ФЕДЕРАЛЬНОЕ АГЕНТСТВО ВОЗДУШНОГО ТРАНСПОРТА

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ ГРАЖДАНСКОЙ АВИАЦИИ» (МГТУ ГА)

Кафедра вычислительных машин, комплексов, систем и сетей

Н.И. Романчева

ИНТЕРНЕТ-ТЕХНОЛОГИИ

Учебно-методическое пособие
по выполнению курсовой работы

для студентов
направления 09.03.01
очной формы обучения

Москва
ИД Академии Жуковского
2018

УДК 004.738.5(07)
ББК 6Ф7.3
Р69

Рецензент:
Егорова А.А. – д-р техн. наук, доц. каф. ПМ

Романчева Н.И.
Р69 Интернет-технологии [Текст] : учебно-методическое пособие по выполнению курсовой работы / Н.И. Романчева. – М. : ИД Академии Жуковского, 2018. – 32 с.

Данное учебно-методическое пособие издается в соответствии с рабочей программой учебной дисциплины «Интернет-технологии» по учебному плану для студентов направления 09.03.01 очной формы обучения.

Рассмотрено и одобрено на заседании кафедры 26.12.2017 г. и методического совета 26.12.2017 г.

УДК 004.738.5(07)
ББК 6Ф7.3

В авторской редакции

Подписано в печать 26.04.2018 г.
Формат 60х84/16 Печ. л. 2 Усл. печ. л. 1,86
Заказ № 267/0403-УМП01 Тираж 30 экз.

Московский государственный технический университет ГА
125993, Москва, Кронштадтский бульвар, д. 20

Издательский дом Академии имени Н. Е. Жуковского
125167, Москва, 8-го Марта 4-я ул., д. 6А
Тел.: (495) 973-45-68
E-mail: zakaz@itsbook.ru

ВВЕДЕНИЕ

Курсовая работа по дисциплине Интернет-технологии выполняется обучающимся по направлению подготовки 09.03.01 Информатика и вычислительная техника (бакалавриат) в 8 семестре. В рамках курсовой работы должно быть разработано кроссплатформенное приложение с использованием Интернет-технологий, при выполнении курсовой работы используются знания, полученные студентами при изучении дисциплины Интернет-технологии, а также дисциплин Информатика, Сети и телекоммуникации, Технологии программирования.

1. ЦЕЛИ И ЗАДАЧИ КУРСОВОЙ РАБОТЫ

1.1 Целью курсовой работы является приобретение практических навыков по разработке структуры кроссплатформенного приложения, алгоритмов и программ для их реализации с использованием Интернет-технологий на основе модели клиент-сервис.

1.2 Задачей курсовой работы является формирование компетенций ОПК-2, ОПК-5, ПК-2, ПК-8 в части освоения приемов и методов проектирование web-элементов пользовательского интерфейса и разработки приложения по заданным исходным данным:

- web-приложение контроля посещения учебных занятий;
- разработка приложения в облаке;
- EDL-программа планирования стратегии агента;
- программа управления процессами обработки неструктурированных данных.

2. ОРГАНИЗАЦИЯ И ПОСЛЕДОВАТЕЛЬНОСТЬ ВЫПОЛНЕНИЯ КУРСОВОЙ РАБОТЫ

Курсовая работа является формой самостоятельной работы студента и выполняется по индивидуальному заданию.

Задание на курсовую работу выдается преподавателем на 17-ой неделе 7 семестра, защита проводится на 6 и 7 неделях 8 семестра. К защите представляется пояснительная записка. На защите демонстрируется выполнение программы, доклад - не более 7 минут с презентацией разработанной программной продукции с использованием средств MS PowerPoint.

В ходе выполнения курсовой работы студент консультируется с руководителем, назначенным кафедрой.

За правильность проектных решений, качество оформления работы, своевременность выполнения отдельных этапов и представления к защите отвечает студент.

2.1 Задание на курсовую работу

Задание на курсовую работу выдается преподавателем в соответствии с приведенными вариантами и оформляется обучающимся на бланке, приведенном в приложении Б.

Допускается выполнение курсовой работы по индивидуальному заданию в соответствии с заявкой предприятия, подтвержденного соответствующим письменным обращением на имя декана факультета.

2.2 Объем и содержание курсовой работы

Работа состоит из пояснительной записки (ПЗ), программы, представленной на сменном носителе, справки о заимствовании материала.

2.2.1 Структура курсовой работы и требования к ее содержанию

Работа должна содержать следующие элементы:

- формулировка цели и основных задач исследования; краткая сводка по рассматриваемой научно-практической задаче на основании литературных источников;
- характеристика объекта исследования; обоснования избранного способа решения поставленных задач;
- оценка материалов, привлекаемых к работе;
- описание методики и технологии обработки и анализа исходных данных;
- предложения по совершенствованию существующих технологических схем и методов решения поставленных задач;
- изложение полученных результатов с оценкой их новизны и практической значимости;

в работе должен быть представлен самостоятельно собранный фактический материал.

Курсовая работа включает:

- пояснительную записку;
- иллюстративный материал (графики, схемы, диаграммы и т.п.).

Пояснительная записка формируется в следующей последовательности:

- титульный лист (приложение А);
- задание на курсовую работу (приложение Б);
- аннотация;
- содержание;
- введение;
- основная часть, разбитая на соответствующие разделы, подразделы, пункты;
- заключение;
- список использованных источников;
- Приложения
- спецификация программного обеспечения (приложение В),
- текст программы (приложение Г),

- руководство пользователя (системного программиста) (приложение Д).

Структура пояснительной записки разрабатывается обучающимся совместно с преподавателем-руководителем КР на основе примерной структуры, приведенной ниже.

Наименование разделов КР	Количество страниц записки	Листы графического материала (презентации)
Введение	2	
Основная часть: • Общая часть: -анализ исследуемой задачи, - направления решения и постановка задачи -формулировка технического задания ;	15	2
формулировка возможных вариантов решения основной задачи работы, этапы решения задачи (разработка структуры, алгоритмизация решения и его программная реализация), разработка интерфейса	20	3
	10	1-2
Заключение. Список использованных источников. Приложения: А- спецификация Б – листинг программы В- руководство оператора	1-2 1-2 1 15 10	1
ИТОГО	Не менее 50 стр.	Не менее 6 листов

2.2.2 Требования к ее содержанию курсовой работы

Титульный лист заполняется в соответствии типовым бланком, размещенным на сайте МГТУ ГА. Перед защитой КР титульный лист должен быть подписан студентом и руководителем КР .

Аннотация должна содержать:

-сведения о количестве листов пояснительной записки, содержащихся в ней рисунков и таблиц, о количестве источников и приложений, а также о количестве листов графической документации;

- перечень ключевых слов;

- текст аннотации.

Перечень ключевых слов должен включать от 5 до 15 слов или словосочетаний из пояснительной записки, которые в наибольшей мере характеризуют ее содержание. Ключевые слова приводятся в именительном падеже, прописными буквами в строку через запятые.

Текст аннотации должен содержать: объект исследования или разработки; цель работы; результаты работы и их новизну. Объем текста аннотации – не более 700 знаков.

Во введении отмечается актуальность и значимость темы, степень ее разработанности, а также формулируются цель и задачи работы.

Основной раздел может содержать три раздела, в которых:

- приводится анализ объекта исследования;
- выявляются недостатки и нерешенные проблемы;
- формулируется задание на специальную часть;
- приводятся этапы решения задачи разработка структуры приложения, алгоритма решения задачи, структура программы, проектирование интерфейса приложения (если это предусмотрено заданием на КР);
- приводятся связанные с решением задачи расчеты;
- формулируются основные результаты;
- решаются конкретные задачи, связанные с темой работы.

Содержание основного раздела определяется в зависимости от направления и темы КР.

В заключении подводятся итоги работы, формулируются важнейшие выводы и даются рекомендации о возможности внедрения полученных результатов в практику.

Список использованных источников должен включать не менее 6 наименований учебных, научных и справочных источников. Все источники должны быть разделены на группы:

- нормативные, подзаконные акты и ГОСТы;
- монографии, учебные пособия и справочная литература;
- периодические издания;
- документация предприятия;
- Интернет-источники.

Приложение к курсовой работе может быть представлено в виде иллюстраций, графиков, таблиц, схем и т. д.

2.3 Последовательность выполнения работы

Курсовая работа разрабатывается в последовательности, соответствующей содержанию ПЗ (п.2.2.1).

Расчетно-пояснительная записка и графический материал оформляются в соответствии с требованиями ЕСКД и ЕСПД (Единая система конструкторской документации, Единая система программной документации).

Подготовленная и оформленная работа, прошедшая экспертизу на выполнение требований ЕСКД и ЕСПД представляется преподавателю не позднее, чем за неделю до защиты.

Защита работы происходит на 6 или 7 неделе 8 семестра.

2.4. Защита курсовой работы

Защита курсовой работы проходит перед комиссией, в состав которой входит два преподавателя, включая руководителя КР. Для защиты курсовой работы обучающийся обязан предъявить пояснительную записку, проверенную руководителем КР с резолюцией о допуске к защите, диск, содержащий разработанное ПО КР, доклад с презентацией, справку о заимствовании.

Оценка «отлично» выставляется студенту, обосновавшему актуальность, степень новизны и оригинальности темы и ее решения; доказавшему высокий научно-технический уровень исследования; продемонстрировавшему глубину знаний; использовавшему современные информационные и компьютерные технологии; доказавшему практическую значимость исследования; продемонстрировавшего высокое качество и грамотно изложившего результаты исследования; показавшему общий высокий уровень подготовки по дисциплине.

При этом студент не затрудняется с ответом при видоизменении задания, свободно справляется с задачами, вопросами и другими видами контроля знаний, проявляет знакомство с технической литературой, правильно обосновывает принятые решения, владеет разносторонними навыками и приемами решения практических задач.

Оценка «хорошо» выставляется студенту, обосновавшему актуальность, степень новизны и оригинальности темы и ее решения; доказавшему практическую значимость исследования; продемонстрировавшего хорошее качество и грамотно изложившего результаты исследования; показавшему общий хороший уровень подготовки по дисциплине.

При этом студент не допускает существенных неточностей в ответе на вопрос, правильно применяет теоретические положения при решении практических вопросов и задач, владеет необходимыми приемами их решения.

Оценка «удовлетворительно» выставляется студенту, обосновавшему актуальность темы; продемонстрировавшему достаточное качество; показавшему общий достаточный уровень подготовки по дисциплине.

При этом студент имеет знания только основного материала, но не усвоил его детали, допускает неточности, недостаточно правильные формулировки, нарушения последовательности в изложении материала и испытывает трудности в ответах на вопросы.

Оценка «неудовлетворительно» выставляется студенту, не сумевшему обосновать актуальность, новизну и оригинальность темы; показавшему недостаточный уровень подготовки по дисциплине.

При этом студент неуверенно отвечает или совсем не отвечает на вопросы.

3 ВАРИАНТЫ ЗАДАНИЙ

Задания на курсовую работу являются комплексными и выполняются группой студентов в соответствии с индивидуальными заданиями в рамках общего задания, выдаваемого преподавателем.

Вариант 1

Web-приложение контроля посещения учебных занятий.

Разработать программный комплекс контроля успеваемости студентов. Комплекс реализует клиент-серверную архитектуру. На сервере должна располагаться база данных, содержащая данные на основе учебного плана (название дисциплин, зачетные единицы и др.), списки студенческих групп, списки преподавателей. Программный комплекс включает три приложения: два клиентских приложения (для преподавателей и студентов) и серверное приложение.

Данное приложение должно обеспечивать следующие возможности: создание аккаунта конкретного преподавателя и студента, создание и планирование дисциплины преподавателем, формирование списка студентов в дисциплине, заполнение посещаемости и успеваемости занятий студентами, просмотр промежуточных и итоговых результатов успеваемости через отправку на e-mail адрес.

Кроме того, обеспечен универсальный интерфейс взаимодействия клиентских и серверных компонентов; реализована внутренняя серверная логика; обеспечено стабильное и устойчивое функционирование комплекса; заложены архитектурные приёмы, гарантирующие расширяемость и поддерживаемость программного комплекса.

Вариант 2

Разработка приложения в облаке.

Разработать алгоритм и а) программу обработки или б) программы массовых параллельных операций преобразования на U-SQL, R, Python и .NET с использованием службы Data Like Analytics или Azure HDInsight.

Данное приложение должно обеспечивать: доступ сотрудников к ресурсам фирмы, которая находиться в виде базы данных в облаке (выбор модели развертывания облака студент осуществляет самостоятельно, обосновывая данный выбор в пояснительной записке); удобный и информативный поиск и выдачу сведений необходимых для удовлетворения информационных потребностей пользователей; авторизацию пользователей; оформление заказа с рабочего места пользователя; учет и ведение заказов клиентов фирмы; удобный и простой интерфейс для пользователей; сбор, представление и ведение информационных ресурсов фирмы (например, ресурсов аэропорта и т.п.).

Среда разработки: Visual Studio, Eclipse, IntelliJ, Jupyter.
Рекомендуется использовать элементы голосового интерфейса.

Вариант 3

EDL-программа планирования стратегии агента.

Разработать имитационную модель управления качеством а) обслуживания авиапассажиров, б) технического обслуживания ВС, в) образования, реализованную средствами агентного моделирования, и комплекс программных приложений, осуществляющих поддержку принятия решения в сфере менеджмента качества соответствующей предметной области.

В ходе моделирования должны быть определены время реакции на сообщения, время обработки одного сообщения. Для агента-координатора также определено среднее время, затрачиваемое на распределение одного задания.

Для уменьшения погрешности провести не менее пяти экспериментов со следующими исходными данными на примере варианта «б»: агент-оператор формирует пять заданий для пяти типов ВС, отправляет сформированные задания агенту-координатору; агент-регистратор формирует пять агентов-исполнителей, которые связаны с 5 типами ВС, отправляет ссылки на сформированных агентов агенту-координатору; агент-координатор обрабатывает сообщения согласно разработанному алгоритму, выдавая время распределения одного задания и время обработки одного сообщения; агенты-исполнители имеют после инициализации 5 заданий, непересекающихся во времени, с промежутком между заданиями в 40 минут.

В модели агента-координатора определить идентификатор агента, коллекция сообщений, коллекция агентов-исполнителей, коллекция заданий для распределения, порт для взаимодействия с другими агентами, функции обработки сообщений и распределения заданий. Функции агента выполняются последовательно. Функция распределения заданий в качестве пула агентов использует соответствующую коллекцию агентов-исполнителей, список заданий представлен в виде коллекции заданий.

В модели агента-исполнителя определить идентификатор агента, переменную для временного хранения, выполняемого в данный момент времени задания, коллекция сообщений, коллекция заданий для выполнения, порт для взаимодействия с агентом-координатором, порт для взаимодействия с внешними устройствами, функции обработки сообщений и обработки заданий.

Вариант 4

Разработать программу управления процессами обработки неструктурированных данных.

Провести анализ методов и средств извлечения информации из данных, включая формирование интегрированных, достоверных, эффективно доступных данных для Big Data вычислений.

Исследовать, насколько широко используется известный популярный бренд, сервис (услуга) или проект. Входными данными являются неструктурированные данные из общедоступных источников в Интернет и структурированные данные из СУБД DB2. В процессе работы производится объединение этих разнотипных данных и их совместный анализ с целью определения степени популярности бренда (например, авиакомпания Аэрофлот) в СМИ и эффективность рекламных компаний ПАО Аэрофлот.

Использовать функции построения графиков и агрегации в аналитической утилите BigSheets.

Технологии Biglnsights – Jaql, Flume, Pig, MapReduce и др.

4. ИНФОРМАЦИЯ ДЛЯ ВЫПОЛНЕНИЯ КУРСОВОЙ РАБОТЫ

4.1 Web-приложение контроля посещения учебных занятий

С учётом актуальных требований Российского законодательства в области образования, контроль посещаемости и успеваемости студентов в высших образовательных учреждениях производится по балльно-рейтинговой системе, которая требует не только итогового контроля, но и текущего учёта полученных студентами баллов. В условиях частой смены учебных планов (изменение названия дисциплин или количество часов лекций, практических занятий и т.п.) наиболее остро стоит проблема расчета баллов для учёта и контроля посещаемости и успеваемости студентов.

Успешность изучения отдельных дисциплин и активность студента оценивается суммой набранных баллов, которые в совокупности будут определять рейтинг студента. Рейтинг студента определяется общим средним показателем успеваемости и активности студента. Повышается объективность оценки студенческих достижений в учёбе. В балльно-рейтинговой системе оценка на экзамене добавляет баллы к тем, которые были ранее набраны за семестр. Балльно-рейтинговая система позволяет более точно оценивать качество учёбы. В течение каждого семестра студент может набрать 100 баллов по следующим показателям: посещение лекций и лабораторных занятий и самостоятельная работа студентов - 60 баллов; промежуточная аттестация (контрольная работа, зачёт) - до 30 баллов; до 100 баллов; социальные характеристики - до 10 баллов. Данный показатель, накопленный за весь срок обучения, служит главным признаком успешности освоения студентом образовательной программы и на выходе определяет общий рейтинг выпускника.

Следовательно, по причине большого объёма данных, возникает острая необходимость в автоматизации, которая позволила реализовать следующие задачи: дополнения существующего журнала в бумажном виде; улучшения и ускорения процедуры заполнения журнала; улучшения процесса подачи сведений в деканат; ведения архивов статистической информации о прошедших

семестрах без ограничения сроков давности; обеспечения сбора и первичной обработки исходной информации, необходимой для подготовки отчётности; повышения качества информации; создания единой системы отчётности.

Внедрение информационных технологий в образовательный процесс порой приводит не только к его модернизации, но и к его усложнению, что ведёт к увеличению второстепенной нагрузки на преподавателя. Поэтому в рассматриваемой курсовой работе ставится задача разработать программный комплекс, позволяющий улучшить значения следующих показателей: время сбора и первичной обработки исходной информации; время, затрачиваемое на информационно-аналитическую деятельность. Реализуемые функции программного обеспечения, специфичные для каждой категории клиентов могут быть представлены в виде графической структуры, отображённой на рис. 4.1.

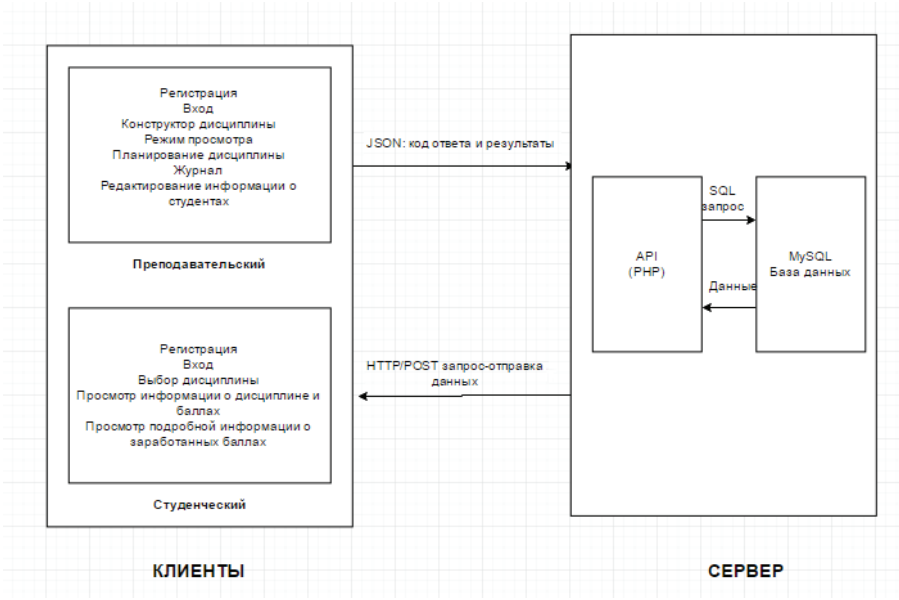

Рисунок 4.1 — Структура программного обеспечения.

Для приложения преподавателя можно выделить:

1. Выбор дисциплин. "Создать" — переход к окну создания новой дисциплины; "Дисциплина 1" — выбор созданной дисциплины; "выбрать" — переход к ранее выбранной созданной дисциплине.
2. Конструктор дисциплины. Выбор данных из справочников: Направление подготовки; группа; форма обучения; итоговая аттестация; курсовая работа. Выбор числового значения из выпадающего списка: количество зачётных единиц; количество лекций; количество практик; количество лабораторных работ; КДЗ. "назад" — переход к странице выбора

дисциплин; "сохранить" — сохранение в базу данных результатов конструктора; "перейти к планированию" — переход к странице планирования дисциплин.

3. Планирование дисциплины. Выбор значения из выпадающего списка: Лекция (№ занятия; посещение; активность); Практика (№ занятия; посещение; активность); Лабораторная работа (№ занятия; посещение; активность); КДЗ (№ занятия; посещение; активность); Курсовая работа (№ занятия; посещение; активность). Выбор данных из справочников: Персональные достижения; Итоговая аттестация. "Вернуться к редактированию" — переход к странице редактирования дисциплин; "Сохранить" — сохранение в базу данных результатов планирования; "Журнал" — переход на страницу с выбором дисциплины, типом занятия и датой.
4. Информация о студенте. Ввод данных вручную: Фамилия; Имя; Отчество; № зачётки. Выбор данных из имеющихся справочников — Логин. "Сохранить" — переход на страницу просмотра информации; "Удалить" — переход на страницу журнала.
5. Режим просмотра. Выбор данных из имеющихся справочников: Дисциплина 1. "Назад" — переход к предыдущей странице; "выбрать" — переход на страницу со списком студентов и их баллами за все занятия.
6. Дисциплина. Просмотр таблицы группы с подгруженными данными из базы. "назад" — переход на страницу с выбором дисциплины для просмотра.
7. Выбор занятия. Выбор значения из выпадающего списка; Дисциплина; Тип занятия; номер занятия/дата. "Режим просмотра" — переход на страницу с баллами за все занятия; "Конструктор дисциплин" — переход на страницу конструктора дисциплин; "Перейти" — переход на страницу заполнения журнала.
8. Журнал. Просмотр и редактирование таблицы группы с подгруженными данными из базы. "Назад" — переход на страницу с выбором занятия и датой; "Применить" — сохранить изменения в существующей базе.

Для приложения студента:

1. Дисциплины. Выбор значений из выпадающего списка — название дисциплины/преподаватель. "Перейти" — переход на страницу с баллами.
2. Предмет и баллы. Информация о выбранном предмете, подгруженного из выбранной базы. "Назад" — переход на страницу с выбором дисциплины"; "Подробнее" — переход на подробное описание заработанных баллов.
3. Общие данные по предмету. Подгрузка информации из выбранной базы по предмету. "Назад" — переход на страницу выбора дисциплин.

Клиентские приложения для студента и преподавателя взаимодействуют с SQL-базой данных через API сервера (PHP), обращающийся к ней через

интерфейс СУБД (PDO). На рис. 4.2 представлена схема взаимодействия компонентов комплекса, таких как «Исходные данные», «Обработка данных», «Представление данных», формирующие клиентское приложение программной модели.

1. Получение исходных данных

Источником данных выступает MySQL база данных. База данных содержит такую информацию как: список дисциплин, список учащихся той или иной группы, их успеваемость и посещаемость и т.д.

2. Обработка данных

Полученные исходные данные обрабатываются, приводятся к единому формату.

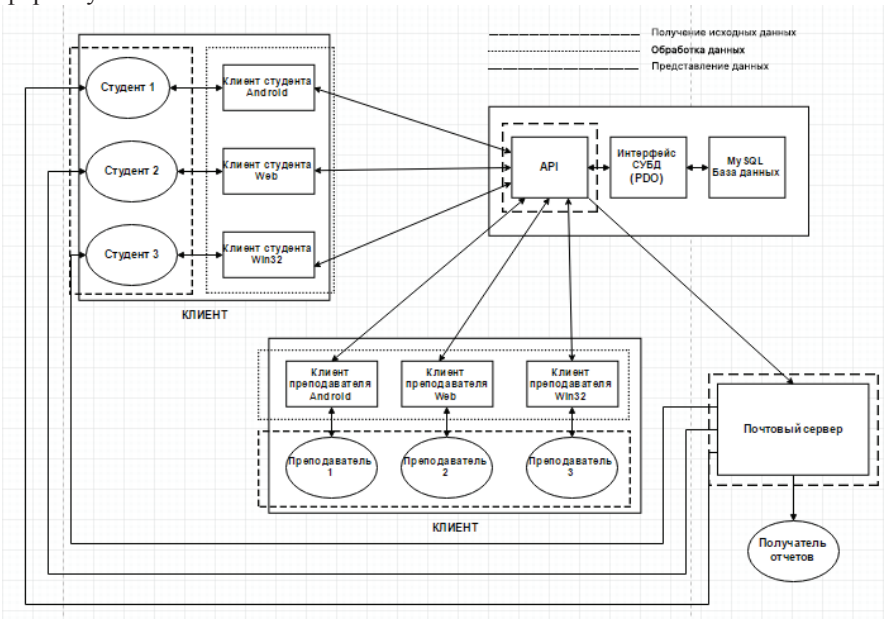

Рисунок 4.2 - Схема взаимодействия компонентов комплекса.

Происходит комбинирование с другими данными, а также для загрузки данных в хранилище данных.

3. Представление данных

Данные, полученные в результате расчётов, представляются в виде информационных панелей, на которых результаты отображаются в форме таблиц.

Блоки взаимодействуют друг с другом следующим образом:

Программа -клиент для студента и преподавателя взаимодействуют с SQL базой данных через API сервера (PHP), обращающийся к ней через интерфейс СУБД (PDO).

Взаимодействие клиента для студента и API сервера:

1. Клиент отправляет POST-запрос, через протокол HTTP, с данными о регистрации. API сервера возвращает код ошибки в формате JSON.
2. Клиент отправляет POST-запрос, через протокол HTTP, с данными для авторизации. API сервера возвращает данные о пользователе и номер сессии или описание ошибки в формате JSON.
3. Клиент отправляет POST-запрос, через протокол HTTP, с данными на восстановление пароля.
4. Клиент отправляет POST-запрос, через протокол HTTP, с данными для подтверждения.

Взаимодействие клиента для преподавателя и API сервера:

1. Клиент для преподавателя отправляет POST-запрос, через протокол HTTP, на получение списка дисциплин преподавателя. API сервера возвращает массив дисциплин в формате JSON.
2. Клиент для преподавателя отправляет POST-запрос, через протокол HTTP, на получение плана дисциплины. API сервера возвращает группы по типам занятий в формате JSON.
3. Клиент для преподавателя отправляет POST-запрос, через протокол HTTP, на получение списка дисциплин дисциплины по указанной секции. API сервера возвращает массив занятий в формате JSON.
4. Клиент для преподавателя отправляет POST-запрос, через протокол HTTP, на получение списка студентов. Клиент для преподавателя возвращает массив студентов в формате JSON.
5. Клиент для преподавателя отправляет POST-запрос, через протокол HTTP, на добавление студентов к дисциплине. API сервера возвращает код ошибки в формате JSON.

Взаимодействие API сервера и SQL базы данных через интерфейс СУБД:

API сервера выполняет SQL-запрос к SQL базе данных через интерфейс СУБД. База данных через интерфейс СУБД возвращает данные.

Взаимодействие API сервера и почтового сервера пользователей:

1. API сервера отправляет код завершения регистрации на почтовый сервер пользователей в виде электронного письма.
2. API сервера отправляет код подтверждения смены пароля на почтовый сервер пользователей в виде электронного письма.

Взаимодействие почтового сервера пользователей и получателя отчётов:

Почтовый сервер пользователей отправляет получателю отчётов таблицу дисциплины в виде электронного письма.

Взаимодействие почтового сервера пользователей и клиента для студента или преподавателя:

1. Почтовый сервер пользователей отправляет клиенту для студента код завершения регистрации через ссылку.
2. Почтовый сервер пользователей отправляет клиенту для студента код подтверждения смены пароля через ссылку.

Серверная часть программного обеспечения представляет собой реализацию интерфейса унифицированного API, которые возможно использовать всеми клиентскими приложениями. Универсальность API позволяет использовать множество фронтендов (например, Web, Android, Win32 и т. д.), что требуетсяь согласно заданию в реализации курсовой работы.

Унифицированное API представляет собой HTTP интерфейс, доступный по протоколу HTTPS (HTTP + TLS) для обеспечения конфиденциальности передаваемых данных. Согласно исходным данным передача аргументов к API обеспечена с помощью запросов POST, а ответы от API поступают в формате JSON. В связи с тем, что данные форматы являются де-факто стандартом при реализации интерфейсов для веб-совместимых приложений, совместимость с ними является встроенным функционалом PHP.

Код серверной части написан на языке PHP с использованием драйвера баз данных PDO и базы данных MySQL. Данный подход не только обеспечивает выполнение исходных данных, но также является максимально гибким, так как позволяет сделать код легко переносимым в рамках множества серверов, а также использовать в дальнейшем другие виды баз данных (драйвер PDO является унифицированным драйвером для MySQL, MariaDB, PostgreSQL, SQLite и др.)

Функционал, общий для клиентских студенческих и серверных приложений, предоставляет стандартные методы регистрации, авторизации, изменения и восстановления пароля, а также выхода из системы.

login:
Производит авторизацию.

STRING api - версия реализуемого API
STRING user - логин пользователя
STRING pass - пароль пользователя
Ответ:
ERROR error - результат
GUID session - сессия пользователя
STRING name - имя пользователя
STRING surname - фамилия пользователя
STRING middlename - отчество пользователя
STRING studid - номер зачётной книжки (отсутствует для преподавателя)

logout:
Производит деавторизацию.

STRING api - версия реализуемого API
GUID session - сессия пользователя
Ответ:
ERROR error - результат

recover:
Производит восстановление пароля.

STRING api - версия реализуемого API
EMAIL email - e-mail пользователя
STRING url - базовая ссылка, которая будет дополнена параметрами email=<email>&key=<key> и выслана на почту
Ответ:
ERROR error - результат

recover_done:
Производит смену пароля по восстановлению.

STRING api - версия реализуемого API
EMAIL email - e-mail пользователя
OPT GUID key - ключ восстановления (при отсутствии старого пароля)
OPT STRING oldpass - старый пароль (при отсутствии key)
STRING newpass - новый пароль
Ответ:
ERROR error - результат

register:
Производит регистрацию пользователя в системе.

STRING api - версия реализуемого API
EMAIL email - e-mail пользователя
STRING user - логин пользователя
STRING pass - пароль пользователя
STRING name - имя пользователя
STRING surname - фамилия пользователя
STRING middlename - отчество пользователя
STRING studid - номер зачётной книжки (отсутствует для преподавателя)
STRING url - базовая ссылка, которая будет дополнена параметрами email=<email>&key=<key> и выслана на почту
Ответ:
ERROR error - результат

register_done:
Производит подтверждение регистрации пользователя в системе.

STRING api - версия реализуемого API
EMAIL email - e-mail пользователя
GUID key - ключ регистрации
Ответ:
ERROR error - результат

4.2 Оценка качества кода

В качестве количественных показателей корректности (правильности, безошибочности) программных средств (ПС) естественно использовать ту или иную меру близости рассматриваемого ПС к идеальному, т.е. абсолютно корректному, абсолютно правильному, не содержащего ошибок.

При введении показателя r̄ исходим из того очевидного положения, что число ошибок r, допущенных программистами при написании программы с заданными характеристиками (объём, сложность, разветвлённость и т.п.), является величиной случайной. Будучи по определению величиной дискретной, число ошибок в ПС задаётся распределением вероятностей

$$P_i = Bep\{r = i\},\ i = 0,1,2 \qquad (1)$$

Тогда в соответствии с определением

$$K_{00} = Bep\ \{\ r = 0\} = P_0 = 1 - \sum_{i=1}^{\infty} P_i$$

Если предположить, что число ошибок в программе распределено по закону Пуассона, т.е.

$$P_i = \frac{\bar{r}^i}{i!} * e^{-\bar{r}}, \qquad (2)$$

где r̄. — среднее число ошибок в ПС;

i — частота возникновения ошибок,

то, как видно из первой формулы, задание математического ожидания r̄ полностью и однозначно определяет это распределение; в частности, определяет и значение показателя:

$$K_{00}: K_{00} = e^{-\bar{r}} \qquad (3)$$

Таким образом, при принятии указанного предположения показатели r̄ и K_{00} оказываются связанными жёсткой функциональной зависимостью.

Также существует модель Миллса, в соответствии с которой можно искусственно ввести некоторое количество ошибок в код:

$$N = \frac{S * n}{v}, \qquad (4)$$

где n - первоначальное количество ошибок в программе,

S - количество искусственно внесенных ошибок,

n- число найденных собственных ошибок,

V - число обнаруженных к моменту оценки искусственных ошибок.

Типичная плотность ошибок в проекте на число строк кода в соответствии с [9] приведена ниже:

Размер проекта (число строк кода)	Типичная плотность ошибок
Менее 2K	**0 – 25** ошибок на 1000 строк кода
2K – 16K	**0 – 40** ошибок на 1000 строк кода
16K – 64K	**0.5 – 50** ошибок на 1000 строк кода
64K – 512K	**2 – 70** ошибок на 1000 строк кода
512K и более	**4 - 100** ошибок на 1000 строк кода

ПРИМЕР. Расчёт для созданного программного кода

Введём искусственно 3 ошибки. Возьмём среднее число ошибок из первой строки таблицы диапазона ошибок 1. Далее возьмём среднее количество ошибок, обнаруженных моменту оценки искусственных ошибок:

$$N = (3*2)/2; N = 3$$

Подставляем это значение в формулу (2) и получаем $\zeta = (3^5)/5!*2,8^{-5}$, далее получаем $\zeta=0,0117$.

Если по формуле посчитать количество моделируемых строк, то получим: K=z*l, где z - количество реальных строк, 6.84 - коэффициент сложности языка.

Подставим значения и рассчитаем K=1208*6.84. K=8262.72.

Рассчитаем реальное оценочное количество ошибок:

$$Q=w*e,$$

где w — количество ошибок, e — коэффициент критичности ошибок:

$$Q=7*3= 21.$$

Далее при подстановке в формулу (1) имеем

$$K_{00} = 1-\sum_{i=1}^{\infty} \zeta\ i\ = 0,754$$

Таким образом, смоделированное количество ошибок не превышает допустимую пороговую величину и не представляет риска для функционирования разработанного программного обеспечения.

4.3 Разработка приложения в облаке

В настоящее время, наблюдается тенденция увеличения объёмов перевозок, осуществляемых при помощи воздушных судов. Этому способствуют повышение эффективности современных воздушных судов и совершенствование их технических характеристик, большая стоимость перевозок, осуществляемых автомобильным, водным и железнодорожным транспортом.

Согласно Международной ассоциации воздушного транспорта, более 50 процентов издержек авиакомпаний составляют расходы на наземное обслуживание воздушных судов. Снижение данных издержек позволяет повысить прибыль авиакомпаний и аэропорта. Для снижения издержек необходимо уменьшить время оборота – время, в течение которого воздушное судно находится на земле между моментами прилёта и последующего вылета. Уменьшение времени оборота позволяет обслуживать большее количество рейсов.

Для современных компаний данные лежат в основе принятия правильных решений. Например, чтобы рассчитать премии и резервы для страховых полисов или спрогнозировать финансовый риск для инвестиций, эти компании должны математически оценить вероятность и неопределенность, работу, которая зависит от больших объемов активов, обязательств и данных сценария, чтобы назвать несколько требований.

Сотрудники компаний могут выполнять крупные, сложные задания по требованию и легко обрабатывать, получать доступ и манипулировать огромными объемами данных по всему бизнесу компании в отчетные периоды.

Для компаний и предприятий существует полностью управляемая полнофункциональная служба аналитики с открытым кодом Azure HDInsight, которая позволяет быстро обрабатывать большие объемы данных: поддерживает извлечение, преобразование и загрузку, хранение данных, машинное обучение, Интернет вещей и множество других сценариев. служба хранилища Azure предусматривает два уровня для хранилища BLOB-объектов (хранилища объектов), чтобы данные можно было хранить наиболее эффективно в зависимости от их использования. «Горячий» уровень хранилища Azure оптимизирован для хранения часто используемых данных. «Холодный» уровень хранилища Azure оптимизирован для хранения данных, которые используют редко и долго хранят.

Также для разработки ПО используются платформы с открытым кодом, такие как Hadoop, Spark, Hive, LLAP, Kafka, Storm, R и другие.

Пошаговая схема обучения Azure HDInsight доступна по адресу https://docs.microsoft.com/en-us/azure/hdinsight/kafka/apache-kafka-get-started.

4.4. EDL-программа планирования стратегии агента

В настоящее время существует ряд методик прогнозирования, использующих статистические модели и анализ временных рядов. Для систем, позволяющих автоматизировать процедуру планирования ресурсов аэропорта, занятых в наземном обслуживании воздушных судов (ВС), процесс прогнозирования должен базироваться на элементах системы искусственного интеллекта. Очевидно, что новая концепция интеллектуальных информационных технологий, ориентированная на совместное использование моделей и методов естественного и искусственного интеллекта для виртуальных исследований и прогнозирования состояния и поведения активных систем, к которым относятся информационные системы гражданской авиации, в значительной степени будет определять изменение состояния и показатели поведения системы на очередном шаге. Поэтому, для повышения качества прогнозирования, следует использовать мультиагентный подход, при использовании которого каждому звену, участвующему в технологическом процессе наземного обслуживания ВС, назначается программный агент.

В соответствии с [3] мультиагентная система (МАС) – это система, образованная несколькими взаимодействующими интеллектуальными агентами с реактивной тактикой. Рассматриваемая в курсовой работе система планирования ресурсов аэропорта предназначена для составления и оптимизации плана заданий по обслуживанию воздушных судов [12]. Входными данными для данной системы является информация о расписании рейсов, получаемая от авиакомпаний, базирующихся на территории аэропорта,

а также информация от сотрудников, обслуживающих ВС в соответствии с регламентом ТО, поступающая в режиме реального времени.

Использование мультиагентной системы позволяет добиться описания динамики поведения агентов (набора составляющих системы планирования ресурсов) в сложных ситуациях взаимодействия с другими активными элементами и внешней средой. Поведение агента описывается как некоторая итерационная процедура переработки данных о состоянии других агентов и среды с выбором стратегии целенаправленных действий и представляется последовательностью операций в дискретные временные периоды – временные события. Каждой операции соответствует свой модуль, обеспечивающий восприятие информации и накопление знаний об окружающей среде и среде взаимодействия или конфликта, анализ собственного состояния и состояния контрагентов с выбором или коррекцией целевых функций, механизм взаимодействия и обработки данных от контрагентов [11].

В целом, поведение агента можно представить некоторой рекурсивной формой, описывающей нахождение и выбор на очередном шаге функции перехода от исходного состояния к новому состоянию в направлении улучшения целевой функции. В зависимости от концепции, выбранной для организации МАС, обычно выделяются три базовых класса архитектур: архитектуры, которые базируются на принципах и методах работы со знаниями (deliberative agent architectures); архитектуры, основанные на поведенческих моделях типа «стимул-реакция» (reactive agent architectures); гибридные архитектуры (hybrid architectures). Основное отличие данных архитектур заключается в том, что в делиберативной архитектуре применяются интеллектуальные агенты, поведение которых описывается убеждениями, желаниями и знаниями об окружающем мире. При большом количестве таких агентов возникает перегруженность вычислительных ресурсов программной системы. Поведение реактивных агентов определяется таблицей множества вариантов поведения ("модулей поведения"), решающих некоторое множество задач. Каждый из вариантов поведения моделируется конечным автоматом, входами которого являются состояния внешней среды, воспринимаемые агентом, а выходами являются реакции агента. Обычно варианты поведения представляются в виде многоуровневой структуры, в которой верхним уровням соответствуют более высокие уровни абстракции (обобщения) поведения. Верхние уровни модели ответственны за выбор траекторий перемещения, шаблона поведения при выполнении некоторых конкретных действий, которые должны быть скоординированы с действиями других агентов [3].

Реактивные агенты не используют историю своих предшествующих действий для корректировки поведения. Это значительно упрощает реализацию логики поведения таких агентов. Тем не менее, несмотря на простоту реализации, реактивные агенты применяются в решении достаточно сложных задач. В системах с реактивной архитектурой, могут функционировать от 100 до 100000 одновременно взаимодействующих агентов.

Агентам для решения задач распределения ресурсов соответствуют сущности, выступающие в реальном мире потребителями и/или поставщиками ресурсов, а также сами ресурсы. Планирование таких ресурсов, как ТО ВС, можно разделить на следующие этапы: получение задания; поиск ресурсов; выделение ресурсов. Для решения задачи распределения ресурсов агенты вынуждены взаимодействовать и вступать в переговоры друг с другом, которые моделируются коммуникациями. В большинстве случаев задача распределения ресурсов имеет глобальный критерий эффективности, который необходимо достичь путем соответствующего распределения ресурсов [10]. Тогда поведение агентов можно считать кооперативным, поскольку только учет взаимных интересов позволяет решить задачу, и основано на разработанной математической модели. В то же время каждый агент может иметь свои локальные критерии, определяющие эффективность полученного решения для конкретного агента. Например, одно решение может быть более предпочтительным по сравнению с другим на основе использования локальных критериев эффективности, несмотря на то, что они могут быть абсолютно одинаковы с точки зрения общего критерия эффективности. Локальным критерием при решении задачи планирования ресурсов являются минимизация стоимости выполнения задания при использовании одной единицы ресурса, а также возможность использования этого ресурса во временные рамки, обозначенные параметрами задания.

Мультиагентные технологии позволяют успешно решать задачи управления ресурсами с учетом как глобальных, так и локальных критериев [11]. Агентно-ориентированная модель представляется в виде компонентов, описывающих агентов-исполнителей, агента-координатора и агента-оператора, участвующих в процессе планирования и выделения ресурсов для выполнения заданий (рис.4.3).

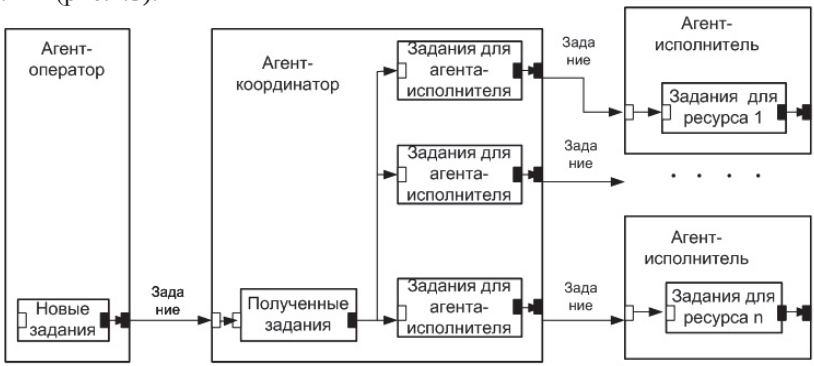

Рисунок 4.3- Агентно-ориентированная модель

Каждому из перечисленных типов агентов присваивается уникальный идентификатор. В данной структуре, взаимодействие агентов происходит через агента-координатора, который является основным агентом. Агенты используют

сообщения для взаимодействия друг с другом. Сообщения также имеют идентификатор, состоящий из идентификатора агента, и порядкового номера последнего сообщения, отправленного данным агентом. Сообщения могут содержать информацию о заданиях, распределяемых агентом-координатором.

Целевая функция направлена на минимизацию времени реакции системы на происходящие события и временных задержек:

$$F(M) = D \cdot (f(a_c) \cdot \sum_{i=1}^{n} f(a_i) + f(a_o) \cdot f(a_r)),\ F(M) \to \min \qquad (5)$$

где M - рассматриваемая мультиагентная система, $f(a_c)$ - целевая функция агента-координатора, $f(a_o)$ -целевая функция агента-оператора, $f(a_r)$ - целевая функция агента-оператора, $f(a_i)$- целевая функция i-го агента-исполнителя, D - среднее значение задержки на передачу одного сообщения между агентами, n - общее количество агентов-исполнителей в мультиагентной системе.

Целевую функцию можно сформулировать в виде задачи минимизации совокупных временных задержек при обработке сообщений, распределении заданий между агентами, а также количества ожидающих обработки сообщений.

Агент-координатор является главным агентом в предложенной мультиагентной системе. Исходя из названия, этот агент координирует действия других агентов посредством отправки управляющих сообщений.

Агент-координатор имеет пул агентов – массив ссылок на существующих агентов-исполнителей.

Целевая функция агента-координатора может быть представлена в виде:

$$f(x) = M \sum_{i=1}^{m} \sum_{j=1}^{n} \Delta t_j D(a_i) \to \min \qquad (6)$$

где a_i – i-й агент-исполнитель, Δt_j – время на обработку j-го задания, $D(a_i)$ – время обращения к i-му агенту-исполнителю, M – общее количество сообщений в буфере агента-координатора, m – количество агентов-координаторов, n – количество сформированных заданий.

Агент-регистратор связывает программный агент с конкретной единицей спецтехники. Агент-регистратор взаимодействует с агентом-координатором. Взаимодействие между агентом-регистратором и агентом-координатором одностороннее, агент-регистратор не обрабатывает сообщения от других агентов.

Целевая функция данного агента, представленная в формуле (7), выражает значение временной задержки, возникающей при формировании нового агента-исполнителя

$$f(a) = \sum(t_c - t_i) \to \min\ , \qquad (7)$$

где t_c – момент времени в конце формирования агента, t_i – момент времени, в который было принято соединение от внешнего устройства.

Агент-исполнитель непосредственно связан с водителем-оператором спецтехники, получает запросы на добавление задания от агента-координатора. Агент-исполнитель проверяет имеющийся список заданий, определяет возможность пересечения с имеющимися заданиями, принимает или отвергает

запрос агента-координатора. После принятия запроса, агент-исполнитель добавляет полученное задание в свой список заданий.

Целевая функция агента-исполнителя может быть представлена как задача минимизации временных промежутков между назначенными заданиями и максимизации показателя загруженности агента. Под показателем загруженности агента-исполнителя понимается отношение суммы временных промежутков между заданиями к полному времени функционирования агента. Целевая функция может быть представлена следующим образом:

$$f(a) = \min\left(\frac{1}{L}\sum_{i=0}^{n}(st(t_i) - fin(t_{i-1}))\right), 0 \le L \le 1 , \qquad (8)$$

где L – показатель загруженности агента, t_i – i-е задание агента-исполнителя, $st(t_i)$ – время начала i-го задания, $fin(t_{i-1})$ – время завершения (i - 1)-го задания, n – общее количество заданий агента-исполнителя.

Агент-оператор взаимодействует с агентом-координатором. База знаний агента-регистратора содержит информацию о типах сообщений и ссылку на агента-координатора мультиагентной системы. Агент-оператор является промежуточным звеном для агента-координатора. Агент-оператор может формировать задания и отправлять сформированные задания агенту-координатору. Агент-координатор, в свою очередь, может отправить сообщение со списком всех выполняемых заданий.

Целевая функция агента-оператора, может быть сформулирована в виде задачи минимизации времени для формирования новых заданий:

$$f(a) = D_s(t_c - t_i) \rightarrow \min , \qquad (9)$$

где D_s – время, затрачиваемое на преобразование строковых данных в целочисленный тип, t_c – момент времени, в который задание сформировано, t_i – момент времени, в который был получен сигнал для формирования задания.

Спроектированная система может быть смоделирована в программном пакете AnyLogic 7, где агенты представлены как экземпляры классов. Агент может рассматриваться как программный модуль, который способен обновлять структуру данных и даже знаний, посредством которых он выполняет свои функции и проявляет свою активность. Он является единицей диспетчеризации и потребления ресурсов системы. Каждый последовательный процесс, выполнение которого, возможно, параллельно с другим процессом, может быть описан как задача, процедура, агент или блок, имеющие некоторый специальный признак, который указывает, что данная часть программы может выполняться одновременно с некоторыми другими частями этой программы.

Каждый агент мультиагентной системы должен исполняться параллельно с остальными агентами. В структуре каждого из агентов также имеется механизм обработки сообщений, получаемых от других агентов. Кроме того, каждый агент может иметь ресурсы, которыми он владеет монопольно, и ресурсы, которые он разделяет с другими процессами. При параллельном выполнении потоков возникает проблема в соревновании за ресурсы. При одновременном доступе к разделяемым ресурсам один из потоков может изменить значение ресурса до завершения использования этого ресурса другим

потоком. Такой сценарий может привести к искажению результатов вычислений.

Обмен сообщениями является основным механизмом передачи информации между агентами в распределенных системах. Возможны два режима обмена сообщениями: синхронный - в этом режиме при посылке сообщения агент-отправитель прекращает выполнение программы до получения ответа от агента-адресата. Такой режим снижает параллелизм работы системы; асинхронный - в этом режиме каждый активный агент располагает очередью сообщений. Когда сообщение посылается, агент-отправитель записывает информацию об отосланном сообщении в очередь ожидаемых ответов, продолжая выполнять программу (не ожидая, пока ответ будет получен). Такой режим не снижает параллелизм работы. Будем использовать второй подход. Кроме того, каждый агент включает в себя: уникальный идентификатор агента строкового типа; массив сообщений, который содержит сообщения, полученные от других агентов, и пригодные для обработки; целочисленное значение порядкового номера последнего отправленного этим агентом сообщения, использующееся для формирования идентификатора сообщения.

Внешняя среда модели представлена в виде агента, в составе которого имеется пространство агентов, состоящее из агента-координатора, агента-регистратора, агента-оператора и пяти агентов-исполнителей. Все агенты имеют соединение с портом агента-координатора. В этом агенте могут обитать одиночные агенты, либо популяции агентов. Под популяцией агентов понимается множество агентов, обращение к которым производится по индексу, аналогично массивам.

Обработка заданий заключается в ожидании сигнала о завершении задания. После получения сигнала, в переменную хранения текущего задания записывается первый элемент из коллекции заданий. Первый элемент коллекции заданий затем удаляется. После чего агент-регистратор формирует нового агента-исполнителя, инициализирует значения идентификатора агента. Агент-регистратор проверяет полученное сообщение, определяя тип ВС, которая будет привязана к агенту-исполнителю. Полученное значение инициализирует соответствующий параметр агента-исполнителя. Ссылка на сформированного таким образом агента-исполнителя помещается в объект типа Message, с последующей отправкой агенту-координатору.

Исходя из целевой функции, можно рассчитать максимальное время реакции системы на происходящее событие:

$$\Delta T_{max} = \sum_{i=1}^{5} \Delta t_i^v + \Delta t_i^o \qquad (10),$$

где Δt_i^v – время реакции i-го типа агента на сообщение, Δt_i^o – время реакции i-го типа агента на внешнее воздействие. Стоит отметить, что полученный показатель времени реакции системы не учитывают временные задержки, возникающие при сетевом взаимодействии с внешними устройствами.

4.5 Управление процессами обработки неструктурированных данных

Появление новых и совершенствование имеющихся информационных технологий связано с необходимостью в минимальные сроки обрабатывать большие объемы информации различного рода, получившие название Большие Данные. Под термином «Большие данные» или "Big Data" подразумевает под собой объемы, исчисляемые не мегабайтами и не гигабайтами информации, а еще большими цифрами. Большая часть эксплуатируемых программных и аппаратных средств не предназначена для эффективной работы в таких условиях. Появление Больших Данных привело к тому, что разработчики ПО и пользователи вынуждены решать проблемы быстрой обработки поступающей информации и надежного ее хранения. Как отмечают эксперты, проблемы, связанные с хранением и обработкой больших массивов информации – только часть всех проблем, требующих скорейшего решения. Всего же выделяется пять основных проблем:

- Volume – объем данных: относится к наборам данных, размер которых выходит за пределы возможностей программных средств типичной базы данных собирать, хранить, обрабатывать и анализировать данные;
- Velocity – пользователь не должен замечать снижение скорости работы приложения при увеличении объема обрабатываемых данных реакция на текущую информацию за время, ограниченное приложением; потоковая обработка (например, GPS данных в реальном времени);
- Variety – информация может быть разного содержания и с разной структурой: способность обработки множества типов, источников и форматов данных от сенсоров, умных устройств, социальных сетей; интеграция большого числа источников, содержащих различные данные (структурированные наряду с сырыми, слабоструктурированными, неструктурированными данными, извлекаемыми из web- страниц, weblog файлов, e-mail, документов и др.);
- Veracity – пользователь должен быть уверен в том, что предоставляемые данные полностью достоверны;
- Value – полученная информация должна представлять ценность для ее обладателя.

IBM Info Sphere Biglnsights представляет собой программную платформу, предназначенную для обнаружения и обработки больших объемов разнотипной информации (которая игнорировалась из-за трудности ее сбора и обработки традиционными методами) и формирования на ее основе данных, необходимых для выработки оптимальных и сбалансированных бизнес-решений.

Более подробная информация доступна по адресу: http:// www.ibm.com/develpperworks/data/library/techarticle/dm-1110biginsightsintro.

ЛИТЕРАТУРА

1. Новые методы работы с большими данными: победные стратегии управления в бизнес-аналитике: Научно-практический сборник. Под ред. А.В. Шмида.- М.: ПАЛЬМИР, 2016.- 528 с.
2. Биберштейн Н., Боуз С., Джонс К и др. Компас в мире SOA: ценность для бизнеса, планирование и план развития предприятия/Пер. с анг.-М.: КУДИЦ-ПРЕСС, 2007.
3. Ю.А.Ивашкин. Агентные технологии и мультиагентное моделирование систем: учеб. пособие / Ю.А. Ивашкин.- МФТИ, 2014.-268с.
4. Гольдштейн А. Мультисервисные сети.- СПб. – Питер, 2001, 680 с.
5. Джамса К., Коуп К. Программирование для Internet в среде Windows.- СПб.: Питер, 1996.-688 с.
6. Сахил Малик Microsoft ADO.NET 2.0 для профессионалов -Pro ADO.NET 2.0. — М.: «Вильямс», 2006. — С. 560
7. Сидни Фейт TCP/IP: Архитектура, протоколы, реализация. – М.: ЛОРИ, 2000 – 756 с.
8. Мазуркевич А., Еловой Д. PHP: настольная книга программиста - Мн.: Новое знание, 2003. - 480 с.
9. Липаев В.В. Программная инженерия сложных заказных программных продук- тов: Учебное пособие. – М.: МАКС Пресс, 2014. – 312 с.
10. Ландсберг С.Е. Особенности построения информационных систем с использованием мультиагентных технологий /С.Е. Ландсберг, А.А. Хованских //Вестник Воронежского государственного университета-Выпуск №3-1, том 10/201. Режим доступа: http://cyberleninka.ru/article/n/osobennosti-postroeniya-informatsionnyh-sistem-s-ispolzovaniem-multiagentnyh-tehnologiy (дата обращения - 01 июля 2017)
11. Ризванов Д.А. Алгоритмы управления ресурсами в сложных системах с применением мультиагентных технологий. // Вестник УГАТУ. – 2013. - № 5. - 8 с.
12. Смирнов Н.Н., Чинючин Ю.М. Основы поддержания летной годности ВС: Учебное пособие. – М.: МГТУ ГА, 2012. – 100 с.
13. Классификация агентов. //Портал искусственного интеллекта: Режим доступа:http://www.aiportal.ru/articles/multiagent-systems/agent-classification.html (дата обращения 01 ноября 2017).
14. https://www.osp.ru – Электронный журнал Открытые системы.
15. http://microsoft.com/php
16. http:// www.microsoft.com/net

СОДЕРЖАНИЕ

ПРИЛОЖЕНИЕ А

МИНИСТЕРСТВО ТРАНСПОРТА РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение высшего образования
МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ГРАЖДАНСКОЙ АВИАЦИИ (МГТУ ГА)

Кафедра вычислительных машин, комплексов, систем и сетей

Курсовая работа
защищена с оценкой

_____

_____
(подпись преподавателя, дата)

КУРСОВАЯ РАБОТА
по дисциплине Интернет-технологии
Вариант №1

Тема: Web-приложение контроля посещения учебных занятий

Выполнил ст. группы ЭВМ 4-1
Иванов Иван Иванович
(Ф.И.О.)
Руководитель:
доцент, к. т. н., Романчева Н.И
(звание, степень Ф.И.О.)

Курсовая работа
допущена к защите

_____

(подпись преподавателя, дата)

МОСКВА - 20___

ПРИЛОЖЕНИЕ Б

МИНИСТЕРСТВО ТРАНСПОРТА РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение высшего образования
МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ГРАЖДАНСКОЙ АВИАЦИИ (МГТУ ГА)

Кафедра вычислительных машин, комплексов, систем и сетей

З А Д А Н И Е

на выполнение курсовой работы
по дисциплине
Интернет-технологии
вариант № 1
Web-приложение контроля посещения учебных занятий

Исходные данные:

Разработать программный комплекс контроля успеваемости студентов. Комплекс реализует клиент-серверную архитектуру. На сервере должна располагаться база данных, содержащая данные на основе учебного плана (название дисциплин, зачетные единицы и др.), списки студенческих групп, списки преподавателей.

Программный комплекс включает клиентское приложение.

Данное приложение должно обеспечивать следующие возможности: создание аккаунта конкретного преподавателя и студента, создание и планирование дисциплины преподавателем, формирование списка студентов в дисциплине, заполнение посещаемости и успеваемости занятий студентами, просмотр промежуточных и итоговых результатов успеваемости через отправку на e-mail адрес.

Разработать универсальный интерфейс взаимодействия клиентских и серверных компонентов; использовать рхитектурные приёмы, гарантирующие расширяемость и поддерживаемость программного комплекса.

Задание выдано « »_____20____ г. (подпись преподавателя)

Задание получил _____ (подпись студента)

Приложение В

Приложение А
Спецификация

Обозначения	Наименование	Примечание
	Документация	
КРномерзачетной 12	Web-приложение	
	контроля посещения	
	учебных занятий. Текст	
	программы.	
КР171825 34	Web-приложение	
	контроля посещения	
	учебных занятий.	
	Руководство оператора.	

Приложение Г

Приложение Б

УТВЕРЖДЕН
КРномерзачетной книжки 12

Web-приложение контроля посещения учебных занятий

Текст программы
КРномерзачетной книжки 12
Листов 12

20___

Приложение Д

Приложение В

УТВЕРЖДЕН
КРномер зачетной книжки 34

Web-приложение контроля посещения учебных занятий

Руководство оператора
КРномерзачетной книжки 34
Листов 18

20___