресурсам, программам, данным). Проще говоря, облачные вычисления – это концепция вычислительного облака, согласно которой программы запускаются и выдают результаты работы в окно стандартного веб-браузера на локальном ПК, при этом все приложения и их данные, необходимые для работы находятся на удалённом сервере в Интернете. Компьютер пользователя выступает как рядовой терминал, подключённый к всемирной Сети.

Становлению концепции вычислительного облака способствовали многие открытия в сфере ИТ. Повсеместное распространение высокоскоростных каналов интернет-связи обеспечило интенсивный обмен данными с компьютерами, находящимися в «облаке». Созревание технологий Web 2.0 позволило выполнять функционально насыщенные веб-приложения непосредственно в окне веб-браузера, а не запускать их на локальном компьютере или в локальной сети. Развитие интернет-сервисов, которые предоставляют доступ к своим данным посредством специальных программных интерфейсов (англ. Application Programming Interfaces, API), также содействовало успеху облачных вычислений. Последнее подтверждается тем фактом что, когда разработчик создает приложение, которое обслуживает удаленных пользователей на основе данных из удаленного источника (например, из Facebook), то и промежуточный этап, — обработка данных, — также может осуществляться на удаленной облачной площадке.

Из рисунка 1.1 видно, что облачные вычисления вобрали в себя много идей из предшествующих, поэтому изначально они носят разносторонний характер: их можно понимать как техническую парадигму, как маркетинговый термин, как перспективное направление для НИОКР и академических исследований.

**Рис. 1.1**. Облачные вычисления – синтез ряда технологий и подходов

## 1.2 Предпосылки распространения облачных вычислений

Основными предпосылками для распространения облачных вычислений послужили унификация и стандартизация технологий, экономия за счет масштаба, эффективность разработки.

### 1.2.1 Унификация и стандартизация

Стандартизация позволяет:

* оперировать общей терминологией;
* определять те технологии, использование которых обязательно для создания совместимых решений.

На сегодняшний момент международная организация IEEE ведёт работу над двумя проектами облачных стандартов: IEEE P2301 — перечни стандартов и спецификаций, необходимых для создания совместимых облачных систем; IEEE P2302 — базовые сведения и рекомендации по обеспечению интероперабельности и переносимости в «облаках».

Унификация и стандартизация позволяют людям быстрее взаимодействовать, например, используя глобальную информационную сеть. С помощью Интернета, пользователи могут получать доступ к данным, используя различные устройства: ноутбуки, мобильные устройства, персональные компьютеры.

Различные стандарты в области ИТ замедляют и усложняют выполнение ряда бизнес процессов. Однако стандарты позволяют регламентировать сегодняшнюю свободу поставщиков, каждый из которых пробует сам определять, какие технологии считать, а какие не считать облачными.

### 1.2.2 Экономические преимущества

Серверы очень редко работают с максимальной загрузкой. Большую часть времени они работают с нагрузкой в 10-50%. Такое использование ресурсов неэффективно, поэтому владельцы крупных вычислительных центров именно в силу масштаба своих площадок (насчитывающих десятки тысяч серверов) в состоянии «поделиться» своими ресурсами с теми компаниям, в распоряжении которых находится не очень большое количество серверов (до 100 единиц).

В отличие от своих малых и средних конкурентов, обладатели крупных вычислительных центров в состоянии добиться 5-7-кратной экономии на электроэнергии и сетевой инфраструктуре, многократном снижении стоимости персонала, когда один специалист обслуживает не десятки или сотни, а тысячи серверов.

Экономический эффект систем облачных вычислений связывают с возможностью доступа нескольких пользователей к одним ресурсам, которые распределяются между ними по мере изменения нагрузки (см. рисунок 1.2).



**Рис. 1.2.** Варьирование количеством серверов в облаке

Увеличение количества используемых источников конкретного ресурса пропорционально увеличению потребности в нём или перенос работающего виртуального сервера на более мощный источник. Два и более источников ресурсов снижает риск неработоспособности виртуального сервера в случае выхода из строя какого-либо из серверов, входящих в группу, обслуживающую данного клиента (вместо вышедшего из строя сервера возможно автоматическое переподключение виртуального сервера к ресурсам другого сервера).

### 1.2.3 Эффективность разработки

Популярность облачных вычислений обусловлена не только развитием самих технологий, но и развитием подходов к разработке корпоративных приложений.

Сейчас разработчики предпочитают экономить собственное время и деньги заказчиков. Они не создают приложение «с нуля», а используют многочисленные готовые компоненты. Практически для всех популярных языков создания веб-приложений сегодня существуют функционально богатые конструкторы (англ. Frameworks). Например, Rails для Ruby, Django для Python, Zend Framework для PHP, Web Forms для .NET, Spring для Java.

Преимущества использования облачных вычислений:

* сокращение сроков разработки (то, что раньше делалось месяцами, сейчас делается за несколько недель или даже дней);
* освобождение разработчиков от рутинных операций, связанных с разработкой базовой функциональности;
* возможность большего уделения внимания творческим задачам и «полировке» веб-приложений, благодаря чему сегодняшние популярные веб-сайты по уровню функциональности и удобства намного превосходят свои аналоги пятилетней давности;
* меньшее уделение внимания вопросам, связанным с модернизацией и развитием базовой, не связанной со спецификой конкретного веб-сайта функциональности систем.

Для разработчика облачные вычисления являются продолжением тенденции освобождения его от рутинных и непрофильных задач. На сегодня такие процедуры, как перенос приложения на промышленный сервер и последующая синхронизация изменений между испытательной системой и промышленным сервером, продолжают отнимать у разработчиков много времени и увеличивают вероятность возникновения ошибок и сбоев. Облачные вычисления — в особенности, решения класса PaaS — позволяют свести к минимуму различия между испытательным и промышленным окружением и максимально упростить синхронизацию изменений и, как следствие, разработка занимает еще меньше времени, а ее результаты становятся более предсказуемыми и надежными.

## 1.3 Разновидность архитектур облачных вычислений

В настоящее время общепринято делить облачные платформы на три категории: инфраструктура как сервис (англ. Infrastructure as a Service, IaaS), платформа как сервис (англ. Platform as a Service, PaaS), приложение как сервис (англ. Software as a Service, SaaS). Часто решения IaaS и PaaS называют общедоступными вычислениями (англ. utility computing).

SaaS-решения ориентированы на конечных пользователей, предложения utility computing могут быть востребованы и теми пользователями, которые сами являются поставщиками SaaS-услуг. Таким образом, роль одного и того же участника на рынке облачных вычислений может быть неоднозначна: с одной стороны, он является потребителем IaaS и PaaS-решений, с другой — поставщиком SaaS-услуг.

Чтобы представить более ясную картину об облачных решениях, стоит более подробно рассмотреть каждое решение в отдельности.

### 1.3.1 Инфраструктура как сервис

Инфраструктура как сервис (англ. IaaS or Infrastructureas a Service, IaaS) – это модель, предоставляющая возможность использования облачной инфраструктуры для самостоятельного управления ресурсами обработки, хранения, сетей и другими фундаментальными вычислительными ресурсами. Для ясности стоит привести примеры. Так например, потребитель может устанавливать и запускать произвольное программное обеспечение, которое может включать в себя операционные системы, платформенное и прикладное программное обеспечение. Потребитель может контролировать операционные системы, виртуальные системы хранения данных и установленные приложения, а также ограниченный контроль набора доступных сервисов, к примеру, межсетевой экран, DNS. Контроль и управление основной физической и виртуальной инфраструктурой облака, в том числе сети, серверов, типов используемых операционных систем, систем хранения осуществляется облачным провайдером.

Несколько лет назад для управления различными типами оборудования требовалось различное ПО управления. Виртуализация позволяет реализовать весь набор функций управления в одной интегрированной платформе.

IaaS – это тот сегмент облачных вычислений, в котором наиболее последовательно воплощаются классические преимущества облачных вычислений: экономия за счет масштабности (чем больше пользователей пользуется ресурсом, тем меньше эксплуатационная стоимость в расчете на одного пользователя), эластичность (возможность конфигурации компонентов облака), модель оплаты – «по счетчику».

Сервис IaaS состоит из трех основных компонентов:

* аппаратные средства (серверы, системы хранения данных, клиентские системы, сетевое оборудование);
* операционные системы и системное ПО (средства виртуализации, автоматизации, основные средства управления ресурсами);
* ПО (например, для управления системами).

Технологии виртуализации IaaS позволяют использовать пользователю оборудование, находящееся в распоряжении провайдера, с возможностью разделения вычислительных мощностей этого оборудования на части, которые соответствуют текущим потребностям бизнеса, тем самым увеличивая утилизацию имеющихся мощностей. Предлагается переход от приобретения, управления и амортизации аппаратных активов к покупке процессорного времени, дискового пространства, сетевой пропускной способности, необходимой для выполнения приложения.

К важнейшим преимуществам IaaS стоит отнести возможность использования лучших архитектур и фреймворков. Например, если раньше каждой компании для реализации необходимой инфраструктуры приходилось "изобретать велосипед" - то сейчас имеется широкий спектр услуг готовых инфраструктур, реализованных с учетом накопленного опыта и знаний.

IaaS избавляет предприятия от необходимости поддержки сложных инфраструктур центров обработки данных, клиентских и сетевых инфраструктур, а также позволяет уменьшить связанные с этим капитальные затраты и текущие расходы.

В настоящее время к глобальным лидерам IaaS можно отнести Amazon.com со своим сервисом Amazon Web Services, Microsoft – Microsoft Windows Server 2008 R2 и Hyper-V, IBM – Enterprise и Enterprise+.

### 1.3.2 Платформа как сервис

Платформа как сервис (англ. Platformas a Service, PaaS) – это модель, предоставляющая возможность использования облачной инфраструктуры для расположения базового программного обеспечения с последующим размещением на нём новых или существующих приложений разработки, тестирования, развертывания и поддержки веб-приложений как услуги.

В состав таких платформ входят инструментальные средства создания, тестирования и выполнения прикладного программного обеспечения (системы управления базами данных, связующее программное обеспечение, среды исполнения языков программирования), предоставляемые облачным провайдером, за исключением разработанных или установленных приложений, а также, по возможности, параметров конфигурации среды (платформы).

Ключевой характеристикой PaaS является применение модели ценообразования – плата за использование. Пользователь платит, когда ему услуга необходима и именно за то, что он использует. Данная гибкая схема ценообразования позволяет в разы снизить затраты.

При использовании PaaS отсутствуют затраты на приобретение, поддержку и модернизацию программного обеспечения и оборудования. Для разворачивания веб-приложений клиенту не нужно приобретать оборудование и программное обеспечение – всё это можно взять в аренду. Поддержкой ПО занимается компания-провайдер.

Важным аспектом доверия к облачным технологиям является отказоустойчивость сервиса, безопасность размещения данных на удалённых серверах, а также маштабируемость сервиса. Масштабируемость (автоматическое выделение и освобождение необходимых ресурсов в зависимости от количества обслуживаемых приложением пользователей), надежность и безопасность уже встроены в PaaS и не требуют дополнительных затрат.

Требования к приложениям, развернутым на основе PaaS:

* автоматическая и надежная поддержка использования в веб-масштабе должна обеспечивать безопасность обмена конфиденциальной информацией и выполнение денежных транзакций;
* должны иметь возможность свободно создавать приложения с поддержкой безопасности данных о клиентах, сетевого трафика, исходного кода (интеллектуальной собственности), даже в случае отказа оборудования, на котором развернута платформа.

Сегодня большинство приложений разрабатываются в одной среде, тестируются в другой среде, а разворачиваются в третьей. Благодаря PaaS, весь перечень операций по разработке, тестированию и разворачиванию веб-приложений можно выполнить в одной интегрированной среде, тем самым исключив затраты на поддержку отдельных сред для отдельных этапов.

Способность создавать исходный код и предоставлять его в общий доступ внутри команды разработки значительно повышает производительность по созданию приложений на основе PaaS. Возможность определения, изменения и отслеживания графиков выполнения задач, областей ответственности, ролей (проектировщики, разработчики, тестеры, QC) на основе прав доступа.

PaaS ориентирована прежде всего на веб-разработчиков. Решения PaaS позволяют упростить разработку и развертывание масштабируемых веб-приложений и сэкономить трудозатраты программистов. Однако абсолютная свобода разработчиков ограничена. Так, не все необходимые для его приложения инфраструктурные компоненты могут быть доступны в «облаке». Например, приложение написано на языке программирования PHP или Java, однако PaaS-сервис ориентирован на язык программирования Ruby.

### 1.3.3 Программное обеспечение как сервис

Программное обеспечение как сервис (англ. Software as a Service, SaaS) – бизнес-модель продажи и использования программного обеспечения, при которой поставщик разрабатывает веб-приложение и самостоятельно управляет им, предоставляя потребителю доступ к программному обеспечению, работающему в облачной инфраструктуре и доступному из различных клиентских устройств или посредством тонкого клиента (например, браузера).

Основное преимущество модели SaaS для потребителя состоит в отсутствии затрат, связанных с установкой, обновлением и поддержкой работоспособности оборудования и работающего на нём программного обеспечения.

Особенности модели SaaS:

* приложение приспособлено для удаленного использования;
* одним приложением пользуется несколько клиентов;
* оплата взимается либо в виде ежемесячной абонентской платы, либо на основе объема операций;
* техническая поддержка приложения включена в оплату;
* модернизация и обновление приложения происходит плавно и прозрачно для клиентов.

В рамках модели SaaS клиенты платят не за владение программным обеспечением как таковым, а за его аренду (за его использование через веб-интерфейс). Для потребителя это означает, что он несет сравнительно небольшие периодические затраты (ему не требуется инвестировать значительные средства в приобретение ПО и аппаратной платформы для его развертывания, а затем поддерживать его работоспособность), чего не скажешь о классической схеме лицензирования ПО.

Подход периодической оплаты предполагает, что если необходимость в программном обеспечении временно отсутствует, то потребитель может приостановить его использование и заморозить выплаты разработчику.

С точки зрения разработчика проприетарного ПО модель SaaS позволяет эффективно бороться с нелицензионным использованием программного обеспечения, поскольку само программное обеспечение не попадает к конечным заказчикам.

Концепция SaaS часто позволяет уменьшить затраты на развёртывание и внедрение систем технической и консультационной поддержки продукта, хотя и не исключает их полностью. Контроль и управление основной физической и виртуальной инфраструктурой облака, в том числе сети, серверов, операционных систем, хранения, или даже индивидуальных возможностей приложения (за исключением ограниченного набора пользовательских настроек конфигурации приложения) осуществляется облачным провайдером.

Следует отметить у SaaS-сервиса ключевые признаки:

* доступ к программному обеспечению, разработанному в соответствии с моделью ПО как услуга, предоставляется удалённо по сетевым каналам и, как правило, через веб-интерфейс, кроме того, могут использоваться тонкие клиенты и терминальный доступ;
* программное обеспечение развёртывается в центре обработки данных в виде единого программного ядра, с которым работают все заказчики;
* программное обеспечение предоставляется на условиях уплаты периодических арендных платежей;
* обслуживание и обновление программного обеспечения выполняется централизованно на стороне поставщика приложения, предоставляемого как услуга (SaaS);
* стоимость технической поддержки обычно включается в арендную плату.

## 1.4 Рынок облачных решений

В настоящее время практически каждая крупная ИТ-компания стала поставщиком облачных вычислений. Однако при более детальном рассмотрении становится понятно, что для разных компаний термин «облачные вычисления» обладает разным содержанием. Для одних компаний это естественное направление развития, для других – направление модернизации бизнеса, для третьих – не более, чем просто маркетинг.

К лидерам на рынке облачных решений можно отнести такие сервисы, как Microsoft Windows Azure, Google App Engine, Force.com, Amazon Web Services.

Windows Azure — это открытая и гибкая облачная платформа, которая позволяет быстро выполнять построение приложений, развертывать их и управлять ими в рамках глобальной сети из центров данных, управляемых корпорацией Microsoft [6]. Клиент может осуществлять построение приложений с помощью любого языка, средства или любой платформы, а также может интегрировать свои общедоступные облачные приложения с существующей ИТ-средой.

Google App Engine — сервис хостинга сайтов и web-приложений на серверах Google с бесплатным именем <имя\_сайта>.appspot.com, либо с собственным именем, задействованным с помощью служб Google.

App Engine представлена в апреле 2008, находится в режиме тестирования, доступны как бесплатные аккаунты: «…до 1 Гб дискового пространства, 10 Гб входящего трафика в день, 10 Гб исходящего трафика в день, 200 миллионов гигациклов CPU в день и 2000 операций отправления электронной почты в день», так и возможность приобретения дополнительных ресурсов [6].

Приложения, разворачиваемые на базе App Engine, должны быть написаны на Python, Java либо Go. Среда исполнения Python включает в себя полную реализацию возможностей самого Python, большинство функций стандартной библиотеки языка, ограниченную версию Django, и т. д.

Предлагается набор API для сервисов хранилища, datastore API, аккаунтов Google, загрузки данных по URL, электронной почты и т. д.

Использование службы аккаунтов Google позволяет быстро начать работу с приложением, нет необходимости проводить отдельную регистрацию учётных данных на каждом сайте. Это также позволяет разработчику не заботиться о реализации ещё одной системы регистрации пользователей специально для своего приложения

Force.com — программная платформа, на которой разработаны Sales Cloud и Service Cloud, предоставляемая подписчикам для самостоятельной разработки приложений и расширений для CRM-системы Salesforce.com. Для разработки пользователями используется собственный Java-подобный язык Apex и собственное средство проектирования Visualforce с выходным форматом на основе XML, обеспечивающее генерацию пользовательских HTML/AJAX- и Flex-интерфейсов. Платформа предоставляется исключительно по подписке, в рамках концепции PaaS. В зависимости от уровня подписки доступны различные технические возможности. Так, в бесплатной версии подписчики могут создать не более десяти сущностей, а в неограниченной версии с ценой $75 на пользователя в месяц — до 2000. Подписчики могут размещать разработанные приложения на платформе Force.com в специальном каталоге — App Exchange — и предлагать свои разработки другим заказчикам, в том числе на коммерческой основе.

В качестве системы управления базами данных платформа использует три реплицируемых кластера Oracle RAC из восьми узлов каждый, кластеры расположены в трёх удалённых друг от друга центрах обработки данных. В одной схеме Oracle Database обрабатываются данные сразу нескольких компаний-подписчиков, используется стабильная схема данных, предварительно подготовленная к расширению дополнительными объектами таким образом, что в одних и тех же таблицах хранятся данные различных подписчиков, независимо от различия в специфических атрибутах объектов для различных подписчиков. Широко используется секционирование таблиц базы данных. Утверждается, что на момент 2010 года платформу и все системы, на ней работавшие, обслуживали всего пять администраторов баз данных.

Amazon Web Services , AWS — инфраструктура Web Services платформы в облаке, представленная компанией Amazon в начале 2006 года. В данной инфраструктуре представлено много сервисов для предоставления различных услуг, таких как: хранение данных (файловый хостинг, распределённые хранилища данных), аренда виртуальных серверов, предоставление вычислительных мощностей и др.

vCloud Express - это набор услуг, предоставляемых партнерами VMware, по аренде виртуальных машин и учету мощностей. VMware, в свою очередь, предоставляет таким партнерам инструментальные средства для организации "точек продаж".

В таблице 1 приведены основные характеристики сервисных решений.

**Таблица 1.** Основные характеристики сервисов

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Описание | Microsoft  Windows Azure | Google App Engine | Amazon.com,  AWS | Salesforce.com, Forse.com |
| Операционная система | Windows Azure(без права администрирования) | Любой вариант семейства Linux | Выбор ОС зависит от клиента | Своя запатентованная ОС |
| Вычислительные единицы | Web роли, Work роли | Web App Tasks | Amaz | Собственная среда исполнения |
| Хранение данных | Tables, Blobs, Queues и Drives | Blobstore, Datastore, Task Queues | Amazon Simple Storage Service, Amazon Simple Queue Service | Database |
| Реляционная база данных | Microsoft SQL Azure |  | Amazon Relational Database Service |  |
| Механизм передачи данных | Windows Azure CDN |  | Amazon Cloud Front |  |
| Интеграция с традиционным ПО | платформа Windows Azure,  App Fabric Service Bus,  Access Control |  |  | Перенос пользовательских данных в существующие SF.com аккаунты |
| Управление | Windows Azure,  App Fabric Controller,  Management APls | Administration Console | Amazon Management Console,  Amazon Cloud Watch |  |
| Ценовая политика: вычислительная способность/ пропускная способность/ хранение данных | от 0.12$ в час/вых.0.15$ за Гб и вх. 0.10$ за Гб/0.15$ за Гб в мес. | от 0.10$ в час/ вых.0.12$ за Гб и вх. 0.10$ за Гб/0.15$ за Гб в мес. | от 0.12$ в час для Windows и 0.085$ в час для Linux/вых.0.15$ за Гб и вх. 0.10$ за Гб/0.15$ за Гб в мес. | цена на сервис будет зависеть от количества пользователей |

В основном конкуренты Windows Azure являются сервисы, предоставляющие либо инфраструктуру, либо программное обеспечение. Amazon’s AWS и VMwarev Cloud Express являются примерами инфраструктуры в качестве сервиса. Они предлагают пользователю «чистую» виртуальную машину с полным административным контролем над операционной системой. И Google App Engine, и Force.com следует относить скорее к категории платформы в качестве сервиса, предлагающие пользователю лишь приложения для запуска на платформе. Windows Azure сочетает в себе стратегии PaaS и SaaS: в качестве платформы давая возможность пользователю пользоваться приложением, при этом пользователь не имеет никаких административных прав, и, с другой стороны, в качестве инфраструктуры.

## 1.5 Сравнительный анализ облачных решений с точки зрения модели предоставления сервисов

Сравнительный анализ возможностей существующих облачных сервисов можно сделать исходя из таблицы 2.

**Таблица 2.** Возможности сервисов на основе облачных технологий

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Возможности | Microsoft Windows Azure | Google AppEngine | Amazon.com  AWS | Salesforce.com  Forse.com |
| Запуск существующего традиционного ПО |  |  | Да |  |
| Интеграция с существующим традиционным ПО | Да |  |  |  |
| Создание маштабируемого веб-приложения с реляционными БД | Да |  | Да |  |
| Создание высокомасштабируемых веб-приложений | Да | Да | Да | Да |
| Создание высокопроизводительных распределённых приложений | Да |  | Да |  |
| Создание высокомасштабируемых веб-приложений с фоновой обработкой данных | Да | Да | Да |  |

По разным оценкам ведущих поставщиков информации в сфере ИТ совокупный объем мирового рынка ПО в 2013 г. достигнет примерно $ 9 млрд. Потенциальный рынок платформенных сервисов типа Azure в России к 2013 г. можно оценить примерно в $100 млн.

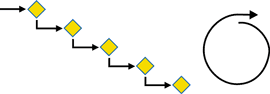
Весь рынок SaaS гораздо больше платформенной части – и в России, и в мире. Его объем оценивается в $12 млрд. Платформенная часть при этом сейчас составляет около 3%, но по мере того, как будет развиваться идея облака, она будет увеличиваться. В сложившейся ситуации очевидно, что доля платформ вроде Azure в России хоть и будет расти, но несколько более низкими темпами, чем по всему миру.

# 2 КОНСТРУКТОРСКАЯ ЧАСТЬ

В целях создания качественного и поддерживаемого программного продукта было принято решение использовать гибкую методологию разработки дипломного проекта. Гибкая методология разработки — серия подходов к разработке программного обеспечения, ориентированных на использование итеративной разработки и динамическое формирование требований и обеспечение их реализации в результате постоянного взаимодействия внутри самоорганизующихся рабочих групп, состоящих из специалистов различного профиля. Существует несколько методик, относящихся к классу гибких методологий разработки и среди них была выбрана методика, предлагаемая компанией Микрософт, Microsoft Solution Framework.

## 2.1 Методология Microsoft Solution Framework

Говоря о моделях процессов жизненного цикла проектов разработки ПО, в первую очередь нужно упомянуть о двух основных схемах: водопадной и спиральной (см. рис. 2.1), которые отражают два разных подхода к организации этих работ.

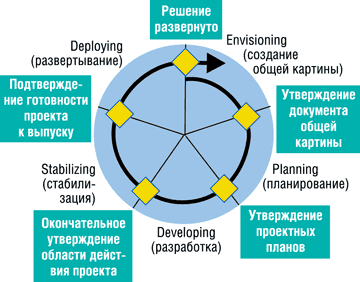


**Рис. 2.1.** Водопадная и спиральная модели разработки

Водопадная модель предусматривает четкий переход от этапа к этапу: работы следующего этапа начинаются только после выполнения всех задач предыдущего. Такой стиль подходит для проектов, в которых проектные требования четко определяются заранее и с большой вероятностью не будут корректироваться потом. Данная схема организации разработки очень удобна с точки зрения управления проектом, так как позволяет четко сформулировать состав и обязанности его участников и контролировать графики выполнения проекта.

Спиральная модель обычно ориентируется на крайний случай, когда требования и параметры проекта непрерывно корректируются, а новые требования формулируются лишь по мере необходимости выполнения конкретных работ. Такая схема часто ассоциируется с понятием "экстремальной разработки"; при этом исполнитель и заказчик работают в постоянном тесном сотрудничестве, клиент привлекается на каждом этапе, формулируя свои соображения по поводу созданных компонентов. Однако при такой организации очень велик риск, что процесс разработки выйдет из-под контроля, поэтому реально данная модель используется лишь в относительно небольших проектах.

Однако проблема заключается в том, что чаще всего все требования на задание действительно практически невозможно определить заранее, к тому же даже сформулированные требования подвергаются коррекции. Но тогда требуется повысить уровень управляемости проектом, без чего создание сложного ПО просто невозможно. Компромисс между этими противоречивыми требованиями и предоставляет модель процессов MSF, в которой сочетаются водопадная и спиральная модели разработки: проект реализуется поэтапно, с наличием соответствующих контрольных точек, а сама последовательность этапов может повторяться по спирали, что и показано на рисунке 2.2.



**Рис. 2.2.** Этапы и контрольные точки модели MSF

В этом случае планирование на основе промежуточных контрольных точек и предсказуемость водопадной модели дополняются наличием обратной связи с заказчиком и творческим подходом к решению задач, характерными для спиральной модели. Таким образом, модель процессов MSF позволяет создавать и развертывать решения уровня предприятия, обеспечивая прогнозируемость хода проектов и учитывая реальные условия их выполнения.

### 2.1.1 Создание общей картины приложения

На этом этапе решаются следующие основные задачи:

* определение состава команды;
* структуры проекта;
* определение бизнес-целей;
* оценка существующей ситуации;
* создание документа общей картины и области действия проекта;
* определение требований и профилей пользователей;
* разработка концепции решения;
* риска;
* закрытие этапа.

На этапе выделяются две промежуточные контрольные точки: "Организован костяк команды" и "Создана общая картина решения".

**Для точки «Организован костяк команды»** не обязательно нужен полный поименный список команды, но в документе структуры проекта должны быть определены роли и обязанности каждого члена команды, а также описана иерархия отчетности и ответственности в группе, каналы взаимодействия с заказчиком и общая структура команды.

**Если говорить о точке «Создана общая картина решения», то р**ечь идет о разработке концепции решения, которым должна руководствоваться команда для достижения долгосрочных бизнес-целей проекта. Область действия проекта определяет, что включается в контекст проекта, а что выходит за его рамки. На этой временной точке речь идет о создании первой версии документа, который находится в стадии рецензирования участниками команды и согласования заказчиком.

Этап завершается контрольной точкой "Утверждение документа общей картины и области действия проекта".

Применительно к описываемому дипломному проекту на данном этапе мной и научным руководителем была поставлена цель, выявлены основные бизнес процессы, а так же проработана общая концепция решения.

### 2.1.2 Планирование

На этапе планирования команда решает, что нужно разработать, и формирует планы реализации продукта. Готовится функциональная спецификация, создается проект решения, детализируются планы работы, выполняется оценка стоимости и сроков получения результатов.

На этом этапе проводится анализ требований, которые делятся на бизнес-требования, пользовательские, функциональные и системные требования. После сбора и анализа требований команды разрабатывается проект решения, определяются профили пользователей, после чего формируются сценарии применения решения, выполняемые пользователями одного типа, а затем определяются варианты использования системы.

Этап состоит из трех стадий: концептуальное, логическое и физическое проектирование. На стадии **концептуального** проектирования задача рассматривается с точки зрения пользовательских и бизнес-требований и заканчивается определением набора сценариев использования системы. При **логическом** проектировании задача рассматривается с точки зрения проектной команды, решение представляется в виде набора сервисов. И уже на стадии **физического** проектирования задача рассматривается с точки зрения программистов, уточняются используемые технологии и программные интерфейсы.

В ходе данного этапа решаются такие задачи:

* разработка проекта и архитектуры решения;
* функциональной спецификации;
* планов проекта;
* календарного графика;
* создание среды разработки, тестирования и пилотной эксплуатации;
* этапа.

Контрольные точки этапа планирования связаны с достижением следующих результатов: функциональная спецификация, план управления рисками, определение среды разработки и тестирования, генеральный план и календарный график проекта.

Результаты данного этапа служат для принятия компромиссных решений в дальнейшем.

Исходя из описанного выше, на данном этапе была создана архитектура решения, а также разработан календарный план, в соответствии с которым шла работы над дипломным проектом.

### 2.1.3 Разработка

На этапе разработки создается решение, в том числе пишется и документируется код. В начале этого этапа команда проверяет выполнение всех задач, характерных для предыдущих этапов, а затем приступает к решению следующих задач:

* создание прототипа приложения;
* разработка программных компонентов приложения;
* создание решения (последовательность ежедневных или более частых сборок приложения);
* закрытие разработки (реализация всех функций, поставка кода и документации).

Результаты этапа предполагают следующие элементы: исходный текст кода и исполняемые файлы, сценарии установки и конфигурации для развертывания, окончательная функциональная спецификация, элементы поддержки решения, спецификации и сценарии тестирования.

Основная контрольная точка этапа - "Окончательное утверждение области действия проекта". В этот момент все функции продукта готовы и прошли тестирование в рамках своего модуля. После этого продукт готов к внешнему тестированию и стабилизации. Кроме того, заказчики, пользователи, сотрудники службы поддержки и сопровождения, а также ключевые участники проекта могут предварительно оценить продукт и указать все недостатки, которые нужно устранить до его поставки.

Прототип системы видеоконференцсвязи был построен на этом этапе работы над дипломным проектом, а так же разработаны ключевые компоненты решения.

### 2.1.4 Стабилизация

Данный этап - подготовка к выпуску окончательной версии продукта, доведение его до заданного уровня качества. Здесь выполняется комплекс работ по тестированию (обнаружение и устранение дефектов), а также проверяется сценарий развертывания продукта и проводится пилотная эксплуатация.

Тестирование подразумевает следующие основные виды работ:

* тестирование компонентов;
* баз данных;
* тестирование инфраструктуры;
* защиты;
* тестирование интеграции;
* анализ удобства работы с продуктом;
* тестирование (включая анализ ресурсоемкости и производительности);
* тестирование;
* ведение отчетности по тестированию.

Когда решение становится достаточно устойчивым, проводится его пилотная эксплуатация в тестовой среде с привлечением пользователей и применением реальных сценариев работы.

Один из главных показателей этапа стабилизации - число обнаруженных ошибок. Сходимость этой величины в сторону устойчивого уменьшения - признак того, что близится завершение работ над продуктом. Важнейшая промежуточная контрольная точка - появление версии, в которой усилиями самой проектной команды не обнаружено ни одной ошибки. Далее следуют выпуски кандидат-релизов продукта для их исследования в условиях пилотной эксплуатации. Завершающая контрольная точка - подтверждение готовности продукта к выпуску и полноценному развертыванию в промышленной среде.

В процессе работы над проектом постоянно производилось тестирование программной составляющей решения, а так же в конце было произведено интеграционное тестирование все созданной системы.

### 2.1.5 Развертывание

На этом этапе выполняется установка решения и необходимых компонентов окружения, проводится его стабилизация в промышленных условиях и передача проекта в руки группы сопровождения. Кроме того, анализируется проект в целом на предмет уровня удовлетворенности заказчика. Основные контрольные точки данного этапа таковы:

* развернуты основные компоненты;
* развернуто решение в целом;
* решение стабилизировано;
* развернуто и передано в эксплуатацию заказчику.

Следует подчеркнуть, что момент завершения данного этапа бывает достаточно сложно формально определить, так как выявление неполадок может продолжаться и в ходе промышленной эксплуатации. Именно поэтому необходимо четко сформулировать критерии для завершающей контрольной точки этапа развертывания и не пытаться отладить абсолютно все.

Программное решение было развернуто на платформе Windows Azure, детальный процесс был описан мною в журнале MSDeveloper [5].

## 2.2 Механизмы обработки видео

### В этой части рассмотрены три механизма обработки видеопотока.

### 2.2.1 IIS Smooth Streaming

Технология Smooth Streaming - это технология адаптивной трансляции потокового видео по протоколу HTTP.

Smooth Streaming является приложением для Internet Information Services (IIS) 7.0 позволяет создать на базе веб-сервера вещание через технологию Smooth Streaming, с существующей готовой клиентской инфраструктурой на базе Microsoft Silverlight.

Smooth Streaming обеспечивает высокое качество просмотра с возможностью массового масштабирования в сети и распределения транслируемого содержимого, что делает качественной трансляцию видео через интернет.

Smooth Streaming использует простую, но мощную концепцию доставки небольших фрагментов контента (обычно за две секунды) и проверки того, что каждый из них имеет надлежащее время и воспроизводится на ожидаемом уровень качества. Если фрагмент не отвечает этим требованиям, следующий фрагмент будет доставлен на более низком уровне качества. И наоборот, когда позволят условия, качество последующих фрагментов будет на более высоком уровне. После того как сервер [IIS](http://ru.wikipedia.org/wiki/IIS) получает запрос о трансляции видео, он будет динамически создавать виртуальные кэшируемые фрагменты из видео файлов. В результате конечный пользователь увидит видео в наилучшем качестве в зависимости от своей пропускной способности.

### 2.2.2 FFMpeg

FFmpeg — это набор свободных библиотек с открытым исходным кодом, которые позволяют записывать, конвертировать и передавать цифровые аудио- и видеозаписи в различных форматах. Он включает libavcodec — библиотеку кодирования и декодирования аудио и видео и libavformat — библиотеку мультиплексирования и демультиплексирования в медиаконтейнер. Название происходит от названия экспертной группы MPEG и FF, означающего fast forward.

FFmpeg разработан под ОС на основе Linux, однако может быть скомпилирован под многие другие операционные системы. Разработчики не выпускают релизов и рекомендуют использовать последнюю версию из Git. Распространяется под лицензиями GNU LGPL или GNU GPL.

### 2.2.3 Expresssion Encoder SDK

Microsoft Expression Encoder позволяет кодировать аудио и видео высокого качества в форматы подходящие для прямых («живых») трансляций и по требованию,  для различных устройств и веба. С помощью Expression Encoder можно импортировать файлы различных форматов и перекодировать их в такие форматы как Windows Media, MP4 или IIS Smooth Streaming файлы. Также можно записывать видео с мониторов пользователей и использовать его для презентации.

Microsoft Encoder SDK также включает следующие изменения в API, которые охватывают множество возможностей. Например, форматы контента для публикации добавлены в возможность прямой трансляции для очевидного отличия транслируемого формата медиа контента. Digital Right Management (DRM)  добавлена в Transcoding и Live Broadcasting кодирование.

**2.3 Выводы**

В главе рассмотрена методология Microsoft Solution Framework, показано из каких этапов состоял процесс наисания дипломного проекта, описано какие процессы были выполнены на каждом из этапов.

На качество работы программного продукта, построенного на выбранной платформе, оказывает существенной влияние механизм обработки и загрузки видеопотока. Поэтому были исследованы основные программные продукты для этой цели и средства разработки, поставляемые с ними.

Выбор был сделан в пользу решения Microsoft Expression Encoder в связи с предлагаемыми возможностями, легкостью освоения и возможности использование GPU для обработки видеопотока.

# 3 ТЕХНОЛОГИЧЕСКАЯ ЧАСТЬ

## 3.1 Хранение данных и доступ к ним в Windows Azure

Служба хранения Windows Azure состоит из следующих компонентов:

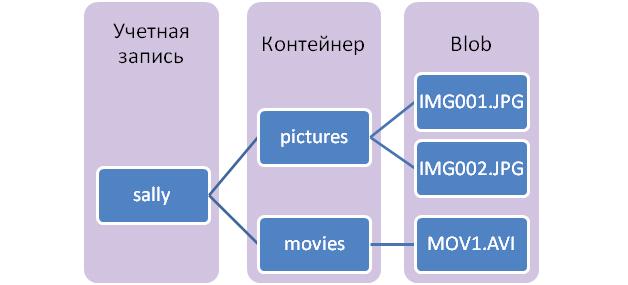
* блоб-объекты — используются для хранения неструктурированных двоичных и текстовых данных;
* очереди — используются для хранения сообщений, доступных клиентам, и обеспечения надежного обмена сообщениями между экземплярами ролей;
* таблицы — используются для хранения нереляционных структурированных данных;
* диски Windows Azure — используются для подключения томов NTFS, доступных для кода, выполняемого в службе Windows Azure.

## 3.2 Учетная запись хранилища

Любой доступ к Windows Azure Storage осуществляется через учетную запись хранилища. Это самый высокий уровень пространства имен для доступа к очередям и их сообщениям. Для использования Windows Azure Storage пользователь должен создать учетную запись хранилища. Выполняется это через веб-интерфейс портала Windows Azure Portal. При создании учетной записи пользователь получает 256-разрядный секретный ключ, который впоследствии используется для аутентификации запросов этого пользователя к системе хранения. В частности, с помощью этого секретного ключа создается подпись HMACSHA256 для запроса. Эта подпись передается с каждым запросом данного пользователя для обеспечения аутентификации через проверку достоверности подписи HMAC. Учетная запись может иметь множество очередей.

## 3.3 Хранилище бинарных данных BLOB Storage

Общее представление хранилища Windows Azure Blob представлено на рисунке 3.1.



**Рис. 3.1.** Общее представление хранилища Blob

Контейнер Blob обеспечивает группировку набора объектов blob. Область действия имени контейнера ограничена учетной записью.

Политики совместного использования задаются на уровне контейнера. В настоящее время поддерживаются «Public READ» («Открытое чтение») и «Private» («Закрытый»). Если для контейнера определена политика«Public READ», все его содержимое открыто доступно для чтения без необходимости аутентификации. Политика «Private» означает доступ с аутентификацией, т.е. только владелец соответствующей учетной записи имеет доступ к объектам blob этого контейнера.

С контейнером могут быть ассоциированы метаданные, которые задаются в виде пар <имя, значение>. Максимальный размер метаданных контейнера – 8КБ.

Существует возможность получения списка всех объектов blob контейнера.

Blob-объекты blob хранятся в контейнерах Blob Container и их область действия ограничена этими контейнерами. Каждый blob может быть размером до 50ГБ и имеет уникальное в рамках контейнера строковое имя. С blob могут быть ассоциированы метаданные, которые задаются в виде пар <имя, значение> и могут достигать размера 8КБ для blob. Метаданные blob могут быть получены и заданы отдельно от данных blob.

Для доступа к Windows Azure Blob используется приведенные выше подходы. URI конкретного blob структурирован следующим образом: *http://<учетная запись>.****blob****.core.windows.net/<контейнер>/<имя blob>*

Первая часть имени хоста образована именем учетной записи хранилища, за которым следует ключевое слово «blob». Это обеспечивает направление запроса в часть Windows Azure Storage, которая обрабатывает запросы blob. За именем хоста идет имя контейнера, «/» и затем имя blob.

Существуют ограничения именования учетных записей и контейнеров. Например, имя контейнера не может включать символ «/».

### 3.3.1 Интерфейс REST объектов Blob

Любой доступ к Windows Azure Blob выполняется через стандартные HTTP-команды PUT/GET/DELETE интерфейса REST.

К командам HTTP/REST, поддерживаемым для реализации операций с blob, относятся:

* put Blob – вставить новый или перезаписать существующий blob с заданным именем;
* get blob – получить весь blob или диапазон байтов blob, используя стандартную HTTP-операцию для возвращения диапазона GET;
* delete Blob – удалить существующий blob;
* copy blob – копировать blob из исходного blob в заданный blob в рамках одной учетной записи хранилища. Это обеспечит копирование всего blob, включая его метаданные, свойства и список блоков. Команда Copy Blob может использоваться наряду с Delete Blob для переименования blob, перемещения blob из контейнера в контейнер и для создания резервных копий существующих объектов blob;
* block list – извлечь список блоков, переданных как часть blob.

Все эти операции с blob могут быть выполнены с использованием следующего URL:

*http://<учетная запись>.****blob****.core.windows.net/<контейнер>/<имя blob>*

Один запрос PUT обеспечивает возможность загрузки в облако blob размером до 64МБ. Для загрузки blob, размер которого превышает 64МБ, используется технология загрузки блоками.

### 3.3.2 Blob как список блоков

Один из целевых сценариев Windows Azure Blob – эффективная загрузка объектов blob, размером десятки гигабайт. Windows Azure Blob обеспечивает это следующим образом.

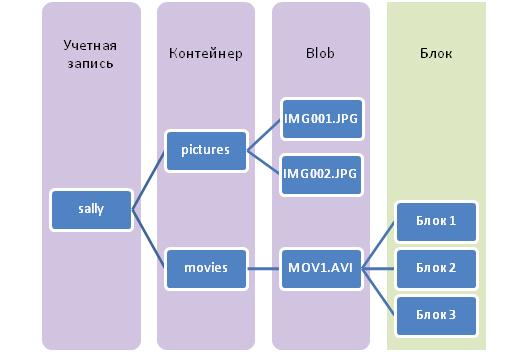
Загружаемый Blob (например, Movie.avi) разбивается на последовательные блоки. Например, ролик размером 10ГБ может быть разбит на 2500блоков по 4МБ, при этом первый блок будет представлять диапазон байтов данных от 1 до4194304, второй блок – от 4194305 до 8388608 и т.д.

Каждому блоку присваивается уникальный ID или имя. Область действия этого уникального ID ограничена именем загружаемого blob. Например, первый блок может быть назван «Block 0001»,второй – «Block 0002» и т.д.

Каждый блок размещается в облаке. Для этого используется операция PUT с указанием представленного выше URL с запросом, определяющим, что это операция размещения блока, и ID блока. Продолжая пример, при размещении первого блока будет указано имя blob «Movie.avi» и ID блока «Block 0001».

Когда все блоки размещены в Windows Azure Storage, подтверждаем список неподтвержденных блоков, переданных, чтобы представить blob, с которым они ассоциированы. Выполняется это с помощью операции PUT с указанием представленного выше URL с запросом, определяющим, что это команда block list. После этого HTTP-заголовок включает список блоков, которые должны использоваться в этом blob. В случае успешного завершения этой операции получаем список блоков, представляющий пригодную для чтения версию blob. Теперь blob может быть считан с помощью описанной выше команды GET Blob.

На рисунке 3.2 представлено место блоков в концепции данных Windows Azure Blob.



**Рис. 3.2.** Общее представление хранилища Blob, добавление блоков

Доступ к объектам blob может осуществляться посредством операций PUT и GET с использованием следующего URL:

*http://<учетная запись>.blob.core.windows.net/<контейнер>/<имя blob>*

В примере, представленном на рисунке 3.2, рисунки со следующими URL могут быть размещены одной операцией PUT:

*http://sally.blob.core.windows.net/pictures/IMG001.JPG*

*http://sally.blob.core.windows.net/pictures/IMG002.JPG*

Те же URL могут использоваться для возвращения объектов blob. Одна операция PUT может обеспечить размещение в хранилище объектов blob размером до 64МБ. Для сохранения объектов blob размером больше 64 МБ и вплоть до 50 ГБ необходимо сначала разместить все блоки посредством соответствующего количества операций PUT и затем, с помощью все той же операции PUT, передать список блоков, чтобы обеспечить пригодную для чтения версию blob. В примере, который иллюстрирует рисунке 3.2, только после размещения всех блоков и подтверждения их принадлежности blob посредством списка блоков blob может быть считан с использованием следующего URL:

*http://sally.blob.core.windows.net/pictures/MOV1.AVI*

Операции GET всегда выполняются на уровне blob и не предполагают использования блоков.

### 3.3.3 Абстракции данных блоков

Каждый блок идентифицирует ID блока размером до 64 байт. Область действия ID блока ограничена именем blob, поэтому разные объекты blob могут иметь блоки с одинаковыми ID. Блоки неизменны. Каждый блок может быть размером до 4 МБ, и один blob может включать блоки разного размера. Windows Azure Blob обеспечивает следующие операции уровня блока:

* put block – загрузить блок blob. Успешно загруженный посредством операции PUT block блок не является частью blob до тех пор, пока это не будет подтверждено списком блоков, загружаемым операцией PUT block list;
* put block list – подтвердить blob через предоставление списка ID блоков, его составляющих. Указанные в этой операции блоки должны быть уже успешно загружены через вызовы PUT. Порядок блоков в операции PUT block list обеспечит пригодную для чтения версию blob;
* get block list – извлечь список блоков, переданный ранее для blob операцией PUT block list. В возвращаемом списке блоков указываются ID и размер каждого блока. Эта функция также может использоваться для извлечения списка неподтвержденных блоков, в который входят блоки, переданные для blob посредством вызовов Put Block, но до сих пор не подтвержденный командой PUT block list.

Block ID является частью метаданных, которые можно отслеживать для каждого блока. Один из примеров применения идентификатора блока –использование в качестве Block ID хэша содержимого блока. Это позволяет контролировать целостность данных для каждого блока данных, извлекаемого из системы.

### 3.3.4 Сценарии загрузки блоков

Загрузка blob в виде списка блоков обладает следующими преимуществами:

* возможность продолжения – для каждого блока в отдельности можно проверить успешность загрузки, в случае сбоя повторить попытку загрузки и продолжить выполнение с этого момента;
* параллельная загрузка – загрузка блоков может выполняться параллельно, что обеспечивает сокращение времени загрузки очень больших объектов blob;
* загрузка не по порядку – блоки могут загружаться в произвольном порядке. Значение имеет лишь порядок блоков в списке операции PUT block list. Список блоков в операции PUT block list определяет пригодную для чтения версию blob;
* ассоциация Block ID с каждым блоком в виде метаданных – при передаче blob посредством блоков для подтвержденных блоков сохраняются ID и размеры.

Данные могут быть извлечены позже операцией Get Block list. Приложение может интерпретировать эти данные как метаданные, которые могут быть заданы и ассоциированы с каждым блоком хранящегося blob. Таким образом, в качестве части ID блока можно сохранить MD5 содержимого блока. Тогда если приложению необходимо проверять контрольную сумму MD5 для заданных диапазонов байт, возвращенных командой GET blob, сначала с помощью GET можно получить список блоков и получить блоки через соответствующие заданные диапазоны, и затем выполнить контроль суммы MD5 для каждого блока, загруженного с помощью команды GET blob с заданным диапазоном.

Используя представленный на рисунке 3.4 пример, можно описать различные сценарии, возможные при использовании блоков для загрузки объектов blob.

|  |  |
| --- | --- |
| http://i.msdn.microsoft.com/dynimg/IC367443.jpg | **Put Block**(Block Id1);  **Put Block**(Block Id3);  **Put Block**(Block Id4);  **Put Block**(Block Id2);  **Put Block**(Block Id4);  **Put Block List**(Block Id2, Block Id3, Block Id4); |

**Рис. 3.4.** Сценарий загрузки блоков

Сценарий 1. Загрузка блоков с одинаковыми ID блока. Для одного blob загружаются блоки с одинаковыми ID блока, при формировании окончательно версии blob в операции PUT block list из всех блоков с одинаковым ID будет использоваться только загруженный самым последним. В примере выше загружаются два блока с Block Id4,и только второй из них будет использоваться в окончательном списке блоков blob.

Сценарий 2. Загрузка блоков в произвольном порядке. Блоки могут загружаться в порядке, отличном от указанного в окончательном списке блоков blob. В примере выше в окончательном списке блоки располагаются в порядке Block Id2, Block Id3 и Block Id4, но загружались они в другой последовательности. Упорядочивание данных blob (через операцию GET) в пригодную для чтения версию выполняется соответственно списку, указанному в операции PUT block list. Таким образом, если извлечь и прочитать blob от начала до конца, будет получено содержимое блока Block Id2,затем Block Id3 и Block Id4.

Сценарий 3. Неиспользуемые блоки. Некоторые блоки не используются, более того, некоторые блоки могут никогда не войти в окончательный список блоков blob. Эти блоки будут удалены системой в процессе сборки мусора. В рассматриваемом примере такими блоками являются Block Id1 и первый блок с ID Block Id4. Как только blob создан посредством операции PUT block list, все загруженные, но не вошедшие в список блоков операции PUT block list блоки будут удалены путем сборки мусора.

Загрузка большого blob может занимать довольно длительное время. При этом загруженные, но не использованные блоки занимают место в хранилище. Многие загруженные блоки могут никогда не войти в список PUT block list. В случае отсутствия активности для данного blob в течение длительного периода времени (в настоящее время этот период составляет неделю), эти неиспользованные блоки будут удалены системой в процессе сборки мусора.

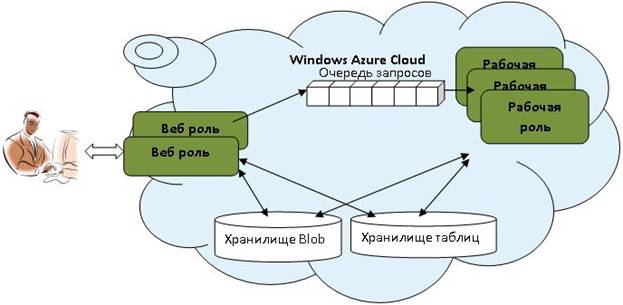
Интересным является сценарий параллельной загрузки блоков для одного blob. В этом случае заслуживают внимания два вопроса: ID блоков и приоритет.

Если для загрузки блоков в один blob приложение использует множество клиентских модулей записи, то во избежание конфликтов ID блоков должны быть уникальными среди всех этих модулей записи, или они должны представлять содержимое записываемого блока. Таким образом, если один и тот же блок записывается несколькими клиентами, ID блока во всех клиентах будет одинаковым, поскольку он представляет одни и те же данные. Во избежание ошибок при потенциальной возможности записи одного Blob несколькими модулями записи в качестве ID блока рекомендуется использовать хеш (например, MD5-хеш) содержимого блока. Таким образом, ID блока будет представлять его содержимое.

Приоритет имеет первая фиксация – в ситуации, когда множество клиентов выполняют загрузку блоков для одного blob параллельно, приоритет имеет первая фиксация blob посредством операции PUT block list (или модуль записи, вызвавший PUT blob первым). Все остальные неиспользованные блоки, загруженные другими модулями записи для blob с этим именем, будут удалены в процессе сборки мусора. Следовательно, для эффективного параллельного обновления blob необходима координация всех клиентов, ведущих запись параллельно.

### 3.3.5 Windows Azure Queue

Windows Azure Queue позволяет разделить разные части приложения в облаке, что делает возможным использование разных технологий для создания этих приложений, и их масштабирование соответственно нуждам трафика.



**Рис. 3.5.** Построение приложений для облака с использованием Azure Queue

Рисунок 3.5 иллюстрирует простой и распространенный сценарий для приложений в облаке. Имеется ряд веб ролей, на которых размещается интерфейсная логика обработки веб-запросов. Также существует ряд рабочих ролей, реализующих бизнес-логику приложения. Веб роли обмениваются информацией с рабочими ролями посредством наборов запросов. Устойчивое состояние приложения может сохраняться в хранилищах Windows Azure Blob и Windows Azure Table.

Приложение он-лайн сервиса видеохостинга. Пользователи могут загружать видео в это приложение; после этого приложение может автоматически преобразовывать и сохранять этот видео файл в различных форматах мультимедиа; кроме того, приложение автоматически индексирует данные описания видео для упрощения поиска (например, по ключевым словам описания, именам актеров, режиссеров, названию и т.д.).

Такое приложение может использовать описанную ранее архитектуру. Веб-роли реализуют уровень представления и обрабатывают веб-запросы пользователей. Пользователи могут загружать видео через веб-интерфейсы. Видео-файлы могут храниться как большие двоичные объекты в хранилище Azure Blob. Приложение может также обслуживать ряд таблиц для учета имеющихся видео файлов и хранения индексов, используемых для поиска. Рабочие роли выполняют преобразование входящих видео файлов в разные форматы и сохранение их в хранилище Azure Blob. Рабочие роли также отвечают за обновление таблиц этого приложения в Azure Table. При получении запроса пользователя (например, запроса на загрузку видео) веб-роли создают рабочий элемент и помещают его в очередь запросов. Рабочие роли извлекают эти рабочие элементы из очереди и обрабатывают соответствующим образом. В случае успешной обработки рабочая роль должна удалить рабочий элемент из очереди во избежание повторной его обработки другой рабочей ролью.

Благодаря применению Windows Azure Queue такая архитектура обладает рядом преимуществ: масштабируемость, разделение ролей, всплески трафика.

Приложение может легче масштабироваться соответственно нуждам трафика.

Во-первых, длина очереди напрямую отражает насколько хорошо рабочие роли справляются с общей рабочей нагрузкой. Увеличение очереди свидетельствует о том, что рабочие роли не могут обрабатывать данные с необходимой скоростью. В этом случае приложению может потребоваться увеличить количество рабочих ролей, чтобы обеспечить более быструю обработку. Если длина очереди неизменно остается близкой к нулю, это означает, что серверная часть приложения обладает большими вычислительными мощностями, чем требуется. В этом случае приложение может сократить количество рабочих ролей для обеспечения рационального расходования ресурсов. Отслеживая размеры очереди и настраивая количество внутренних узлов, приложения могут эффективно масштабироваться соответственно объемам трафика.

Во-вторых, применение очередей позволяет разделить части приложения и выполнять их масштабирование независимо друг от друга, что намного упрощает эту задачу. В данном примере веб-роли и рабочие роли отделены друг от друга и обмениваются информацией через очереди. Это позволяет настраивать количество веб-ролей и рабочих ролей независимо друг от друга без влияния на логику приложения. Таким образом, приложение может свободно масштабировать критически важные компоненты, добавляя для них больше ресурсов или компьютеров.

В-третьих, применение очередей обеспечивает гибкость для эффективного использования ресурсов в приложении, что повышает эффективность масштабирования. То есть для рабочих элементов с разными приоритетами и/или разных размеров могут использоваться разные очереди, которые будут обрабатываться разными пулами рабочих ролей. В этом случае  
приложение может распределять соответствующие ресурсы (например, с точки зрения количества серверов) для каждого пула, таким образом, обеспечивая эффективное использование доступных ресурсов при разных характеристиках трафика. Например, рабочие элементы, имеющие критически важное значение для всей задачи, могут быть помещены в отдельную очередь. Это обеспечит их обработку независимо от всех остальных задач. Кроме того, отдельная очередь может использоваться для рабочих элементов, обработка которых требует привлечения большого количества ресурсов (например, преобразование видео). Для каждой из этих очередей могут использоваться разные пулы рабочих ролей. Приложение может настраивать размер каждого из пулов независимо, соответственно поступающему трафику.

Использование очередей позволяет разделить разные части приложения, что обеспечивает существенную гибкость и расширяемость с точки зрения построения приложения. Сообщения в очереди могут быть в стандартном или расширяемом формате, таком как XML, что обеспечивает независимость компонентов на обоих концах очереди друг от друга, поскольку они могут понимать сообщения в очереди. Разные части системы могут быть реализованы с использованием разных технологий и языков программирования. Например, компонент на стороне очереди может быть написан на .NET  
Framework, а другой компонент – на Python. Более того, изменения, происходящие внутри компонента, не имеют никакого влияния на   
остальную систему. Например, компонент может быть изменен с применением совершенно другой технологии или языка программирования, при этом система будет продолжать работать точно так же, и для этого не придется изменять другие компоненты, поскольку использование очередей обеспечивает разделение компонентов. Использование очередей делает возможным сосуществование в системе разных реализаций одного и того же компонента, т.е. приложение может свободно переходить к новым технологиям. В рассматриваемом выше примере можно постепенно уходить от компонентов, созданных с применением устаревших технологий, и заменять их новыми реализациями. Старая и новая реализации могут выполняться одновременно на разных серверах и обрабатывать рабочие элементы одной очереди. При этом другие компоненты приложения никак не будут затронуты.

Очереди обеспечивают буферизацию, что компенсирует всплески трафика и сокращает влияние, оказываемое сбоями отдельных компонентов. В рассматриваемом ранее примере возможны случаи поступления большого числа запросов в короткий промежуток времени. Рабочие роли не могут быстро обработать все запросы. В этом случае запросы не отклоняются, а буферизуются в очередь, и рабочие роли получают возможность обрабатывать их в собственном нормальном темпе, постепенно возвращаясь к обычному режиму работы. Это позволяет приложению обрабатывать неравномерный трафик без снижения надежности. Использование очередей также снижает влияние, оказываемое сбоями отдельных компонентов. В рассматриваемом выше примере в случае сбоя нескольких рабочих ролей очередь обеспечит буферизацию всех рабочих элементов, поступивших во время простоя рабочих ролей, таким образом, эти элементы не будут утрачены. Когда рабочие роли вернуться в работу, они смогут продолжить обработку рабочих элементов из очереди и, в конце концов, вернуться к нормальному режиму обработки данных по мере их поступления. Такие сбои не оказывают никакого влияния на другие компоненты. Обратите внимание, что рабочие элементы, обрабатываемые рабочими ролями на момент их сбоя, также не будут утеряны, поскольку возвратятся в очередь по истечении времени ожидания видимости (англ. Visibility Timeout), что обеспечивает сохранность данных при сбоях компонентов. Таким образом, приложение может переживать сбои без потери надежности.

Итак, благодаря модели очереди приложения застрахованы от потери данных и снижения надежности даже в условиях систематических сбоев компонентов приложения. Для обеспечения корректной работы этой модели разработчик приложения должен обеспечить идемпотентность обработки рабочих элементов очереди рабочими ролями. Благодаря этому, прежде чем рабочий элемент будет полностью обработан и удален из очереди, допускаются его многократные частичные обработки, прерывающиеся в результате сбоев.

### 3.3.6 Очередь

Очередь содержит множество сообщений. Область действия имени очереди ограничена учетной записью. Количество сообщений в очереди не ограничено. Сообщение хранится максимум неделю. Система удаляет сообщения, поступившие более недели назад, в процессе сборки мусора. С очередями могут быть ассоциированы метаданные. Метаданные представляются в форме пар <имя, значение>, их размер может составлять максимум 8KБ на очередь.

### 3.3.7 Сообщения.

Сообщения хранятся в очередях. Каждое сообщение может быть размером не более 8 КБ. Для хранения данных большего размера используются хранилища Azure Blob или Azure Table, а в сообщении указывается имя большого двоичного объекта или сущности. Стоит обратить внимание, что когда сообщение помещается в хранилище, его данные могут быть двоичными. Но при извлечении сообщений из хранилища ответ формируется в формате XML, и данные сообщения возвращаютсяbase64-кодированными. Сообщения могут возвращаться из очереди в любом порядке, и сообщение может быть возвращено несколько раз. Рассмотрим некоторые параметры, используемые Azure Queue Service.

*Message ID* – это значение GUID, которое идентифицирует сообщение в очереди.

*Visibility Timeout –* целое значение, определяющее время ожидания видимости сообщения в секундах. Максимальное значение – 2 часа. Значение по умолчанию – 30 секунд.

*Pop Receipt* – строка, возвращаемая для каждого извлеченного сообщения. Эта строка, вместе с Message ID, необходима для удаления сообщения из очереди (англ. Queue). Данный параметр следует рассматривать как непрозрачный, поскольку в будущем его формат и содержимое можно изменить.

*Message TTL* – определяет срок жизни (англ. time-to-live, TTL) сообщения в секундах. Максимально допустимый срок жизни – 7дней. Если этот параметр опущен, срок жизни по умолчанию – 7дней. Если в течение срока жизни сообщение не будет удалено из очереди, оно будет удалено системой хранения в процессе сборки мусора.

URI конкретной очереди структурировано следующим образом:

*http://<учетнаязапись>.****queue****.core.windows.net/<ИмяОчереди>*

Первая часть имени хоста образована именем учетной записи хранилища, за которым следует ключевое слово «queue». Это обеспечивает направление запроса в часть Windows Azure Storage, которая обрабатывает запросы очереди. За именем хоста идет имя очереди. Существуют ограничения именования учетных записей и очередей. Например, имя очереди не может включать символ«/».

### 3.3.8 Интерфейс REST очереди

Любой доступ к Windows Azure Queue выполняется через HTTP-интерфейс REST. Поддерживаются как HTTP, так и HTTPS протоколы.

К командам HTTP или REST на уровне учетной записи относятся: List Queues – представить список всех очередей данной учетной записи.

К командам HTTP или REST на уровне очереди относятся:

* create Queue – создать очередь под данной учетной записью;
* delete Queue – удалить указанную очередь и ее содержимое без возможности восстановления;
* set Queue Metadata – задать или обновить определяемые пользователем метаданные очереди. Метаданные ассоциированы с очередью в виде пар имя или значение. Эта команда обеспечит перезапись всех существующих метаданных новыми;
* get Queue Metadata – извлечь определяемые пользователем метаданные очереди, а также примерное число сообщений в заданной очереди.

Операции уровня очереди могут выполняться с использованием следующего URL:

*http://<учетнаязапись>.****queue****.core.windows.net/<ИмяОчереди>*

К командам HTTP или REST, поддерживаемым для реализации операций на уровне сообщения, относятся:

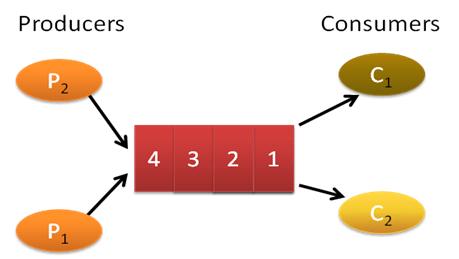
* put Message (Queue Name, Message, Message TTL) – добавить новое сообщение в конец очереди. Message TTL определяет строк жизни данного сообщения. Сообщение может храниться в текстовом или двоичном формате, но возвращается base 64-кодированным;
* get Messages (Queue Name, Num Of Messages N, Visibility Timeout T) – извлечь N сообщений из начала очереди и сделать их невидимыми в течение заданного Visibility Timeout промежутка времени T. Эта операция возвратит ID сообщения, возвращенного вместе с Pop Receipt. Сообщения могут возвращаться из очереди в любом порядке, и сообщение может быть возвращено несколько раз;
* message (Queue Name, Message ID, PopReceipt) – удалить сообщение, ассоциированное с данным Pop Receipt, возвращенным ранее вызовом Get Message. Обратите внимание, что если сообщение не будет удалено, оно повторно появится в очереди по истечении Visibility Timeout;
* peek Message (Queue Name, Num Of Messages N) – извлечь N сообщений из начала очереди, не делая сообщения невидимыми. Эта операция возвратит ID сообщения для каждого возвращенного сообщения;
* clear Queue –удалить все сообщения из заданной очереди. Заметьте, что вызывающая сторона должна повторять эту операцию до тех пор, пока получает сообщения об успешном ее выполнении, это обеспечит удаление всех сообщений очереди.

Операции уровня сообщения могут быть выполнены с использованием следующего URL:

*http://<учетнаязапись>.****queue****.core.windows.net/<ИмяОчереди>/messages*

### 3.3.9 Сценарий поставщик-потребитель

На рисунке 3.6 представлен пример, иллюстрирующий семантику Windows Azure Queue.



**Рис. 3.6.** Пример использования очереди

В примере на рисунке 3.6 поставщики (P1и P2) и потребители (C1 и C2) обмениваются информацией через представленную выше очередь. Поставщики формируют рабочие элементы и помещают их в виде сообщений в очередь. Потребители изымают сообщения/рабочие элементы из очереди и обрабатывают их. Может существовать множество поставщиков и множество потребителей. Рассмотрим последовательность действий.

C извлекает сообщение из очереди. Эта операция возвращает сообщение 1 и делает его невидимым в очереди на 30секунд (принимаем в данном примере, что используется Visibility Timeout по умолчанию, что составляет 30секунд).

Затем появляется C и извлекает из очереди другое сообщение. Поскольку сообщение 1по-прежнему невидимое, эта операция не увидит сообщения 1 и возвратит C сообщение 2.

По завершении обработки сообщения 2 C вызывает Delete, чтобы удалить сообщение 2 из очереди.

Теперь представим, что C дает сбой в ходе обработки сообщения 1, таким образом, C не удаляет это сообщение из очереди.

По истечении времени Visibility Timeout для сообщения 1, оно опять появляется в очереди.

После того, как сообщение 1повторно появилось в очереди, оно может быть извлечено последующим вызовом от C .Тогда C полностью обработает сообщение 1 и удалит его из очереди.

Как показано в этом примере, семантика API очереди гарантирует каждому сообщению в очереди шанс быть обработанным полностью, по крайней мере, один раз. То есть, если возникает сбой потребителя в период после извлечения им сообщения из очереди и до его удаления, сообщение опять появится в очереди по истечении времени Visibility Timeout. Это обеспечивает возможность этому сообщению быть обработанным полностью другим потребителем.

## 3.4 Table Storage

Ниже представлен обзор модели данных таблицы Windows Azure Table.

**Таблица (англ. Table)** – содержит набор сущностей. Область действия имен таблиц ограничена учетной записью. Приложение может создавать множество таблиц в рамках учетной записи хранилища.

**Сущность (строка) (англ. Entity (Row)).** Сущности (сущность является аналогом «строки») – это основные элементы данных, хранящиеся в таблице. Сущность включает набор свойств. В каждой таблице имеется два свойства, «Partition Key» и «Row Key», которые образуют уникальный ключ для сущности.

**Свойство (столбец) (англ. Property (Column))** – представляет отдельное значение сущности. Имена свойств чувствительны к регистру. Для значений свойств поддерживается богатый набор типов.

**Ключ секции (англ. Partition Key)** – первое свойство ключа каждой таблицы. Эта система использует данный ключ для автоматического распределения сущностей таблицы по множеству узлов хранения.

**Ключ строки (англ. Row Key)** – второе свойство ключа таблицы. Это уникальный ID сущности в рамках секции. Partition Key в сочетании с Row Key уникально идентифицирует сущность в таблице.

**Временная метка (англ. Time stamp)** – каждая сущность имеет версию, сохраняемую системой.

**Секция (англ. Partition)** – набор сущностей в таблице с одинаковым значением ключа секции.

**Порядок сортировки (англ. Sort Order)** – для CTP-версии предоставляется всего один индекс, в котором все сущности сортированы по Partition Key и затем по Row Key. Это означает, что запросы с указанием этих ключей будут более эффективными, и все возвращаемые результаты будут сортированы по Partition Key и затем по Row Key.

Таблица имеет гибкую схему. Windows Azure Table отслеживает имя и типизированное значение каждого свойства каждой сущности. Приложение может моделировать фиксированную схему на стороне клиента, обеспечивая одинаковый набор свойств для всех создаваемых сущностей.

Имена таблиц могут включать только буквенно-цифровые символы. Имя таблицы не может начинаться с цифрового символа. Имена таблиц чувствительны к регистру. Имена таблиц должны включать от 3до 63 символов.

Имя свойства – допускаются только буквенно-цифровые символы и «\_».

Сущность может иметь до 255 свойств, включая обязательные системные свойства: Partition Key, Row Key и Time stamp. Имена всех остальных свойств сущностей определяются приложением.

Свойства Partition Key и Row Key строкового типа размером не более 1 кБ.

Свойство Time stamp является доступным только для чтения обслуживаемым системой свойством, которое должно рассматриваться как непрозрачное свойство.

Отсутствие фиксированной схемы – Windows Azure Table не сохраняет никакой схемы, поэтому все свойства хранятся как пары <имя, типизированное значение>. Это означает, что свойства сущностей одной таблицы могут сильно отличаться. В таблице даже может быть две сущности, свойства которых имеют одинаковые имена, но разные типы значений. Однако имена свойств в рамках одной сущности должны быть уникальными.

Суммарный объем всех данных сущности не может превышать 1МБ. Сюда входит размер имен свойств, а также размер значений свойств или их типов, включая и два обязательных свойства ключей (Partition Key и Row Key).

Используются стандартные ограничения для Http.sys, согласно которым размер сегмента URI не может превышать 260 символов. Это приводит к ограничению размера секции и ключа строки, поскольку Get Row, Delete, Update, Merge требуют включения секции и ключа строки как части одного сегмента URI. Например, следующий URI определяет одну сущность с Partition Key «pk» и RowKey «rk»:

*http://myaccount.windows.core.net/Customers(PartitionKey="pk",RowKey="rk")*

Из-за налагаемых Http.sys ограничений выделенная часть не может включать более 260 символов. Для решения этой проблемы операции, имеющие такое ограничение, могут выполняться с помощью Entity Group Transactions (Транзакции групп сущностей), поскольку в Entity Group Transactions URI, определяющий ресурс, является частью тела запрос (см. табл. 3).

**Таблица 3.** Перечень типов свойств

|  |  |
| --- | --- |
| **Тип свойства** | **Описание** |
| Binary | Массив байтов размером до 64 КБ. |
| Bool | Булево значение. |
| Date Time | 64-разрядное значение, представляющее время в формате UTC. Поддерживаемый диапазон значений: от 1/1/1601 до 12/31/9999. |
| Double | 64-разрядное значение с плавающей точкой. |
| GUID | 128-разрядный глобально уникальный идентификатор. |
| Int | 32-разрядное целое значение. |
| Int64 | 64-разрядное целое значение. |
| String | 16-разрядное UTF-кодированное значение. Размер строковых значений может быть до 64 КБ. |

### 3.4.1 Секционирование таблиц

Windows Azure Table обеспечивает возможность масштабирования таблиц до тысяч узлов хранения через распределение сущностей в таблице. При распределении сущностей желательно обеспечить, чтобы сущности, входящие в одно множество, располагались в одном узле хранения. Приложение формирует эти множества соответственно значениям свойства Partition Key сущностей.

Приложениям должна быть известна рабочая нагрузка каждой отдельно взятой секции. Для обеспечения желаемых результатов тестирование должно моделировать максимальную рабочую нагрузку.

Таблица 4 содержит множество версий документов. Каждая сущность данной таблицы соответствует определенной версии определенного документа. В этом примере ключом секции таблицы является имя документа, и ключом строки – номер версии. Имя документа и версия уникально идентифицируют каждую сущность таблицы. В данном примере секцию образуют все версии одного документа.

**Таблица 4. Примеры секций**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Ключ секции | Ключ строки | Свойство 3 | Свойство N |  |
| Имя документа | Версия | Время изменения | Описание |  |
| Examples Doc | V1.0 | 8/2/2007 | Завершенная версия | Секция 1 |
| Examples Doc | V2.0.1 | 9/28/2007 | Рабочая версия Алисы |
| FAQ Doc | V1.0 | 5/2/2007 | Завершенная версия | Секция 2 |
| FAQ Doc | V1.0.1 | 7/6/2007 | Рабочая версия Алисы |
| FAQ Doc | V1.0.2 | 8/1/2007 | Рабочая версия Салли |

Стоит уделить внимание назначениям секций таблицы и принципах выбора ключа секции.

### 3.4.2 Масштабируемость таблицы.

Хорошая масштабируемость системы хранения достигается за счет распределения секций по множеству узлов хранения.

Система отслеживает характер использования секций и автоматически равномерно распределяет эти секции по всем узлам хранения. Это позволяет системе и приложению масштабироваться соответственно количеству запросов к таблице. То есть если некоторые секции запрашиваются больше других, система автоматически разнесет их на несколько узлов хранения, таким образом, распределяя трафик между множеством серверов. Однако секция, т.е. все сущности, имеющие одинаковый ключ секции, будут обслуживаться как один узел. Но даже несмотря на это объем данных в рамках секции не ограничен емкостью хранилища одного узла хранения.

### 3.4.3 Транзакции над группами сущностей

Приложение может автоматически выполнять транзакцию над сущностями, хранящимися в одной таблице и одной секции(т.е. имеющих одинаковое значение ключа секции).Таким образом, приложение может автоматически выполнять множество операций или Create, или Update, или Delete для множества сущностей за один пакетный запрос к системе хранения, поскольку все сущности располагаются в одной таблице и имеют одинаковое значение ключа секции. При выполнении транзакции обеспечивается изоляция моментального снимка независимо от того, были операции над сущностями выполнены успешно либо все они дали сбой. Все остальные запросы, выполняющиеся параллельно в это же время, не будут видеть результат транзакции, поскольку работают с моментальным снимком, сделанным до ее завершения. Результат транзакции становится видимым запросам только после полного успешного ее завершения.

Для Entity Group Transaction указание заголовка версии является обязательным, должна быть указана версия "2009-04-14" или более поздняя.

### 3.4.4 Расположение сущностей

Сущности одной секции хранятся вместе. Это обеспечивает наиболее эффективную обработку запросов к секции. Более того, в этом случае приложение может использовать все преимущества эффективного хэширования и других оптимизаций производительности, обеспечиваемых расположением данных в секции.

В примере выше секцию образуют все версии одного документа. Таким образом, для извлечения «всех версий данного документа» необходимо выполнить доступ всего к одной секции. С другой стороны, чтобы получить все версии документов, измененные до 5/30/2007, придется запрашивать несколько секций. Поскольку запросу придется проверять все секции, которые к томе же могут располагаться на разных узлах хранения, такой запрос будет менее эффективным и более ресурсоемким.

### 3.4.5 Программирование таблиц

Для таблиц и сущностей поддерживаются следующие базовые операции:

* создание таблицы или сущности;
* таблицы или сущности с применением фильтров;
* обновление сущности (но не таблицы);
* таблицы или сущности;
* над группами сущностей, которые поддерживают транзакции с множеством сущностей одной таблицы и секции.

В таблице 5 перечислены основные операции над таблицами и сущностями. Для работы с таблицами в .NET-приложении можно просто использовать ADO.NET Data Services.

В следующей таблице приведен список предлагаемых API. Поскольку применение ADO.NET Data Services в итоге сводится к передаче REST-пакетов, приложения могут использовать REST напрямую. Кроме того, что REST обеспечивает возможность доступа к хранилищу посредством не .NET языков, он также позволяет реализовывать более тонкое управление сериализацией или десериализацией сущностей, что пригодится при работе с такими сценариями, как наличие разных типов сущностей, или если необходимо, чтобы объект имел большее количество свойств, чем допускается для данной сущности.

**Таблица 5.** Основные операции над таблицами и сущностями

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Операция** | **ADO.NET Data Services** | **HTTP-команда** | **Ресурс** | **Описание** |
| Запрос | LINQ Query | GET | Таблица | Возвращает список всех таблиц данной учетной записи хранилища. В случае наличия фильтра таблицы возвращаются соответственно фильтру. |
| Сущность | Возвращает все сущности заданной таблицы или подмножество сущностей, если задан фильтр. |
| Обновление всей сущности | Update Object & Save Changes (Save Changes Options. Replace On Update) | PUT | Сущность | Обновляет значения свойств сущности. Операция PUT замещает всю сущность и может использоваться для удаления свойств. |
| Частичное обновление сущности | Update Object & Save Changes() | MERGE | Сущность | Обновляет значения свойств сущности. |
| Создание новой сущности | Add Object & Save Changes() | POST | Таблица | Создает новую таблицу в это учетной записи хранилища. |
| Сущность | Вставляет новую сущность в названную таблицу. |
| Удаление сущности | Delete Object & Save Changes() | DELETE | Таблица | Удаляет таблицу в данной учетной записи хранилища. |
| Сущность | Удаляет сущность из названной таблицы. |
| Транзакция над группой сущностей | Save Changes (Save Changes Options.Batch) | POST | $batch | Поддержка транзакции над группой сущностей обеспечивается посредством пакетной операции над сущностями одной таблицы с одинаковым ключом секции. В ADO.NET DataServices опция Save Changes требует, чтобы запрос выполнялся как одна транзакция. |

К расширенным операциям с таблицами относятся:

* разбиение на страницы;
* конфликтов, возникающих в результате параллельных обновлений.

Приведём пример. В таблице «Blogs» хранятся блоги для приложения MicroBlogging.

В приложении Micro Blogging есть две таблицы: Channels (Каналы) и Blogs (Блоги). Имеется список каналов, блоги публикуются в определенном канале. Пользователи подписываются на каналы и ежедневно получают новые блоги этих каналов.

Над таблицей Blogs можно производить следующие операции: описание схемы таблицы; создание таблицы; вставка блога в таблицу; получение списка блогов из таблицы; обновление блога в таблице; удаление блога из таблицы; вставка множества блогов в таблицу

### 3.4.6 Описание класса сущности для таблицы

Схема таблицы описывается как C#-класс. Такую модель использует ADO.NET Data Services. Схема известна только клиентскому приложению и упрощает доступ к данным. Сервер схему не применяет.

Рассмотрим описание сущностей Blog, хранящихся в таблице Blogs. Каждая сущность блога содержит данные: имя канала (Channel Name) – канал, в котором размещается блог, – дата размещения, текст (Text) – содержимое тела блога, – рейтинг (Rating) – популярность этого блога.

Для данной таблицы «Blogs» мы выбрали в качестве Partition Key имя канала и в качестве Row Key – дату размещения блога. Partition Key и RowKey – ключи таблицы Blogs, они объявляются посредством атрибута класса Data Service Key (Ключ сервиса данных). Таблица «Blogs» секционирована по именам каналов (англ. Channel Name). Это позволяет приложению эффективно извлекать самые недавние блоги канала, на который подписан пользователь. Кроме ключей, в качестве свойств объявлены характерные для пользователя атрибуты. Все свойства имеют открытые (англ. public) методы считывания и присвоения значения и хранятся в таблице Windows Azure Table.

В примере ниже: Text и Rating хранятся для экземпляра сущности в таблице Azure. Rating As String нет, потому что для него не определен метод присвоения значения. Id не хранится,потому что методы доступа не public.

C#:

[*Копировать*](javascript:CodeSnippet_CopyCode('CodeSnippetContainerCode_6e3f6d0d-bbca-4a4c-9287-ee592366c383');)

*[Data Service Key("Partition Key", "Row Key")]*

*publicclass Blog*

*{*

*// Channel Name*

*publicstring Partition Key { get; set; }*

*// Posted Date*

*publicstring Row Key { get; set; }*

*// Определяемые пользователем свойства*

*publicstring Text { get; set; }*

*publicint Rating { get; set; }*

*publicstring RatingAsString { get; }*

*protectedstring Id { get; set; }*

*}*

### 3.4.7 Создание таблицы

Создание таблицы Blogs для учетной записи хранилища аналогично созданию сущности в основной таблице «Tables». Эта основная таблица определена для каждой учетной записи хранилища, и имя каждой таблицы, используемой учетной записью хранения, должно быть зарегистрировано в основной таблице. Описание класса основной таблицы приведено ниже, где свойство Table Name (Имя таблицы) представляет имя создаваемой таблицы.

C#:

[*Копировать*](javascript:CodeSnippet_CopyCode('CodeSnippetContainerCode_0b8e1557-8adc-4f30-86d4-26334d6544d7');)

*[Data Service Key("TableName")]*

*publicclass Table Storage Table*

*{*

*publicstring Table Name { get; set; }*

*}*

*Фактическое создание таблицы происходит следующим образом:*

*// Uri сервиса: “http://<Account>.table.core.windows.net/”*

*Data Service Context context = new Data Service Context(service Uri);*

*Table Storage Table table = new Table Storage Table("Blogs");*

*// Создаем новую таблицу, добавляя новую сущность*

*// в основную таблицу "Tables"*

*context.Add Object("Tables", table);*

*// результатом вызова Save Changes является отклик сервера*

*Data Service Responseresponse = context.Save Changes();*

serviceUri – это uri сервиса таблицы, http ://<Здесь указывается имя учетной записи>. table . core . windows . net /. DataServiceContext (Контекст сервиса данных) – один из основных классов сервиса данных ADO.NET, представляющий контекст времени выполнения для сервиса. Он обеспечивает API для вставки, обновления, удаления и запроса сущностей с помощью либо LINQ, либо REST ful URI и сохраняет состояние на стороне клиента.

### 3.4.8 Использование REST API

Результатом всех рассматриваемых выше операций является передача HTTP-сообщений на сервера. Приложение может отказаться от использования клиентской библиотеки .NET и работать на уровне HTTP или REST.

### 3.4.9 Параллельные обновления

Для обновления сущности необходимо выполнить следующие операции:

* получить сущность с сервера;
* обновить объект локально и вернуть его на сервер.

Предположим, два процесса, выполняющихся параллельно, пытаются обновить одну и ту же сущность. Поскольку шаги 1 и 2 не являются неделимыми, на любом из них может возникнуть ситуация внесения изменений в уже устаревшую версию сущности. Для решения этой проблемы Windows Azure Table использует нежесткую блокировку.

Для каждой сущности система сохраняет версию, которая изменяется сервером при каждом обновлении.

При извлечении сущности, сервер отправляет эту версию клиенту в виде ETag HTTP.

Когда клиент передает запрос UPDATE (обновить) на сервер, он отправляет на него этот ETag в виде заголовка If-Match.

Если версия сущности, хранящаяся на сервере, аналогична ETag в заголовке If-Match, изменение принимается, и хранящаяся на сервере сущность получает новую версию. Эта новая версия возвращается клиенту как заголовок ETag.

Если версия сущности на сервере отличается от ETag в заголовке If-Match, изменение отклоняется, и клиенту возвращается HTTP-ошибка «precondition failed» (необходимое условие не выполнено).

При получении ошибки «precondition failed» типовым поведением клиентского приложения будет повторение всей операции.

Приложение должно извлечь этот объект снова, т.е. получить его последнюю версию и ещё обновить объект локально и вернуть его на сервер.

При использовании клиентской библиотеки .NET приложение получает HTTP-код ошибки в виде исключения Data Service Request Exception.

В примере ниже два разных клиента выполняют один и тот же код для изменения текста. Эти два клиента пытаются задать свойству «Text» разные значения. Рассмотрим возможную последовательность событий, иллюстрирующую обработку параллельных обновлений.

1. Оба клиента извлекают сущность. При этом для каждой сущности извлекается ETag, например, «v1». Оба клиента полагают, что предыдущая версия сущности – «v1».

2. Каждый клиент локально обновляет свойство Text.

3. Каждый клиент вызывает методы Update Object и Save Changes.

4. Каждый клиент отправляет на сервер HTTP-запрос с заголовком «If-Match: v1».

5. Запрос одного из клиентов попадает на сервер первым.

5.1. Сервер сравнивает заголовок «If-Match» с версией сущности. Они совпадают.

5.2. Сервер применяет изменение.

5.3. Версия сущности на сервере обновляется и становится«v2».

5.4. В качестве ответа клиенту отправляется новый заголовок «ETag:v2».

6. Далее на сервер поступает запрос другого клиента. На этот момент изменения первого клиента уже применены.

6.1. Сервер сравнивает заголовок «If-Match»с версией сущности. Они не совпадают, поскольку версия сущности уже изменена на «v2»,тогда как в запросе указывается версия «v1».

6.2. Сервер отклоняет запрос.

C#:

[*Копировать*](javascript:CodeSnippet_CopyCode('CodeSnippetContainerCode_aff5dee7-4344-4b10-893a-4e5945628493');)

*// Задаем такой вариант объединения, который обеспечивает*

*// сохранение обновлений, но позволяет обновление etag.*

*// По умолчанию применяется значение Append Only, при котором*

*// уже отслеживаемая сущность не перезаписывается значениями,*

*// полученными с сервера, в результате чего в случае изменения*

*// сущности на сервере используется недействительный etag.*

*context.Merge Option = Merge Option.Preserve Changes;*

*Blog blog =*

*(from blog in context.Create Query<Blog>("Blogs")*

*where blog.Partition Key == "Channel9"*

*&& blog.RowKey == "Oct-29"*

*select blog).First Or Default();*

*blog.Text = "Hi there again";*

*try*

*{*

*context.Update Object(blog);*

*Data Service Response response = context.Save Changes();*

*}*

*catch (Data Service Request Exception e)*

*{*

*Operation Response response = e.Response.First();*

*if (response.Status Code == (int)Http Status Code. Precondition Failed)*

*{*

*// выполняем запрос объекта повторно, чтобы получить*

*// последний etag, и проводим обновление*

*}*

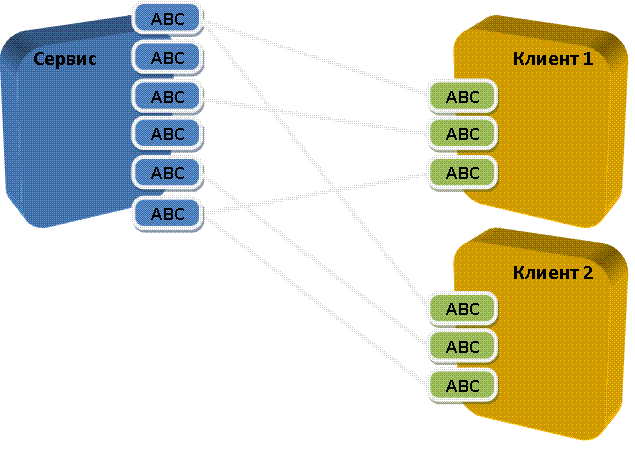
*}*

## 3.5 Технология «Windows Communication Foundation»

В основе Windows Communication Foundation лежит SOA (англ. Service Oriented Architecture) - сервис-ориентированная архитектура. Идея SOA состоит в том, что на сервере работает некоторое количество сервисов, предоставляющих доступ к какой-то информации и позволяющих оперировать ею. Клиенты в свою очередь имеют на своей стороне классы-прокси, которые содержат ссылки на соответствующие операции на стороне сервиса. Таким образом, вызывая локально методы класса-прокси, происходит удаленный вызов метода на сервисе. Физическое взаимодействие между клиентом и сервисом, как правило, происходит путем обмена сообщениями (например, в формате SOAP –  англ. Simple Object Access Protocol — простой протокол доступа к объектам) между ними. Клиент может получить описание сервиса, используя, например, WSDL или MEX, но это повод для отдельной статьи.

В основе функционирования Windows Communication Foundation лежат так называемые конечные точки (англ. endpoint), которые составляет связка "Address - Binding - Contract", "ABC". Каждая составляющая играет важную роль в определении конечной точки. Компонента "Address" содержит месторасположение конечной точки. Адрес может быть как абсолютным, так и относительным (относительно базового адреса). Компонента "Binding" задает привязку и, фактически, определят транспорт, на основе которого будет происходить взаимодействие. В объектной модели Windows Communication Foundation уже определен ряд классов-привязок, например, Basic Http Binding (простая привязка на основе HTTP), Net Tcp Binding (привязка на основе транспорта TCP) и т.д. Ну и, наконец, компонента "Contract" задает контракт, на основе которого будет происходить взаимодействие клиента и сервиса. Фактически, определение контракта регламентирует, какие операции "умеет" выполнять сервис. На основе контракта строится класс-прокси на клиенте.

Как правило, на сервисной стороне задается множество конечных точек. Так, для одного и того же контракта могут быть задано несколько конечных точек (например, для привязок Basic Http и Net Tcp). Кроме того, вы можете разместить несколько сервисов на сервисной стороне. В итоге, сервисная сторона распределенного приложения будет выглядеть как совокупность конечных точек. Клиент в этом случае создает соединение с той конечной точкой, которая нужна именно ему. Пример схемы соединения сервиса и клиентов показан на рисунке\_3.8.



**Рис. 3.8.** Схема соединения сервиса и клиента

# 4 ЭКОНОМИЧЕСКАЯ ЧАСТЬ

## 4.1 Введение

Дипломный проект посвящен разработке и реализации системы связи с использованием облачных технологий в интегрированной среде Visual Studio 2010. На сегодняшний день облачные технологии дают возможность перейти от приобретения, управления и амортизации аппаратных активов к покупке процессорного времени, дискового пространства, сетевой пропускной способности, необходимой для выполнения компьютерного приложения. Модель предполагает дальнейшее развитие в программный комплекс, предназначенный для коммуникации и взаимодействия пользователей системы в режиме реального времени. Таким образом, модель системы связи – это важная и актуальная задача.

Экономическим преимуществом разработки является снижение затрат на средства обеспечения связи между преподавательским персоналом вузов друг с другом, а также со студентами: денежные отчисления на командировки сотрудников, предоставление специализированных помещений. Повышение эффективности взаимодействия достигается за счет реализации системы на платформе разработки облачных приложений Windows Azure с использованием реляционной базы - Windows Azure Table Storage, структурированного хранилища состояний сервиса - Windows Azure Blob Storage. Экономическая выгода продукта также подкреплена использованием бесплатной студенческой версии Windows 7, Expression Encoder.

Проект изначально не преследует цель получения прибыли.

## 4.2 Основные этапы проекта разработки программного продукта

Разработка ПП разбивается на следующие этапы (стадии): техническое задание; эскизный проект; технический проект; рабочий проект; внедрение.

В таблице 4.1 можно посмотреть содержание каждого из этапов.

Таблица 4.1− Основные этапы разработки ПП.

|  |  |  |
| --- | --- | --- |
| № | Этап разработки | Содержание работ |
| 1 | Техническое задание (ТЗ) | Предпроектное исследование сервисов Infrastructure as a Service, Platform as a Service, Software as a Service. Постановка задачи, выбор основных требований. Расчёт технико-экономического обоснования разработки. Выбор технологии программирования - Entity Framework с использованием языка программирования C#. Разработка календарного плана выполнения работ. |
| 2 | Эскизный проект (ЭП) | Разработка структуры программного продукта. Формирование структуры и формы представления входных и выходных данных. Определение основных средств системы: Windows Azure Blob Storage, Windows Azure Table Storage. Разработка общих алгоритмов решения задач. Выработка плана внедрения системы. Создание пояснительной записки в соответствии с ГОСТ. Согласование и утверждение эскизного проекта. |
| 3 | Технический проект (ТП) | Разработка конкретных алгоритмов решения задач, структуры программы, пояснительной записки. Выбор среды программирования, конфигурации технических средств. Создание программной документации в соответствии с ГОСТ. Согласование и утверждение технического проекта. |
| 4 | Рабочий проект (РП) | Реализация модуля на языке программирования в интегрированной среде Visual Studio 2010. Разработка программной документации и методики испытаний. Проведение всех видов испытаний. Отладка программы с помощью Visual Studio Team System. Сдача проекта в опытную эксплуатацию. |
| 5 | Документация и внедрение (В) | Подготовка и передача программы и программной документации для оформления и утверждения акта о передаче. Передача программного продукта заказчику. Настройка и внедрение. |

## 4.3 Расчет трудоемкости разработки программного продукта

Одним из основных затратных показателей являются совокупные затраты на оплату труда исполнителей. Расчёт трудоёмкости является основополагающим для определения общих (совокупных) затрат на реализацию проекта, поэтому расчёту трудоёмкости уделено особое внимание.

Трудоемкость разработки программной продукции зависит от степени новизны разработки, сложности алгоритма ее функционирования, объема используемой информации и вида ее обработки, уровня используемого алгоритмического языка программирования.

По степени новизны разрабатываемая программная продукция может быть отнесена к одной из четырех групп таблице 6.

**Таблица 6.** – Классификация программных продукций по степени новизны

|  |  |
| --- | --- |
| Название группы | Описание |
| А | Разработка программных комплексов, требующих использования принципиально новых методов их создания, проведение НИРС и т.п. |
| Б | Разработка программной продукции, не имеющей аналогов, в том числе разработка пакетов прикладных программ. |
| В | Разработка программной продукции, имеющей аналоги. |
| Г | Разработка программной продукции, основанной на привязке типовых проектных решений. |

В нашем случае разрабатываемая программа относится к группе «В».

Трудоемкость оценивают на основе известной трудоемкости разработки аналогичного ПО с учетом отличительных особенностей данного проекта, отражаемых введением поправочных коэффициентов. Этот подход предполагает наличие специального фонда алгоритмов и программ–аналогов.

Трудоемкость разработки программной продукции *τПП* может быть определена по формуле 4.1.

*τПП=τТЗ+τЭП+τТП+τРП+τВ ,* (4.1)

где *τТЗ* — трудоёмкость разработки технического задания на создание ПП; *τЭП* — трудоёмкость разработки эскизного проекта; *τТП* — трудоёмкость разработки технического проекта ПП; *τРП* — трудоёмкость разработки рабочего проекта ПП; *τВ* — трудоёмкость внедрения разработанного ПП.

### 4.3.1 Трудоемкость разработки технического задания

Трудоёмкость разработки технического задания рассчитывается по формуле 4.2 [3]:

*τТЗ=TЗРЗ+TЗРП ,*  (4.2)

где *TЗРЗ* — затраты времени разработчика постановки задач (проектировщика) на разработку ТЗ, чел.–дни; *TЗРП* — затраты времени разработчика программного обеспечения на разработку ТЗ, чел.–дни.

Значения величин TЗРЗ и TЗРП рассчитываются по формулам 4.3 и 4.4 соответственно [3]:

*TЗРЗ=tТЗ·KЗРЗ*, (4.3) *TЗРП=tТЗ·KЗРП ,* (4.4)

где *tТЗ* — норма времени на разработку ТЗ на ПП в зависимости от функционального назначения и степени новизны разрабатываемого ПП, чел.–дни; *KЗРЗ* — коэффициент, учитывающий удельный вес трудоёмкости работ, выполняемых разработчиком постановки задач на стадии технического задания (в случае совместной с разработчиком ПП разработки технического задания *KЗРЗ=0,65*); *KЗРП* — коэффициент, учитывающий удельный вес трудоёмкости работ, выполняемых разработчиком ПП на стадии технического задания (в случае совместной с разработчиком постановки задач *KЗРП=0,35*).

По функциональному назначению программа относится к категории «Управление научно-технической информацией», поэтому примем *tТЗ=24*

Таким образом, получим

*τТЗ=TЗРЗ+TЗРП=tТЗ·KЗРЗ+tТЗ·KЗРП=24·0,65+24·0,35=24,00* чел.–дней.

### 4.3.2 Трудоемкость разработки эскизного проекта

Трудоёмкость разработки эскизного проекта ПП *τЭП* рассчитывают по формуле 4.5 [3]:

*τЭП=TЭРЗ+TЭРП,* (4.5)

где *TЭРЗ* — затраты времени разработчика постановки задач на разработку эскизного проекта, чел.–дни; *TЭРП* — затраты времени разработчика ПП на разработку эскизного проекта, чел.–дни.

Значения величин *TЭРЗ* и *TЭРП* рассчитываются по формулам 4.6 и 4.7 соответственно[3]:

*TЭРЗ=tЭП·KЭРЗ,*  (4.6)

*TЭРП=tЭП·KЭРП,* (4.7)

где *tЭП* — норма времени на разработку эскизного проекта на ПП в зависимости от функционального назначения и степени новизны разрабатываемого ПП, чел.–дни; *KЭРЗ* — коэффициент, учитывающий удельный вес трудоёмкости работ, выполняемых разработчиком постановки задач на стадии эскизного проекта; *KЭРП*— коэффициент, учитывающий удельный вес трудоёмкости работ, выполняемых разработчиком ПП на стадии эскизного проекта.

Т.о. для разрабатываемой программы *tТЗ=70* чел. дней, приняв, что при совместной разработке *KЭРЗ=0,35* и *KЭРП=0,65,* получим по формуле 4.5

*τЭП=TЭРЗ+TЭРП=tЭП·KЭРЗ+tЭП·KЭРП=70·0,65+70·0,35=70,00* чел. –дней.

### 4.3.3 Трудоемкость разработки технического проекта

Трудоёмкость разработки технического проекта τТП определяется по формуле 4.8 [3], как сумма времени, затраченного разработчиком постановки задач и разработчиком ПП:

*τТП=(tТРЗ+tТРП)·KВ·KР,* (4.8)

где *tТРЗ, tТРП* — норма времени, затрачиваемого на разработку технического проекта разработчиком постановки задач и разработчиком ПП соответственно, чел.–дни; *KВ* — коэффициент учёта вида используемой информации; *KР* — коэффициент учёта режима обработки информации.

Для ПО управления научно–технической информацией, приняв количество разновидностей форм выходной информации *kвых=2*, а входной информации *kвх=2,* найдём, что *tТРЗ=34* чел.–дней, а *tТРП=13* чел.–дней.

Для стадии технического проекта с режимом обработки информации — реальный масштаб времени(РВ) и группы новизны «В», найдём *KР=1,26.*

Значение коэффициента K*В* определяют из формулы 4.9 [3]:

*KВ=(KП·nП+KНС·nНС+KБ)/(nБnП+nНС+nБ)*, (4.9)

где *KП, KНС, KБ* — значения коэффициентов учёта вида используемой информации для переменной, нормативно–справочной информации и баз данных соответственно; *nП, nНС, nБ* — количество наборов данных переменной, нормативно-справочной информации и баз данных соответственно.

Для группы новизны «В» найдём *KП=1,00.* Коэффициенты *KНС, KБ* не учитываются, т.к. специфика задачи подразумевает работу с переменной информацией. Примем *nП=1, nНС=2, nБ=0.* Подставив значения в формулу 4.9, получаем: *KВ=(1,00·1)/1=1,00.*

Таким образом, по формуле 4.8 вычислим трудоёмкость разработки технического проекта

*τТП=(tТРЗ+tТРП)KВ·KР=(34+13)·1,00·1,26≈59* чел.–дней.

### 4.3.4 Трудоемкость разработки рабочего проекта

Трудоёмкость разработки рабочего проекта τРП определяется по формуле 4.10 [3]:

*τРП=(tРРЗ+tРРП)·KК·KР·KЯ·KЗ·KИА,* (4.10)

где *tРРЗ, tРРП* — нормы времени, затрачиваемые на разработку рабочего проекта на алгоритмическом языке высокого уровня разработчиком постановки задач и разработчиком ПП соответственно, чел.–дней; *KК* — коэффициент учёта сложности контроля информации; *KР* — коэффициент учёта режима обработки информации; *KЯ* — коэффициент учёта уровня алгоритмического языка программирования; *KЗ* — коэффициент учёта степени использования готовых программных модулей; *KИА* — коэффициент учёта вида используемой информации и сложности алгоритма ПП.

Для ПО управления научно-технической информацией, *kвых=2, kвх=2*, найдём, что *tРРЗ=8* чел.–дней, а *tРРП*=13 чел.–дней.

Приняв степень сложности контроля входной информации за 11, а выходной информации за 21, получим *KК=1,16.* Для группа «В», рабочий проект, РВ найдём, что *KР=1,32.* Для алгоритмического языка высокого уровня *KЯ=1,00.* При степени использования готовых программных модулей 20–25%, получим значение коэффициента *KЗ=0,8.* Значение коэффициента KИА определяют по формуле 4.11[3]:

*KИА=(KП'·nП+KНС'·nНС+KБ'·nБ)/(nП+nНС+nБ),* (4.11)

где *KП', KНС', KБ'* — значения коэффициентов учёта сложности алгоритма ПП и вида используемой информации для переменной, нормативно-справочной информации и баз данных соответственно.

С учётом специфики проекта для 2-й группы сложности алгоритма, и группы новизны «В», найдём: *KП'=1,10, KНС'=0, KБ'=0.* Приняв, как и в п.4.3.3, *nП=1, nНС=0, nБ=0, получаем:*

*KИА=(KП'·nП+KНС'·nНС+KБ'·nБ)/(nП+nНС+nБ)=1,10/·1=1,10.*

По формуле 5.10 найдём трудоёмкость разработки рабочего проекта

*τРП=(tРРЗ+tРРП)·KК·KР·KЯ·KЗ·KИА=(8+13)·1,16·1,32·1,00·0,8·1,10≈28*чел.–дней.

### 4.3.5 Трудоемкость выполнения стадии «Внедрение»

Трудоёмкость выполнения внедрения разработанного ПП рассчитывается по формуле 4.12 [3]:

*τВ=(tВРЗ+tВРП)·KК·KР·KЗ,*  (4.12)

где *tВРЗ, tВРП* — норма времени, затрачиваемого разработчиком постановки задач и разработчиком ПП соответственно на выполнение процедур внедрения ПП, чел.–дни.

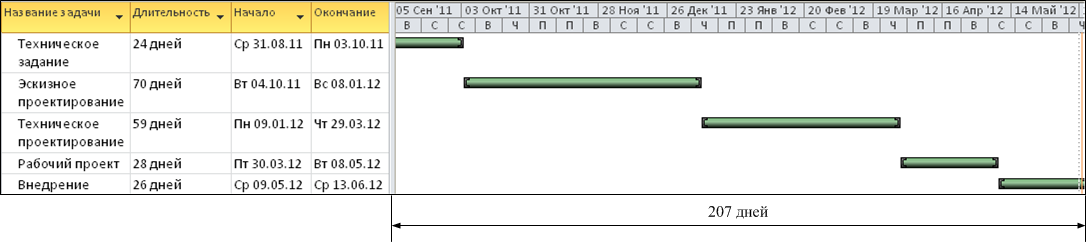
Для ПО управления научно–технической информацией, *kвых=2, kвх=2,* находим *tВРЗ=8* чел.–дней и *tВРП=13* чел.–дней. Следовательно

*τВ=(tВРЗ+tВРП)·KК·KР·KЗ =(8+13)·1,16·1,32·0,80≈26* чел.–дней.

Общая трудоёмкость разработки

*τПП=τТЗ+τЭП+τТП+τРП+τВ=24+70+59+28+26* =207 чел.–дней.

Планирование и контроль хода выполнения разработки проводят по календарному плану выполнения работ. На рисунке 4.1изображена диаграмма Ганта. При построении учитываем одного исполнитель – дипломника и нерабочие дни – выходные и праздники.



**Рис. 4.1****.** Диаграмма Ганта

## 4.4 Расчет затрат на реализацию программного продукта

### 4.4.1 Расчет материальных затрат

Материальные затраты для реализации системы Видеоконференц связи отображены в таблице 7.

**Таблица 7.** Материальные затраты

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование материала | Количество, шт. | Цена за единицу, руб. | Сумма, руб. |
| Компьютер hp 2140 | 1 | 19000 | 19000 |
| Веб-камера hp F380 | 1 | 900 | 900 |
| Итого: |  |  | 19900 |

*Cм=19900* руб.

### 4.4.2 Расчет амортизационных отчислений

В данной статье учитываются суммарные затраты на приобретение оборудования и нематериальных активов, требуемых для разработки данного программного продукта.

Стоимость оборудования распределяется в виде амортизационных отчислений пропорционально времени его использования (см.формулу 4.13 [3]):

*CАМ=КСО·НА·ti/FД,* (4.13)

где *КСО* — балансовая цена оборудования, руб.; *НА*  — норма годовых амортизационных отчислений для оборудования; *FД* – действительный годовой фонд времени, дней; *ti* — время использования оборудования при выполнении данной разработки, дней.

*FД=248* дней. Норму амортизации составляет 12% от первоначальной или восстановительной стоимости основных производственных фондов.

Расходы на амортизационные отчисления

CАМ=19900·0.125·193/248=1986 руб.

### 4.4.3 Расчет заработной платы

В статью «Основная заработная плата» включается основная заработная плата всех исполнителей, непосредственно занятых разработкой данной программной продукции, с учётом их должностного оклада и времени участия в разработке. Расчёт ведётся по формуле 4.14 [3]:

*СЗАРП = СЗ.ОСН.+СЗ.ДОП.+СЗ.ОТЧ. ,* (4.14)

где СЗ.ОСН.—основная заработная плата*,*  СЗ.ДОП.—дополнительная заработная плата*,* СЗ.ОТЧ. —отчисление с заработной платы*.*

Расчет основной заработной платы (оплаты труда непосредственных исполнителей) производится по формуле 4.15 [3]:

*СЗ.ОСН =ТЗАН·ОДН,*(4.15)

где ТЗАН.— число дней, отработанных исполнителем проекта, ОДН—дневной оклад исполнителя.

При 8-и часовом рабочем дне оклад исполнителя рассчитывается по соотношению в формуле 4.16 [3]:

 , (4.16)

где *ОМЕС*- месячный оклад,  *FМ-* месячный фонд рабочего времени.

С учетом налога на доходы физических лиц размер оклада увеличивается, что отражено в формуле 4.17 [3]:

 , (4.17)

где *O* — «чистый» оклад, *ННДФЛ*— налог на доходы физических лиц в размере 13%.

Результаты расчета с перечнем исполнителей и их месячных и дневных окладов, а также их трудозатратах и рассчитанной основной заработной платой каждого исполнителя приведены в таблице 4.4.

В статье «Дополнительная заработная плата» учитываются все выплаты непосредственным исполнителям за время, не проработанное на производстве, и определяются по формуле 4.18 [4]:

CЗ.ДОП=CЗ.ОСН.*·*αД, (4.18)

где αД — коэффициент отчислений на дополнительную зарплату, примем αД=0,2.

Рассчитанные значения заработной платы приведены в таблице 8.

**Таблица 8.** Затраты на основную заработную плату сотрудников

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Исполнитель | Месячный оклад, руб. | Дневная заработная плата, руб. | Продолжительность работы, дн. | Основная заработная плата, руб. | Дополнительная заработная плата, руб. |
| Инженер-программист | 50000 | 2995 | 207 | 619965 | 123993 |

Получаем расходы на заработную плату *CЗП=CЗО+CЗД=619965+123993=743958 руб.*

### 4.4.4 Расчет отчислений в социальные фонды

Отчисления с заработной платы состоят в настоящее время в уплате единого социального налога. Налоговым кодексом РФ определяются ставки налога для отчисления в пенсионный фонд РФ, фонд социального страхования, фонды обязательного медицинского страхования (федеральный и территориальный фонды). На момент расчета экономической части продукта сумму ставок ЕСН соответствует 30%.

Отчисления с заработной платы составят:

*СЗ.ОТЧ.=(СЗ.ОСН+СЗ.ДОП.)·НСОЦ =743958·0,3=223187,4руб*

Таким образом, общие затраты на заработную плату составят:

*СЗАРП =619965+123993+223187,4= 967145,4*

Ставка взносов при общей системе налогообложения разбивается по фондам как показано в таблице 9.

**Таблица 9.** Ставка взносов

|  |  |  |
| --- | --- | --- |
| Наименование фонда |  | Ставка взносов, % |
| Пенсионный фонд | Страховая часть | 16 |
|  | Накопительная часть | 6 |
| Фонд медицинского страхования | ФФОМС | 2,9 |
|  | ТФОМС | 5,1 |
| Всего: |  | 30 |

### 4.4.5 Прочие расходы

Расходы на интернет, виртуальную машину Windows Azure: CПР=520*·*10+6000=5200+6000=11200 руб.

## 4.5 Определение цены программного продукта

Таким образом, затраты на разработку программной продукции (сметная себестоимость): C=958181,6 руб.

На рисунке 4.2 приведена структура затрат на выполнение проекта.

**Рис. 4.2.** Структура затрат

Результаты расчётов затрат на разработку программного продукта сведены в таблице 10.

**Таблица 10.** Затраты на программный продукт

|  |  |
| --- | --- |
| Статья затрат | Сумма, руб. |
| Материальные затраты | 19 900 |
| Амортизация оборудования | 1 986 |
| Заработная плата | 743 958 |
| Отчисления в социальные фонды | 223 187,4 |
| Прочие затраты | 11 200 |
| Итого: | 1 000 231,4 |

Цена программной продукции определяется по формуле 4.19 [3]:

*ЦПП=∆Ц1+Dприб,* (4.19)

где ∆Ц — часть стоимости разработки, приходящаяся на одну копию программы; Dприб — процент прибыли, заложенный в цену(в нашем случае е учитывается, т.к. продукт носит некоммерческий характер).

Частичная стоимость разработки, приходящаяся на каждый комплект ПП, определяется по формуле 4.20 [3] на основании данных о планируемом объеме установок:

*∆Ц=CCNK*, (4.20)

где *CC* — сметная стоимость проекта; *NK* — планируемое число копий ПП.

Количество копий взято с учётом предоставления продукта основному вузу в Москве - МГТУ им.Н.Э. Баумана и двум его филиалам: Калужский филиал и Дмитровский филиал. Для МГТУ им.Н.Э. Баумана в Москве выделено 38 копий, Калужскому и Дмитровскому филиалам 4 и 8 копий соответственно.

ЦПП=∆Ц1+Dприб=CC/NK1+Dприб=1000231,4/50=20 005 руб.

## 4.6 Выводы

В результате расчётов было получено общее время выполнения проекта, которое составило 207 дней.

Для выполнения проекта будет задействован один исполнитель – студент-дипломник.

Подсчитаны затраты на создание системы связи с использованием облачных технологий, которые составили 1 000 231,4 рублей.

Была получена цена одной копии программы 20 005 рубля. Копии рассчитаны на использование внутри вуза.

# 5 ЭКОЛОГИЧЕСКАЯ ЧАСТЬ

## 5.1 Введение

При выполнении производственных задач взаимодействие человека с ПЭВМ влечёт за собой контроль и учёт ряда факторов, влияющих на здоровье и показатели производительности работника.

Рабочее место оператора ПЭВМ представляет собой совокупность физических, химических, биологических, социально-психологических и эстетических факторов внешней среды, воздействующих на человека. В ходе выполнения дипломной работы был осуществлён подбор допустимых значений внешней среды в соответствии с действующими нормами: СанПин 2.2.2/2.4.1340-03 - Гигиенические требования к персональным электронно-вычислительным машинам и организации работы, ГОСТ 12.1.004 – 85 – противопожарная безопасность.

## 5.2 Анализ факторов воздействия среды на оператора ПЭВМ

Быстрое развитие компьютерной техники способствует интенсификации умственного труда, но также приводит к возникновению целого ряда проблем, связанных с воздействием на человека электромагнитных, электростатических полей, создаваемых работающим компьютером и отрицательно воздействующих на организм человека. В связи с подобной реальностью возникла необходимость обозначить основные требования к факторам рабочей среды:

а) комплексное воздействие факторов рабочей среды на человека не должно оказывать отрицательного влияния на здоровье оператора в течение всего рабочего времени;

б) факторы рабочей среды не должны вызывать снижение надёжности, качества и продолжительности работоспособности человека в течение рабочего дня.

Воздействие, которое компьютерная техника способна оказать на человека можно разделить на три основные группы.

Первая группа – физическое воздействие: компьютер (в частности видеотерминалы) является источником электромагнитного поля промышленной частоты, электромагнитного излучения радиодиапазона, электростатического и постоянного магнитного полей, рентгеновского излучения. Так же компьютер и периферийное оборудование: вентиляционные устройства ПЭВМ, агрегаты кондиционирования и вентилирования воздуха,- могут создавать шум, а так же изменять микроклимат и ионизацию воздуха в рабочем помещении.

Вторая группа – нагрузка на опорно-двигательный аппарат человека: интенсивная работа с клавиатурой и "мышкой" может вызывать болевые ощущения в пальцах рук, кистях, запястьях, предплечьях и локтевых суставах. Длительное пребывание в неподвижной, неудобной позе приводит к усталости и болям в позвоночнике, шее, плечевых суставах и мышцах спины.

Третья группа – напряженность труда: работа с компьютером предполагает визуальное восприятие и анализ больших объемов информации, что вызывает утомление зрительного аппарата человека и перегрузку его мозга и центральной нервной системы.

При организации рабочего места оператора ПЭВМ должны быть соблюдены следующие основные условия:

а) допустимые параметры микроклимата;

б) допустимые уровни электромагнитное излучение;

в) эргономичность рабочего места и используемых устройств;

г) оптимальное размещение оборудования, входящего в состав рабочего места;

д) достаточное рабочее пространство, позволяющее осуществлять все необходимые движения и перемещения;

е) обеспечение естественным и искусственным светом в пределах нормы;

ж) контроль уровня акустического;

з) вентиляция рабочего места;

и) обеспечение электробезопасности и пожаробезопасности.

## 5.3 Общие положения организации рабочего места

Согласно СанПиН 2.2.2/2.4.1340-03, помещения для работы с компьютерами должны иметь естественное и искусственное освещение, оборудоваться системами отопления, кондиционирования воздуха или эффективной приточно-вытяжной вентиляции.

Эксплуатация ПЭВМ в помещениях без естественного освещения допускается только при соответствующем обосновании и наличии положительного санитарно-эпидемиологического заключения, выданного в установленном порядке.

Естественное и искусственное освещение должно соответствовать требованиям действующей нормативной документации. Окна в помещениях, где эксплуатируется вычислительная техника, преимущественно должны быть ориентированы на север и северо-восток. Оконные проемы должны быть оборудованы регулируемыми устройствами типа: жалюзи, занавесей, внешних козырьков и др.

Площадь на одно рабочее место пользователей ПЭВМ с ВДТ на базе электронно-лучевой трубки (ЭЛТ) должна составлять не менее 6 м2, в помещениях культурно-развлекательных учреждений и с ВДТ на базе плоских дискретных экранов (жидкокристаллические, плазменные) - 4,5 м2.

При использовании ПВЭМ с ВДТ на базе ЭЛТ (без вспомогательных устройств - принтер, сканер и др.), отвечающих требованиям международных стандартов безопасности компьютеров, с продолжительностью работы менее 4-х часов в день допускается минимальная площадь 4,5 м2 на одно рабочее место пользователя (взрослого и учащегося высшего профессионального образования).

Для внутренней отделки интерьера помещений, где расположены ПЭВМ, должны использоваться диффузно отражающие материалы с коэффициентом отражения для потолка - 0,7 - 0,8; для стен - 0,5 - 0,6; для пола - 0,3 - 0,5.

Полимерные материалы используются для внутренней отделки интерьера помещений с ПЭВМ при наличии санитарно-эпидемиологического заключения. Помещения, где размещаются рабочие места с ПЭВМ, должны быть оборудованы защитным заземлением в соответствии с техническими требованиями по эксплуатации.

Не следует размещать рабочие места с ПЭВМ вблизи силовых кабелей и вводов, высоковольтных трансформаторов, технологического оборудования, создающего помехи в работе ПЭВМ. Расчет воздухообмена следует проводить по теплоизбыткам от оборудования, людей, солнечной радиации и искусственного освещения. Параметры микроклимата, ионного состава воздуха, содержание вредных веществ в нем должны отвечать нормативным требованиям. Звукоизоляция помещений и звукопоглощение ограждающих конструкций помещения должны отвечать гигиеническим требованиям и обеспечивать нормируемые параметры шума на рабочих местах. Помещения должны иметь естественное и искусственное освещение.

В помещениях ежедневно должна проводиться влажная уборка.

Помещения с компьютерами должны быть оснащены аптечкой первой помощи и углекислотными огнетушителями.

## 5.4 Обеспечение параметров микроклимата

Производственные помещения, в которых работа с ПЭВМ является основной (операторские, расчетные, залы вычислительной техники и др.), должны обеспечиваться оптимальные параметры микроклимата. В соответствии с СанПиН 2.2.2/2.4.1340-03 устанавливаются категории тяжести работ 1а и 1б. Вид работы в случае разработки программного продукта «Видео-конференц связь» – основной. Продолжительность работы оператора за ПЭВМ подразумевает 50% времени от смены. Все работы производятся сидя и не требуют особо тяжёлого физического напряжения, расход энергии составляет до 120 ккал/час, поэтому необходимо обеспечить сотрудника помещением с учётом оптимальных норм микроклимата для категории работ 1а.

Температура воздуха на рабочем месте в холодный период года должна быть от 21 до 23°С, в теплый период года — от 22 до 24°С. Разница температуры на уровне пола и уровне головы оператора в положении сидя не должна превышать З°С. Относительная влажность воздуха на рабочем месте оператора должна составлять 40—60%. Скорость движения воздуха на рабочем месте оператора должна быть 0,1 м/с - в холодный период года и 0,2 м/с – в теплый период года.

На рабочем месте, где проводится работа, температура и влажность воздуха должны удовлетворять указанным нормам. Система отопления состоит из основной и вспомогательной системы. Основная система – система приточной вентиляции с подогревом воздуха, также выполняющая роль системы вентиляции помещения, вспомогательная – обогрев помещения за счет батарей центрального отопления.

Проветривание и вентиляция воздуха в помещениях позволяют поддерживать требуемые параметры микроклимата, а также поддерживать постоянный уровень ионизации воздуха. Для повышения влажности воздуха в помещениях с компьютерами следует также применять увлажнители воздуха, заправляемые ежедневно дистиллированной или прокипяченной питьевой водой.

В помещениях, оборудованных ПЭВМ, также необходимо проводить ежедневную влажную уборку после каждого часа работы на ПЭВМ.

Недостаток аэроионов пагубно сказывается на здоровье пользователя, у него снижается иммунитет к различным заболеваниям. Нормы содержания аэроионов в воздухе производственного помещения отражены в таблице 11.

**Таблица 11.** Нормы содержания аэроионов в воздухе рабочих помещений

|  |  |  |
| --- | --- | --- |
| Уровни ионизации | Число ионов в 1 см3 воздуха помещения | |
| положительных | отрицательных |
| Минимально необходимые | 400 | 600 |
| Оптимальные | 1 500 - 3 000 | 3 000 - 5 000 |
| Максимально допустимые | 50 000 | 50 000 |

Содержание вредных химических веществ в производственных помещениях не должно превышать предельно допустимых концентраций (ПДК) загрязняющих веществ в атмосферном воздухе населенных мест в соответствии с действующими гигиеническими нормативами (ГН 2.1.6.1338-03).

## 5.5 Эргономичность

Проектирование рабочих мест, снабженных видеотерминалами, относится к числу важнейших проблем эргономического проектирования в области вычислительной техники. Визуальные эргономические параметры ВДТ являются параметрами безопасности, и их неправильный выбор приводит к ухудшению здоровья пользователей.

Рабочее место и взаимное расположение всех его элементов должно соответствовать (согласно СанПиН 2.2.2./2.4.1340-03) антропометрическим, физическим и психологическим требованиям. Большое значение имеет также характер работы. В частности, при организации рабочего места программиста должны быть соблюдены следующие основные условия:

а) оптимальное размещение оборудования, входящего в состав рабочего места;

б) достаточное рабочее пространство, позволяющее осуществлять все необходимые движения и перемещения;

в) необходимо естественное и искусственное освещение для выполнения поставленных задач;

г) уровень акустического шума не должен превышать допустимого значения;

д) достаточная вентиляция рабочего места.

Главными элементами рабочего места программиста являются письменный стол и кресло. Основным рабочим положением является положение сидя. Работа с вычислительной техникой занимает большую часть времени (до 8 часов машинного времени в сутки). Чтобы такая работа не приводила к быстрому утомлению необходимо создать комфортные для работы условия. Нормальная и безопасная работа пользователя ЭВМ (оператора) во многом зависит от того, в какой мере условия его работы соответствуют оптимальным: комплекс физических, химических, биологических и психофизиологических факторов.

Выбор рабочей позы производится из следующих соображений: рабочая поза сидя вызывает минимальное утомление программиста. Рациональная планировка рабочего места предусматривает четкий порядок и постоянство размещения предметов, средств труда и документации. Высота рабочей поверхности стола для взрослых пользователей должна регулироваться в пределах 680 - 800 мм; при отсутствии такой возможности высота рабочей поверхности стола должна составлять 725 мм. При проектировании рабочего места тщательно подбирается рабочее кресло (подлокотники и подножки, регулировка по высоте, горизонту, по углу наклона спинки, их размеры и свободное пространство вокруг кресла), стол (оптимальная высота – 750 мм над уровнем пола), а также дополнительные тумбы и полки. Рабочий стол должен иметь пространство для ног высотой не менее 600 мм, шириной - не менее 500 мм, глубиной на уровне колен - не менее 450 мм и на уровне вытянутых ног - не менее 650 мм. Рабочий стул (кресло) должен быть подъемно-поворотным и регулируемым по высоте и углам наклона сиденья и спинки, а так же - расстоянию спинки от переднего края сиденья.

### 5.5.1 Оптимальное размещение оборудования

Площадь на одно рабочее место с компьютером для взрослых пользователей должна составлять не менее 6,0 м2, а объем - не менее 20,0 м3.

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

### 5.5.2 Обеспечение электробезопасности

Производителями персональных компьютеров предусмотрены все существующие способы обеспечения электробезопасности. Конструкция использованного в дипломной работе компьютера обеспечивает надежную электробезопасность для работающего с ним человека: по способу защиты от поражения электрическим током удовлетворяет требованиям 1 класса ГОСТ 12.2.007.0 и ГОСТ Р50377; по обеспечению электробезопасности обслуживающего персонала соответствует ГОСТ Р50377.

Защита от поражения электрическим током обеспечивается различными способами: размещением разъемов электропитания на тыльной стороне системного блока и монитора; применением надежных изоляционных материалов; использованием для электропитания клавиатуры, ручных манипуляторов, в интерфейсных кабелях и в элементах регулировки и индикации на лицевой панели системного блока и монитора низковольтных напряжений (не более 12В).

Системный блок и монитор подключены к трехфазной сети переменного тока напряжением 220 В и частотой 50 Гц.

### 5.5.3 Обеспечение допустимого уровня шума

Основной вклад в шум в помещении вносят работающие вентиляторы систем кондиционирования.

В рабочем пространстве, где осуществляется речевой обмен информацией, уровень шума должен быть менее 55 дБ.

При выполнении основной работы на ВДТ и ПЭВМ уровень шума на рабочем месте не должен превышать 60 дБа. На рабочих местах в помещениях для размещения шумных агрегатов вычислительных машин (АЦПУ, принтеры и т.п.) уровень шума не должен превышать 75 дБА. Шумящее оборудование (АЦПУ, принтеры и т.п.), уровни шума которого превышают нормированные, должно находиться вне помещения с ВДТ и ПЭВМ.

Допустимые значения уровней звукового давления в октавных полосах частот и уровня звука, создаваемого ПЭВМ приведены в таблице 12.

**Таблица 12.** Допустимые значения уровней звукового давления

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Уровни звукового давления в октавных полосах со среднегеометрическими частотами | | | | | | | | | Уровни звука в дБА |
| 31,5 Гц | 63Гц | 125 Гц | 250 Гц | 500Гц | 1000 Гц | 2000 Гц | 4000 Гц | 8000 Гц |
| 86 дБ | 71 дБ | 61 дБ | 54 дБ | 49 дБ | 45 дБ | 42 дБ | 40 дБ | 38 дБ | 50 |

Измерение уровня звука и уровней звукового давления проводится на расстоянии 50 см от поверхности оборудования и на высоте расположения источника (источников) звука.

### 5.5.4 Обеспечение допустимых эргономических характеристики дисплеев

Требования к набору параметров для расчета характеристик дисплея отображены в таблице 13.

**Таблица 13.** Допустимые параметры дисплея

|  |  |  |
| --- | --- | --- |
| № | Параметры | Допустимые значения |
| 1. | Яркость белого поля | Не менее 35 кд/кв. м |
| 2. | Неравномерность яркости рабочего поля | Не более +-20 |
| 3. | Контрастность (для монохромного режима) | Не менее 3:1 |
| 4. | Временная нестабильность изображения (мелькания) | Не должна фиксироваться |
| 5. | Пространственная нестабильность изображения (дрожание) | Не более 2 Х 10 (-4L), где L – проектное расстояние наблюдения, мм |

Наиболее критичным параметром, влияющим на физическое здоровье оператора ПЭВМ, является временная нестабильность изображения. Не рекомендуется использовать дисплеи, имеющие данный показатель меньше 80Гц.

### 5.5.5 Обеспечение пожаробезопасности

Помещения, где располагается разработанный модуль или где он будет функционировать, относятся к категории "В" пожаробезопасности - в них находятся твердые сгораемые вещества, не способные взрываться при контакте с кислородом воздуха.

При использовании ПЭВМ возможны аварийные ситуации: короткие замыкания, перегрузки, повышение переходных сопротивлений в электрических контактах, перенапряжение, возникновение токов утечки. Эти случаи могут привести к резкому выделению тепловой энергии, которая может стать причиной возникновения пожара. Для снижения вероятности подобной ситуации необходимо, чтобы помещения были оснащены противопожарной сигнализацией, ручными углекислотными огнетушителями, например, ОУ-3.

Требования к пожаробезопасности зданий и сооружений определяются согласно СНиП 21.01-97.

## 5.6 Расчет системы освещения

Очень важную роль в организации рабочего места играет правильное освещение. Недостаточность освещения приводит к напряжению зрения, ослабляет внимание, приводит к наступлению преждевременной утомленности. Чрезмерно яркое освещение вызывает ослепление, раздражение и резь в глазах. Неправильное направление света на рабочем месте может создавать резкие тени, блики, дезориентировать работающего.

В этой части раздела приведен расчет требуемой системы освещения для помещения, в котором проводилась работа над дипломным проектом. Параметры помещения L × B × H – 4,8 м × 3,8 м × 2,8 м (Планировку смотрите на рисунке 5.1). Задачей данного светотехнического расчета является определение мощности всей осветительной установки для получения заданной освещенности на рабочих местах, при выбранном типе и расположении светильников.

Расчет освещенности производится в следующем порядке:

а) выбор источников света;

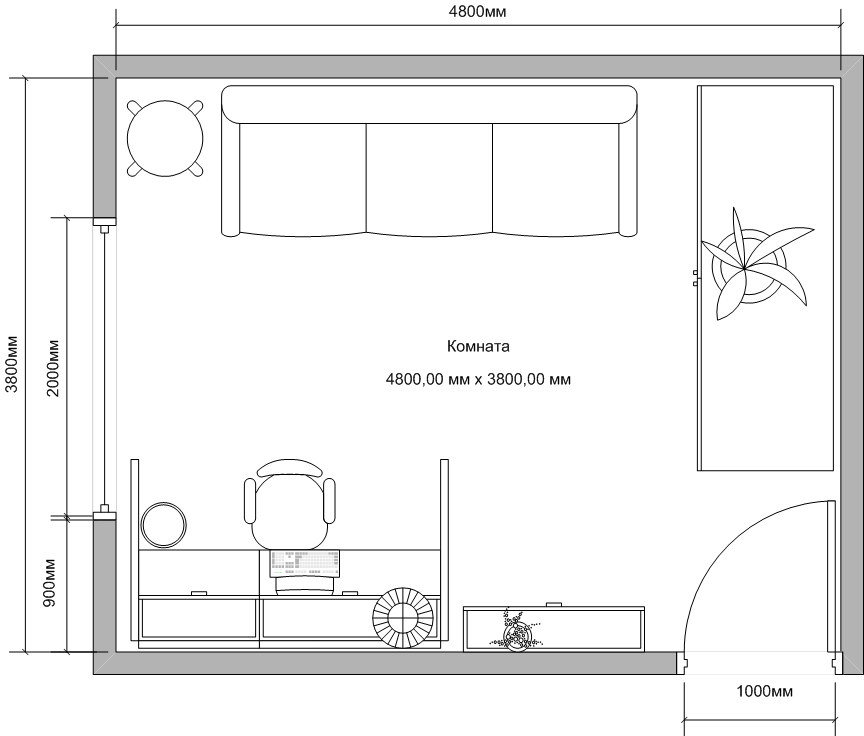
б) выбор системы освещения;

в) выбор типа осветительных приборов и определение высоты их подвеса над рабочей поверхностью;

г) размещение осветительных приборов и определение их количества;

д) выбор освещенности и коэффициента запаса;

е) расчет осветительной установки.



**Рис. 5.1.** Расположение рабочих мест относительно оконных проемов

### 5.6.1 Выбор источников света

К числу источников искусственного света, выпускаемых отечественной и зарубежной промышленностью, относятся лампы накаливания, люминесцентные лампы и лампы ДРЛ. Предпочтение чаще всего отдается люминесцентным лампам, как более экономичным и обладающим более благоприятной цветностью излучения. В системах одного общего освещения офисных помещений, конструкторских бюро и т.п., а также для общего освещения в системе комбинированного освещения во всех случаях также рекомендуется использовать люминесцентные лампы.

### 5.6.2 Выбор системы освещения

В практике проектирования осветительных установок используются следующие системы освещения:

а) общего освещения;

б) комбинированного освещения;

Общее освещение в свою очередь подразделяется на общее равномерное (при равномерном распределении светового потока без учета положения оборудования) и общее локализованное (при распределении светового потока с учетом расположения рабочих мест).

Для применения в офисных помещениях, где нет теней на рассматриваемой поверхности (помещение для работы инженеров - разработчиков) рекомендуется система общего равномерного освещения.

### 5.6.3 Выбор осветительных приборов

При проектировании осветительной установки следует руководствоваться основными показателями выбора светильника: его конструктивное исполнение с учетом условий среды, его светораспределение, экономичность.

Для нашего случая выбираем открытые двухламповые люминесцентные светильники ОДОР – они рекомендуются для нормальных помещений с хорошим отражением потолка и стен, допускаются при умеренной влажности и запыленности. Их минимальная высота подвеса составляет 3,5 метра.

### 5.6.4 Размещение осветительных приборов

Критерии расположения светильников в помещении:

а) обеспечение высокого качества освещения, ограничение ослеплённости и необходимой направленности света на рабочее место;

б) наиболее экономичное создание нормированной освещенности.

Размещение рабочих столов должно быть таким образом, чтобы видеодисплейные терминалы были ориентированы боковой стороной к световым проемам, чтобы естественный свет падал преимущественно слева.

Для равномерного общего освещения светильники с люминесцентными лампами должны располагаться рядами параллельно стенам с окнами.

Относительное расстояние – *λ*, между светильниками должно определяться по формуле 5.1:

*λ=L/h*, (5.1)

где *L* – расстояние между соседними светильниками, м;

*h* – высота подвеса светильника над рабочей поверхностью, м.

При этом различают наиболее оптимальное светотехническое расположение светильников *λС*, при котором достигается наибольшая равномерность освещенности по площади помещения, и энергетически наиболее оптимальное расположение *λЭ*, когда обеспечивается нормируемая освещенность при наименьших энергетических затратах.

Для светильников ОДОР *λС* = 1,4. Показатель *λЭ* для светильников ОДОР не используется. Таким образом, по формуле 5.2 определяем *h*.

*h = H – 0,75 = 2,8 – 0,75 = 2,05* (м), (5.2)

где 0,75 – средняя высота стола,

L = 1,4 \* h = 1,4 \* 2,05 = 2,87 (м), (5.3)

Число рядов светильников в помещении Nb=1,т.к. Nb=3,8:2,87=1,36.

Число светильников в ряду Na =2,т.к. Na=4,8:2,87=1,67.

Двухламповые светильники располагаем в ряд с 2 светильниками, т.о. с 4 лампами.

### 5.6.5 Выбор освещенности и коэффициента запаса

Основные требования и значения нормируемой освещенности рабочих поверхностей изложены в санитарных нормах и правилах. Освещённость должна быть в пределах 300 - 500 Лк. Освещение не должно создавать бликов на поверхности экрана. Нормированная минимальная освещенность (Лк) определяется по таблице 1 разд.5.3 СниП 23-05-95. Работу инженера, в соответствии с этой таблицей, можно отнести к разряду точных работ (3 разряд зрительной работы, подразряд В), следовательно, освещенность должна быть *Е < = 300* Лк при использовании газоразрядных ламп. Коэффициент запаса *К*, учитывающий уменьшение светового потока лампы в результате загрязнения светильников в процессе эксплуатации (его значение определяется по таблице 3 разд. 5.3 СНиП 23-05-95). Для данного расчёта *К* = 1.2.

### 5.6.6 Расчет системы искусственного освещения

Определим световой поток каждой лампы для создания нормируемой освещенности. Методически различают два способа расчета: метод коэффициента использования светового потока и точечный метод.

В расчётах взят метод коэффициента использования светового потока. Он пригоден для помещений с равномерным размещением светильников; при расчете учитывается световой поток, отраженный от стен и потолка, а поэтому данный метод наиболее пригоден для помещений со светлыми потолком и стенами, особенно при использовании светильников рассеянного и отраженного света. Метод пригодится для определения освещенности только на горизонтальной поверхности, не применим для расчета локализованного освещения. Широко используется для расчета осветительных установок с люминесцентными лампами в конструкторских бюро.

Световой поток *F* двух ламп светильника рассчитывается по формуле 5.4:

, (5.4)

где – выбранная нормируемая освещенность, лк;

*S* – площадь помещения, м2;

*k* – коэффициент запаса;

*z* – отношение средней освещенности к минимальной (z = 1,1 – 1,15);

*N* – число светильников – 2 шт.;

- коэффициент использования светового потока ламп, зависящий от типа светильника, коэффициентов отражения потолка и стен и индекса помещения *i*.

= 300Лк (в соотв. с СанПиН 2.2.2.1340-03 раздел 6). Индекс помещения выражает геометрические соотношения в помещении и определяется по формуле 5.5 :

, (5.5)

где *h* – высота подвеса светильника над рабочей поверхностью;

*A=L* и *B* – длинна и ширина помещения соответственно.

Производим расчеты:



Принимаем =50%, =30%. По табличным данным определяем =41%.

 лм (на один двухламповый светильник).

После определения светового потока, подбираем ближайшую стандартную лампу по таблице 14.

**Таблица 14.** Тип лампы в соответствии со значением светового потока

|  |  |
| --- | --- |
| Люминесцентные лампы | |
| Тип лампы | Световой поток, лм |
| ЛДЦ 20 | 820 |
| ЛД 20 | 920 |
| ЛБ 20 | 1180 |
| ЛДЦ 30 | 1450 |
| ЛД 30 | 1640 |
| ЛБ 30 | 2100 |
| ЛДЦ 40 | 2100 |
| ЛД 40 | 2340 |
| ЛБ 40 | 3000 |
| ЛДЦ 80 | 3560 |
| ЛД 80 | 4070 |
| ЛБ 80 | 5220 |

В практике допускается отклонение потока выбранной лампы от расчетного до –10 и +20%, в противном случае задается другая схема расположения светильников. В нашем случае для двухлампового светильника выбираем лампу ЛД 80. Тогда световой поток двухлампового светильника составляет F=4070\*2=8040 (лм) выбранный.

В конце расчета подсчитаем фактическое значение минимальной освещенности рабочей поверхности с учетом выбранной лампы:  лк.

Потребная мощность всей осветительной установки (2 двухламповых светильников) определяется по формуле 5.6:

 (5.6)

где P – мощность одной лампы, Вт.

### 5.6.7 Утилизация элементов системы искусственного освещения

Для освещения были выбраны лампы ЛДЦ 80, которые относятся к классу ртутьсодержащих ламп. Утилизация данного вида ламп соответствует Постановлению № 681 Правительства Российской Федерации от 3 сентября 2010 г. "Об утверждений правил обращения с отходами производства и потребления в части осветительных устройств, электрических ламп, ненадлежащие сбор, накопление, использование, обезвреживание, транспортирование и размещение которых может повлечь причинение вреда жизни, здоровью граждан, вреда животным, растениям и окружающей среде".

## 5.7 Выводы

В данном разделе дипломной работы был проведен анализ всех вредных и опасных факторов, воздействующих на разработчика, выявлен самый неблагоприятный фактор – условия зрительного восприятия, и проведен расчет оптимальной осветительной установки, обеспечивающей нормированное освещение на рабочем месте.

В результате расчетов была разработана осветительная установка, состоящая из 6-ти двухламповых светильников общей потребляемой мощностью 320 Ватт. Фактическое значение минимальной освещенности рабочей поверхности при использовании такой установки составляет 275 Лк при норме 300 Лк.

Мероприятия по обеспечению безопасности труда являются важнейшей составляющей любого производственного и не производственного процесса. При соблюдении их своевременной реализации и проработки зависит здоровье людей, и, как следствие, экономическая эффективность работы предприятия.

# ЗАКЛЮЧЕНИЕ

На примере разработанной системы видеоконференцсвязи в режиме реального времени показаны эффективность и актуальность использования платформы Windows Azure на основе облачных технологий.

Система, разработанная в рамках дипломного проекта, позволит существенно изменить отношение студентов к усвоению образовательного, научно-исследовательского материала. Благодаря системе видеоконференцсвязи студенты, стремящиеся к знаниям, смогут удалённо общаться с преподавателями и решать возникшие вопросы, смогут пересматривать записи лекций и конференций. Если говорить о среднестатистических студентах, то они всегда смогут просмотреть коллоквиумы, презентации, выступления своих лекторов, что значительно поможет в усвоении материала при подготовке к экзаменам, зачётам.

Видеоконференцсвязь, как услуга, будет интересна для абитуриентов. Выпускники школ, стремящиеся получить образование в МГТУ им.Н.Э. Баумана, участвуют в конкурсе «Шаг в будущее». Абитуриентам необходима квалифицированная консультация по интересующей их теме. Эту помощь могут оказать преподаватели нашего вуза, используя приложение.

Система удобна и проста в использовании. Всё, что необходимо – это доступ в Интернет. Затраты на коммуникацию сводятся к минимуму, т.к. пользователи платят за время использования услуг: хранение информации, предоставление вычислительных ресурсов. Использование Azure Table Storage обеспечивает передачу информации в реальном времени, а также хранение видеоматериала. Оплата облачных сервисов осуществляется с помощью подписок, что позволяет сэкономить на покупке и обслуживании программного существенно разработанное изменить в рамках отношение дипломного студентов обеспечения и оборудования.

# СПИСОК ЛИТЕРАТУРЫ

1. Джефф Левинсон Джефф. Тестирование ПО с помощью Visual Studio 2010.: Пер. с англ. – М.: ЭКОМ Паблишерз, 2012. – 336 с.: ил.
2. Кристиан Нейгл, Билл Ивьен и др. C# и платформа .NET 4 для профессионалов.: Пер. с англ.− М.: ООО «И.Д. Вильямс», 2011. – 1440 с.: ил.
3. Меняев М.Ф., Бышовец Б.Д., Пряников И.Ф. Организационно-экономическая часть дипломных проектов, направленных на разработку программного обеспечения. Учебное пособие. – М.: Изд-во МГТУ им. Н.Э.Баумана, 2007. – 30 с.
4. Выполнение организационно-экономической части дипломного проекта по разработке и использованию программного продукта: Методическое пособие. – М.: Изд-во МГТУ им. Н.Э. Баумана, 2006. – 60 с.: ил.
5. Программное решение на платформе Windows Azure. http://www.msdeveloper.ru/magazine/download.aspx/38/item-415.
6. Облачные сервисы. Взгляд из России. Под ред. Е. Гребнева. – М.: CNews, 2011. – 282 с.
7. ГОСТ 2.105-95. Общие требования к текстовым документам.