**Министерство образования и науки РФ**

Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования «Томский государственный университет систем управления и радиоэлектроники» **(ТУСУР)**

Кафедра автоматизации обработки информации (АОИ)

УТВЕРЖДАЮ

Зав. кафедрой АОИ

д-р техн.наук, проф.

\_\_\_\_\_\_\_\_\_\_\_\_Ю.П.Ехлаков

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_г.

АИС приема и обработки заявок на обслуживание (интеграция с системой учета персональных данных сотрудников)

Отчет по преддипломной практике

Студент гр.з-429-б

О.С. Алексеев

«06» марта 2014г.

Руководитель от Руководитель практики

университета \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

«\_\_»\_\_\_\_\_\_\_\_\_\_20\_\_г. «\_\_»\_\_\_\_\_\_\_\_\_\_20\_\_г.

2014

**Министерство образования и науки РФ**

Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования «Томский государственный университет систем управления и радиоэлектроники» **(ТУСУР)**

Кафедра автоматизации обработки информации (АОИ)

УТВЕРЖДАЮ

Зав. кафедрой АОИ

д-р техн.наук, проф.

\_\_\_\_\_\_\_Ю.П.Ехлаков

«\_\_\_»\_\_\_\_\_\_\_\_20\_\_г.

ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ

на преддипломную практику

студенту\_\_\_\_\_\_\_\_Алексееву\_Олегу\_Сергеевичу\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

группа\_\_\_\_\_\_з-429-б\_\_\_\_\_\_\_\_\_\_\_\_факультет\_\_\_\_\_\_\_систем\_управления\_\_\_\_\_\_\_

1. Тема задания\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

2. Вопросы и задачи, подлежащие разработке\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_а)\_Анализ\_предметной\_области,\_построение\_архитектуры\_системы\_\_\_\_\_\_\_\_\_

\_\_б)\_Изучение\_шаблонов\_проектирования\_для\_возможного\_применения\_в\_ито-\_\_\_\_говой\_системе\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_в)\_Выбор\_платформы\_реализации\_и\_каркасов\_разработки\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

3. Методы исследования\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_а)\_Изучение\_проблемы\_сервисного\_обслуживания\_на\_предприятии\_\_\_\_\_\_\_\_\_\_\_б)\_Чтение\_документации\_на\_платформы\_и\_фреймворки\_для\_реализации\_\_\_\_\_\_в)\_Применение\_опыта,\_полученного\_в\_процессе\_обучения\_и\_работы\_\_\_\_\_\_\_\_

4.Тема дипломного проекта: \_\_АИС\_приема\_и\_обработки\_заявок\_на\_обслуживание\_(интеграция\_с\_системой\_учета\_персональных\_данных\_сотрудников)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

5.Перечень задач и работ на период дипломирования \_\_а)\_Проектирование\_системы,\_создание\_прототипа\_(если\_он\_необходим)\_\_\_\_\_\_\_б)\_Разработка\_и\_модульное\_тестирование\_компонентов\_системы\_\_\_\_\_\_\_\_\_\_\_\_\_в)\_Тестирование\_системы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_г)\_Внедрение\_системы\_на\_предприятие\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

РУКОВОДИТЕЛЬ\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(должность, место работы, фамилия, имя, отчество)

|  |  |
| --- | --- |
| Задание выдал руководитель «\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_200\_г.  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(подпись) | Задание принял к исполнению  «\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_200\_г.  \_\_\_\_\_\_\_\_\_\_\_(подпись студента) |

**Содержание**

Введение………………………………………………………………………………...4

Сведения о предприятии..……………………………………………………………...5

Основная часть…..……………………………………………………………………...6

Заключение…………………………………………………………………………….15

Список использованных источников………………………………………………...16

Приложение……………………………………………………………………………17

**Введение**

Основной целью прохождения практики в сети магазинов «Русское ТВ» является подготовка к разработке конечного программного продукта – системе приема и обработки заявок на сервисное обслуживание, с интеграцией с личными делами сотрудников предприятия.

Такая система должна решать актуальную на предприятии проблему – работу с постоянными клиентами, их обслуживание. Для этого создается специализированный call-центр, куда будут поступать заявки для распределения по сотрудникам. На данный момент диспетчеры call-центра обрабатывают заявки стандартными средствами Microsoft Office, из-за чего приходится выполнять работу, которую можно переложить на вычислительные системы, например, просмотр графика работы технических специалистов, поиск свободного специалиста, уведомление его средствами электронной почты. Разрабатываемая система должна делать это автоматически, а связанная с этим работа диспетчера должна заключаться только в приеме звонка и заполнении простой формы.

Для разработки такой системы необходимо владение ООП, умение проектировать приложения масштаба предприятия, в том числе, принятие таких важных архитектурных решений, как выбор платформы и каркасов разработки. Таким образом, основная задача для достижения поставленной цели – достижение необходимого уровня подготовки для проектирования системы подобного масштаба. Помимо имеющегося опыта, способом решения этой задачи является разбор шаблонов проектирования корпоративных приложений, шаблонов объектно-ориентированного проектирования и т.д.

Также, для составления доменной модели, адекватной задаче, необходимо детально изучить проблему с точки зрения бизнеса предприятия.

По результатам преддипломной практики необходимо оформить документ требований к приложению.

**Сведения о предприятии**

Сеть магазинов «Русское ТВ» занимается реализацией оборудования и услуг, связанных со спутниковым телевидением. Компания оказывает такие услуги, как установка оборудования, сервисное обслуживание, ремонт оборудования, помощь в настройке и т.д.

Форма собственности предприятия – индивидуальный предприниматель.

Предприятие делится на несколько отделов (рис. 1), каждый отдел обособлен, все отделы подчиняются администрации.



*Рисунок 1 – Организационная схема предприятия*

Разрабатываемая система напрямую касается call-центра, сведения о рабочих графиках сотрудников должны быть доступны в следствие интеграции с внешней системой учета персональных данных сотрудников, разрабатываемой в рамках предприятия для службы, а сервисный отдел должен получать уведомления из системы по электронной почте.

**Основная часть**

1. **Анализ предметной области**

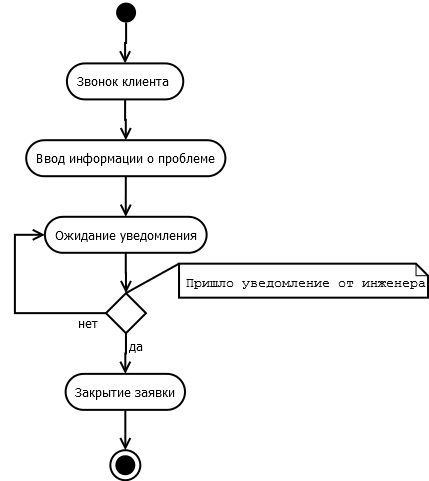
Для автоматизации необходимо детально изучить часть бизнеса предприятия, связанного с приемом заявок call-центром и их обработкой.



*Рисунок 1.1 – Жизненный цикл заявки, работа диспетчера «как есть»*

Основа данного процесса – жизненный цикл заявки (рис. 1.1). Он начинается со звонка клиента в call-центр. Приняв звонок, диспетчер должен выяснить у клиента подробности проблемы. Приняв эту информацию, диспетчер работает уже с ресурсами организации – временем сервис-инженеров: смотрит по рабочему графику, кто работает в нужный день, затем, по расписанию заявок, выясняет, кто свободен в необходимое время. Найдя специалиста, диспетчер делает запись в расписании заявок, затем уведомляет инженера по электронной почте или телефону. После того, как заявка исполнена, инженер уведомляет об этом диспетчера, и диспетчер закрывает заявку.

На данный момент данная проблема не автоматизирована на предприятии, по причине недавнего введения этой части бизнеса. Для автоматизации необходимо разбить жизненный цикл заявки таким образом, чтобы переложить часть работы на вычислительную технику без дублирования работы. На данный момент работа диспетчера включает в себя весь жизненный цикл заявки. Ее можно сократить (рис 1.2), разработав АИС.



*Рисунок 1.2 – Работа диспетчера «как должно быть»*

Таким образом, после внедрения АИС, диспетчеру не нужно будет выполнять работу, связанную с поиском свободного инженера, и с уведомлениями. На данный момент, ручная обработка рабочих графиков инженеров и поиск свободного специалиста по расписанию заявок занимает большую часть времени процесса работы диспетчера с заявкой, при высокой вероятности ошибки. Программная система же может автоматически обработать заранее введенный график и информацию о незавершенных заявках, безошибочно определить свободного специалиста и автоматически назначить заявку на него. Диспетчеру, при таком раскладе, нужно, при получении уведомления по электронной почте или телефону, закрыть заявку.

Нужно учесть, что диспетчеров и сервис-инженеров может быть несколько. Следовательно, для решения проблемы нужна система с быстрым и простым (возможно, одновременным) доступом из нескольких мест. Самый удобный вариант, в связи с этими условиями, разработка WEB-приложения с разграничением прав (диспетчеры, сервис-инженеры) и автоматическими уведомлениями об изменении состояний заявок.

1. **Построение архитектуры АИС**

Современная корпоративная система должна обладать такими качествами, как гибкость, модульность и способность к изменениям без нарушения функциональности. Изменения могут быть различного характера – как в бизнес-правилах (например, добавление дополнительной роли пользователя или статуса заявки), так и в инфраструктуре (например, переход на другого вендора БД, или добавление интеграции с какой-либо внешней системой).

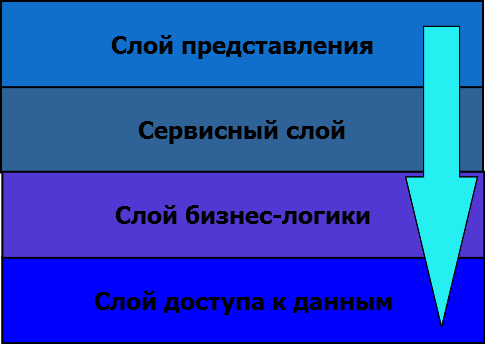
Хорошей практикой для достижения этой цели является разбиение системы на слои (рис 2.1):

* Слой доступа к данным
* Слой бизнес-логики
* Сервисный слой
* Слой представления

В такой архитектуре есть три важных ограничения, за счет которых достигается модульность системы:

1. Взаимодействие между слоями должно происходить «сверху вниз» от слоя представления к слою доступа к данным. При этом нижний слой не должен «знать» ничего о верхнем.
2. Взаимодействовать между собой могут только соседние слои. То есть, слой бизнес-логики взаимодействует со слоем доступа к данным, но не «видит» сервисного слоя и т.д.
3. Слой взаимодействует с нижележащим слоем через абстракции, не привязываясь к конкретной реализации. Соблюдение этого ограничение позволяет разработчику подменять реализации, либо использовать несколько для разных окружений (к примеру, приложение имеет один интерфейс для доступа к данным, реализацию которого можно написать как для реляционной БД, так и для NoSQL).

При соблюдении этих ограничений, разрабатываемая система получает еще одно немаловажное качество – тестируемость. Разработчику предоставляется возможность произвести модульное тестирование каждого слоя по отдельности, подменив слой и объекты, с которыми взаимодействует тестируемый слой, «заглушками», предоставляющими тестовые данные.



*Рисунок 2.1 – Многослойная архитектура системы*

Каждый слой в системе решает свои задачи:

* Слой доступа к данным обеспечивает взаимодействие с источником данных. В его задачи так же входит отображение полученных данных на т.н. объекты передачи данных (data transfer object), предназначенные для доставки данных слою бизнес-логики.
* Слой бизнес-логики содержит в себе объекты, отвечающие за бизнес-процесс, который автоматизирует данная система. На этом слое определяется основное поведение системы, связанное с заявками.
* Сервисный слой содержит функциональность, не связанную с бизнесом, но необходимую в системе (т.н. логика приложения), такую, как уведомление по электронной почте, интеграция с внешними системами и т.д.
* Слой представления отвечает за отображение данных на клиенте. В данном случае это web-страницы, но, при необходимости, остается возможность добавлять клиентов (например, мобильный клиент, консоль, или обычные экранные формы.

Такая архитектура системы лучше всего ложится на объектно-ориентированную парадигму программирования.

**3. Построение информационно-логической модели.**

Объектно-ориентированная парадигма программирования в корпоративных приложениях предполагает разбиение предметной области на пакеты (пространства имен), которые, в свою очередь, разбиваются на классы, каждый из которых решает свою задачу. При этом, хорошей практикой разработки является соблюдение принципов SOLID. SOLID – это аббревиатура, каждая буква которой обозначает один принцип:

* S – Single responsibility (одиночная ответственность). Каждый класс должен решать только одну задачу.
* O – Open/closed (принцип открытости/закрытости). Программные сущности должны быть открыты для расширения, но закрыты для изменения.
* L – Liskov substitution (принцип Лисков). Любой объект в программе может быть подменен своим наследником без изменения свойств программы. Назван в честь польской программистки Барбары Лисков.
* I – Interface segregation (принцип разделения интерфейсов). Несколько специализированных интерфейсов лучше, чем один универсальный.
* D – Dependency Inversion (принцип инверсии зависимостей). Зависимости внутри системы строятся на основе абстракций. Модули верхнего уровня не зависят от модулей нижнего уровня. Абстракции не должны зависеть от деталей. Детали должны зависеть от абстракции.

Проектирование системы целесообразно проводить с соблюдением этих принципов.

Проектирование слоя бизнес-логики данной системы целесообразно начать с декомпозиции основного объекта системы, коим является заявка. Исходя из проанализированных данных call-центра предприятия, было выявлено, что в заявку вносятся:

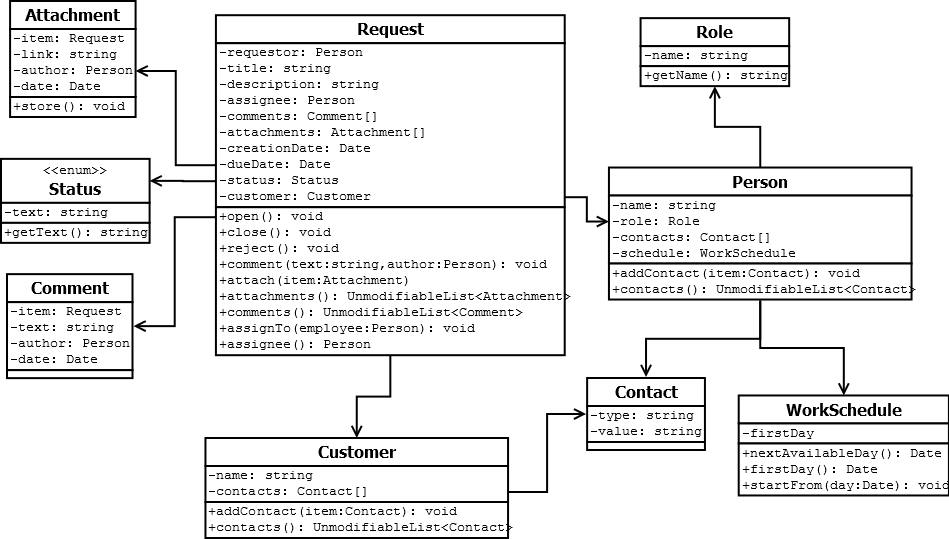
* Данные о диспетчере, принявшем заявку.
* Заголовок, идентифицирующий заявку.
* Краткое описание заявки.
* Данные о заявителе (заказчик, у которого есть договор с предприятием).
* Дата создания заявки.
* Дата крайнего срока исполнения заявки.
* Инженер, на которого назначена заявка.
* Статус заявки (открыта, отложена, не исполнена, отказана, закрыта)
* Комментарии диспетчера и/или инженера.

Из полей заявки целесообразно представить в виде классов: сотрудников предприятия, заказчиков, комментарии к заявкам и статусы. Статусы заявок здесь – перечислимый тип.

В ходе дальнейшей декомпозиции выявлены следующие типы объектов и построен прототип модели бизнес-слоя (рис. 3.1):

* Роль сотрудника на предприятии (диспетчер, инженер).
* Рабочий график сотрудника.
* Контакты (адрес, телефон, e-mail и т.д.)

Здесь объект «Контакт» может относиться как к сотруднику, так и к заказчику.



*Рисунок 3.1 – Прототип модели бизнес-слоя*

1. **Средства разработки АИС.**

Для разработки системы необходимо выбрать платформу, язык и фреймворки, позволяющие решить задачу за минимальное время и с минимальными затратами.

Исходя из опыта разработки приложений, для корпоративных систем самые подходящие и популярные платформы – Java и .NET. Обе платформы имеют достаточно низкий для опытного разработчика порог вхождения и обширную документацию.

Каждая платформа имеет как плюсы, так и минусы, поэтому выбирать нужно, опираясь на поставленную задачу и ограничения. Основные ограничения здесь – время и материальные ресурсы. Время разработки приложений, в данном случае, от платформы не зависит, т.к. процесс разработки на Java и .NET схож. Поэтому, нужно учитывать материальные ресурсы. Сами по себе, обе платформы бесплатны, но для разработки с их помощью нужны такие инструменты, как среда разработки и сервер приложений.

Что касается среды разработки, то для .NET альтернатив практически нет – существует мощная MS Visual Studio. Но, этот вариант для решения задачи не подходит, ввиду его стоимости.

Для разработки Java-приложений существует IntelliJ IDEA, ее стоимость ниже на порядок.

Следующий инструмент – сервер приложений. Для .NET-приложений необходима Windows со встроенным IIS (Internet Information Services). Стоимость этого продукта достаточно высока.

Одна из альтернатив для Java – Oracle WebLogic. Он полностью бесплатен, последние версии доступны на официальном сайте Oracle в свободной загрузке. Соответственно, для сопряжения с сервером приложений, удобнее всего использовать и систему управления базами данных Oracle.

Учитывая схожесть платформ и наложенные на задачу ограничения, выбор в пользу Java очевиден.

Что касается фреймворка, то, исходя из опыта разработки, выбор практически безальтернативно падает на Spring Framework. В среде Java он больше всего подходит для разработки корпоративных приложений, т.к. включает в себя компоненты для таких задач как доступ к данным, обеспечение безопасности, аспектно-ориентированное программирование, программирование уровня представления.

По результатам практики был составлен документ требований к системе (Приложение А).

**Заключение**

По окончанию преддипломной практики были рассмотрены такие важные архитектурные вопросы, как выбор языка и платформы разработки и фреймворком и анализ части бизнеса предприятия, связанной с обработкой заявок. Все решения приняты, исходя из уже полученного опыта работы и наложенных на проект ограничений. Исходя из принятых решений, составлен документ требований к системе.

Принятые решения:

1. Использовать для разработки системы платформу Java.
2. Использовать для разработки системы Spring Framework.
3. Использовать для разворачивания системы сервер приложений Oracle WebLogic.

Система будет разрабатываться по классической многослойной архитектуре, содержащей слой доступа к данным, слой бизнес-логики, сервисный слой и слой представления.

# Список использованных источников

1. Р. Мартин «Чистый код» – Пер. с англ. – Санкт-Петербург, Изд-во «Питер», 2012.-464с.
2. Э. Гамма, Р. Хелм, Р. Джонсон, Д. Влиссидес «Приемы объектно-ориентированного проектирования. Паттерны проектирования» – Пер. с англ. – Санкт-Петербург, Изд-во «Питер», 2012.-368с.
3. Э. Хант, Д. Томас «Программист-прагматик» – Пер. с англ. – Санкт-Петербург, Изд-во «Лори», 2012.-270с.
4. С. Макконнелл «Совершенный код» – Пер. с англ. – М., Изд-во «Русская редакция», 2012.-896 стр., ил

**Приложение А**

Требования к АИС приема и обработки заявок на обслуживание (интеграция с системой учета персональных данных сотрудников)

Общие требования:

1. АИС должна являться корпоративным web-приложением.
2. Система должна быть реализована на платформе Java.
3. Система должна использовать Spring Framework для управления доступа к данным, разграничения прав, разработки слоя представления.
4. Система должна разворачиваться на сервере приложений Oracle WebLogic.
5. Данные приложения должны храниться в базе данных под управлением СУБД Oracle.

Требования к логике приложения:

1. Приложение должно иметь экран авторизации с вводом логина и пароля.
2. Права в приложении должны быть разграничены на администратора, диспетчера и инженера.
3. Приложение должно иметь экран заявки с формой ввода, согласно информации, касающейся запроса от клиента: наименование заявки, описание заявки, контакты клиента, время исполнения заявки.
4. Приложение должно иметь возможность комментирования заявок.
5. В приложении должна быть возможность назначить заявку на свободного специалиста.
6. Сведения о том, какой специалист свободен в назначенной время, должны рассчитываться автоматически.
7. Сведения о специалистах должны синхронизироваться с системой учета персональных данных сотрудников.
8. В системе должна присутствовать возможность построения отчетов по динамике заявок за период.
9. В системе должна присутствовать возможность вывода списка заявок на печать.
10. В системе должна присутствовать возможность вывода деталей заявки на печать.
11. Система должна уведомлять инженера об открытии заявки по электронной почте.
12. Система должна уведомлять инженера об изменении статуса заявки по электронной почте.
13. Инженер должен иметь возможность изменения статуса заявки.
14. Система должна уведомлять диспетчера об исполнении инженером статуса заявки.
15. Диспетчер и инженер должны иметь возможность комментирования заявки.
16. Диспетчер и инженер должны иметь возможность присоединять файлы к заявке.

Ориентировочный срок установки системы в окружение предприятия – 31 мая 2014 года.

Руководитель от предприятия \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/Верещагин Р. Е./