МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н.Г. ЧЕРНЫШЕВСКОГО»

Кафедра математического и компьютерного моделирования

Цифровой ВУЗ. РП и ФОС

АВТОРЕФЕРАТ МАГИСТЕРСКОЙ РАБОТЫ

студентки 2 курса 247 группы

направление 09.04.03 – Прикладная информатика

механико-математического факультета

Жиц Марии Григорьевны

Научный руководитель
доцент, к.ф.-м.н., доцент А.А. Орёл

Зав. кафедрой
зав.каф., д. ф. – м. н., доцент Ю.А. Блинков

Саратов 2019

В введении магистерской работы «Цифровой ВУЗ. РП и ФОС» рассматриваются вопросы, связанные с проектированием модуля информационной системы «Цифровой ВУЗ».

Актуальность темы обусловлена тем, что в современных условиях всеобщая информатизация процессов практически во всех сферах жизни общества не могла не затронуть и сферу образования. Важным блоком в этой области является цифрализация процессов разработки и функционирования документации, регламентирующей учебную деятельность на стадии разработки и дальнейшей поддержки методического обеспечения учебного процесса.

Цель работы - проектирование и реализация инструментария для первичной разработки рабочих программ учебных дисциплин на основе федеральных государственных образовательных стандартов высшего образования (ФГОС ВО), а также их возможной (при необходимости) корректировки в процессе дальнейшего использования.

Для достижения данной цели в ходе исследования должны были решаться следующие задачи:

- изучение документации, регламентирующей разработку и реализацию учебного процесса в организациях высшего образования в Российской Федерации;
- изучение существующего инструментария для проектирования и реализации модулей в информационных системах, сбор и анализ информации об их практическом применении, сравнение существующих инструментов с целью выбора оптимальных для решения поставленной задачи;
- проектирование и реализация модуля «Рабочая программа».

Объекты исследования - ФГОС ВО (3++) по направлению подготовки 09.04.03 и иная учебная документация ФГБОУ ВО СГУ имени Н.Г. Чернышевского. Предмет исследования - программное обеспечение для реализации модулей информационных систем, связанных с разработкой текстово-цифровой и графической документации с возможностью внесения впоследствии необходимых изменений.

Магистерская работа состоит из введения, трех глав, заключения, списка использованных источников и одного приложения.

Основное содержание работы

В первой главе «Теоретические основы разработки рабочих программ. Постановка задачи» рассматриваются базовые документы, на основании которых должны разрабатываться рабочие программы учебных дисциплин и их основные элементы, учитываемые в ходе проектирования модуля. Особое внимание уделено требованиям федеральных государственных образовательных стандартов высшего образования (ФГОС ВО) по направлениям подготовки. Конкретное содержание ООП определяется образовательной организацией самостоятельно на основании этих требований, с учетом соответствующей примерной ООП, включенной в специальный реестр. Далее на основе этой программы разрабатывается учебный план, жестко определяющий перечень всех учебных дисциплин с указанием общего количества учебных часов, отведенных на освоение каждой учебной дисциплины, с распределением по видам учебной деятельности, включая самостоятельную работу студента.

На основании ФГОС ВО и утвержденного учебного плана по соответствующему направлению подготовки разрабатываются рабочие программы для каждой учебной дисциплины.

ФГОС ВО (3++) содержит исчерпывающий перечень УК - универсальных, не зависящих от профиля будущей профессиональной деятельности компетенций и ОПК - общепрофессиональных компетенций, формируемых на основе профессиональных стандартов, соответствующих профессиональной деятельности выпускников, а в случае отсутствия таких стандартов - на основе анализа требований, предъявляемых к выпускникам на рынке труда. Каждый обучающийся в ходе освоения соответствующей ООП должен приобрести все эти УК и ОПК. Следовательно, рабочая программа любой учебной дисциплины должна отражать одну или несколько конкретных УК и ОПК (в полном соответствии с формулировками ФГОС ВО), которые формируются у студентов при изучении данной дисциплины.

Также из ФГОС ВО в рабочую программу должны быть внесены требования к методическому и материально-техническому обеспечению учебного процесса, в том числе при обучении инвалидов и лиц с ограниченными

возможностями здоровья. Значительная часть этих требований может быть унифицирована и вноситься во все (или многие) РП автоматически.

Из учебных планов, разработанных на базе ФГОС ВО, в рабочую программу переносятся данные о количестве учебных часов по видам деятельности и формам промежуточной аттестации обучающихся.

Дальнейшая разработка РП проводится авторами самостоятельно по всем предусмотренным структурным блокам:

1. Цели освоения дисциплины.
2. Место дисциплины в структуре ООП.
3. Компетенции обучающегося, формируемые в результате освоения дисциплины.
4. Структура и содержание дисциплины.
5. Образовательные технологии.
6. Учебно-методическое обеспечение самостоятельной работы студентов. Оценочные средства для текущего контроля успеваемости, промежуточной аттестации по итогам освоения дисциплины.
7. Данные для учета успеваемости студентов в БАРС.
8. Учебно-методическое и информационное обеспечение дисциплины (модуля).
9. Материально-техническое обеспечение дисциплины.

Рабочая программа, разработанная в соответствии с изложенными требованиями, одобренная и утвержденная в установленном порядке, допускается к использованию в учебном процессе.

Вторая глава «Проектирование модуля» состоит из двух разделов.

В первом разделе «Выбор инструментов реализации» выделено три подраздела, в которых подробно рассматривается существующий инструментарий программирования и обосновывается сделанный автором выбор инструментов.

В подразделе 2.1.1 «*База данных*» приведены основания выбор системы управления базами данных (СУБД).

Особое значение при разработке информационной системы придается оптимальному выбору системы управления базами данных (СУБД). Изначаль-

но предстояло сделать выбор между типами СУБД - реляционными и нереляционными.

Реляционные СУБД представляют собой набор таблиц с четкой структурой, которые позволяют хранить достаточно большой объем информации. Базы данных можно легко масштабировать. При этом они весьма устойчивы и просты в понимании. Однако, в настоящее время приложения все чаще имеют сложную структуру данных, реализация которых с помощью реляционных СУБД превращается в очень трудоемкий процесс. Поэтому предпочтение при проектировании модуля было отдано нереляционным СУБД , основанным на хранении документов.

Практически все документно-ориентированные СУБД основаны на технологии NoSQL. Сам термин NoSQL обозначает целый ряд подходов, проектов, направленных на реализацию моделей баз данных, имеющих существенные отличия от используемых реляционных СУБД с доступом к данным средствами языка SQL. NoSQL - важный и полезный инструмент, но он не может считаться универсальным. Основная цель подхода - расширить возможности БД там, где SQL недостаточно гибок, и не вытеснять его там, где он справляется со своими задачами.

В основе идеи NoSQL лежит:

- Нереляционная модель данных,
- Открытый исходный код,
- Хорошая горизонтальная масштабируемость (автоматическое распределение между несколькими серверами).

Далее были рассмотрены несколько наиболее популярные NoSQL СУБД.

CouchDB - документно-ориентированная система управления базами данных, не требующая описания схемы данных. Эта программа является свободной, открытой и написана на языке Erlang. CouchDB можно рассматривать как сервер веб-приложений; для реализации этой идеи в нее встроенный производительный веб-сервер, а программный код, как и данные, сохраняется в той же базе данных. Для автоматизации работы с приложениями CouchDB используется утилита CouchApp.

Подобно иным документно-ориентированным СУБД, и в отличие от реляционных СУБД, CouchDB предназначена для работы с полуструктурированной информацией и имеет ряд особенностей:

1. данные сохраняются не в строках и колонках, а в виде JSON-подобных документов, моделью которых является не таблицы, а деревья;
2. не поддерживается типизация элементов данных, то есть сопоставление отдельным полям документов типов integer, DATE и пр.
3. целостность базы данных обеспечивается исключительно на уровне отдельных записей, но не на уровне связей между ними. Связи между таблицами или записями принципиально не поддерживаются, соответственно операция объединения (JOIN) между таблицами не определена. Одновременно может быть запущенно несколько потоков для чтения базы данных и только один - для записи.

Внешний интерфейс (API) к данной СУБД построен на основе архитектуры REST, то есть сама база данных, отдельные записи, отображения и запросы представляют собой ресурсы, которые имеют уникальный адрес (URL) и поддерживают операции GET, PUT, POST, DELETE.

Взаимодействие между отдельными компонентами СУБД, то есть с серверами представлений осуществляется с помощью текстового протокола, а данные передаются в формате JSON.

Hbase — это распределенная, колоночно-ориентированная, мультиверсионная база типа «ключ-значение». Данные организованы в таблицы с проиндексированные первичным ключом, который обозначают как RowKey. Для каждого первичного ключа может храниться неограниченный набор атрибутов (или колонок).Для каждого атрибута может храниться несколько различных версий, имеющих разные временные метки. Записи физически хранятся в отсортированном по первичному ключу порядке.

Apache HBase рассчитана на поддержание высокой производительности при увеличении масштаба до сотен узлов для работы с миллиардами строк и миллионами столбцов. Она использует распределенную файловую систему Hadoop в качестве отказоустойчивого хранилища данных.

В результате проведенного анализа выбор был сделан в пользу СУБД *MongoDB* - документно-ориентированной СУБД с открытым исходным ко-

дом, не требующая описания схемы таблиц. Классифицирована как NoSQL, использует JSON-подобные документы и схему базы данных. MongoDB реализует новый подход к построению баз данных, где нет таблиц, схем, запросов SQL, внешних ключей и многих других элементов, характерных для объектно-реляционных баз данных. Благодаря этому MongoDB работает быстрее, обладает лучшей масштабируемостью, ее легче использовать.

Данная система предоставляет пользователям значительные возможности для решения практически всех задач, возникающих как в ходе разработки модуля, так и при его последующем практическом использовании. Особо отметим следующие возможности СУБД MongoDB.

Во-первых, СУБД MongoDB поддерживает ad-hoc-запросы: они могут возвращать конкретные поля документов и пользовательские JavaScript-функции. Поддерживается поиск по регулярным выражениям. Также можно настроить запрос на возвращение случайного набора результатов.

Во-вторых, в СУБД MongoDB имеется поддержка индексов.

В-третьих, система может работать с набором реплик, то есть содержать две или более копии данных на различных узлах. Каждый экземпляр набора реплик в любой момент может выступить в роли как основной, так и вспомогательной реплики.

Система вполне может быть использована в качестве файлового хранилища с балансировкой нагрузки и репликацией данных. MongoDB поддерживает коллекции с фиксированным размером, сохраняющие порядок вставки и выполняющие функции кольцевого буфера по достижении заданного размера.

Вся система MongoDB может представлять не только одну базу данных, находящуюся на одном физическом сервере. Функциональность MongoDB позволяет расположить несколько баз данных на нескольких физических серверах, и эти базы данных смогут легко обмениваться данными и сохранять целостность.

Главным достоинством СУБД MongoDB по сравнению с реляционными базами данных является хранение не строк, а документов, которые могут хранить сложную по структуре информацию. Сам документ можно представить как хранилище ключей - простых меток, с которыми ассоциируются

определенные фрагмент данных, и значений. Каждому ключу сопоставляется определенное значение.

Исходя из вышеизложенного предпочтение при проектировании и реализации модуля было отдано системе MongoDB, так как она наилучшим образом подходит для решения поставленных задач, наиболее проста в освоении и использовании и не слишком ресурсоемка.

В подразделе 2.1.2 «*Язык программирования*» подробно рассматривается язык программирования Python, являющийся высокоуровневым языком программирования общего назначения. Он ориентирован на повышение производительности разработчика и читаемости кода. Синтаксис ядра Python очень компактен, но при этом стандартная библиотека языка включает широкий объём полезных функций.

Python может поддерживать различные виды программирования, в том числе структурное, объектно-ориентированное, функциональное, императивное и аспектно-ориентированное. Его основные архитектурные черты - возможность динамической типизации, автоматическое управление памятью, наличие механизмов обработки исключений. Поддерживаются многопоточные вычисления, высокоуровневые структуры данных, разбиение программ на модули либо, напротив, их объединение в пакеты.

Немаловажным достоинством языка программирования Python является его расширяемость. Но главное достоинство языка программирования Python - это наличие большого числа подключаемых к программе модулей, которые обеспечивают различные дополнительные возможности. Такие модули пишутся на C или непосредственно на Python и могут быть разработаны любыми достаточно квалифицированными программистами. В качестве примера можно привести библиотеки, реализующие GUI-интерфейс, т.е. пользовательский интерфейс, о которых подробно рассказывается в подразделе 2.1.3.

Единственным недостатком языка программирования Python является сравнительно невысокая скорость выполнения Python-программы, что обусловлено ее интерпретируемостью. Но этот недостаток вполне перекрывается достоинствами языка при написании программ, для которых скорость выполнения не является критичным показателем.

В подразделе 2.1.3. «*Библиотека реализации графического интерфейса*» рассмотрены три наиболее популярные библиотеки: PySide, PyQT и Tkinter.

Графический интерфейс реализуется, как правило, с помощью сторонних библиотек. Самые популярные из них -PySide, PyQt и tkinter.

PySide - привязка языка Python к инструментарию Qt, совместимый на уровне API с PyQt.

Qt представляет собой кросс-платформенный фреймворк для приложений и пользовательских интерфейсов. Он позволяет разработчикам писать приложения однократно и разворачивать их на операционных системах без переписывания исходного кода, что очень удобно при участии в проекте большого числа разработчиков. Таким образом, благодаря объединению мощностей Qt и Python, Qt представляет собой первоклассную платформу для быстрой разработки программного обеспечения на языке Python, доступную на всех основных операционных системах. PySide доступна для свободного использования в открытых, закрытых, а также коммерческих проектах.Исходные коды PySide открыты и доступны.

PyQt — набор «привязок» графическогофреймворка Qt для языка программирования Python, выполненный в виде кросс-платформенного расширения Python. PyQt включает в себя Qt Designer (Qt Creator) — дизайнер графического интерфейса пользователя. Программа pyuic генерирует Python-код из файлов, созданных в Qt Designer. Кроме того, можно добавлять в Qt Designer новые графические элементы управления, написанные на Python.

Qt Designer - кросс-платформенный компоновщик макетов и форм графического интерфейса пользователя. Он позволяет быстро спроектировать виджеты и диалоги, создавая экранные формы с использованием тех же виджетов, на основании которых будет осуществляться работа приложения. Формы, созданные с Qt Designer, являются полностью функциональными, а также могут просматриваться в режиме реального времени.

Для реализации разрабатываемого модуля оптимальной является библиотека tkinter, так как она не требует установки дополнительных пакетов, установки дополнительного программного обеспечения, взаимодействует со всеми версиями языка Python, в том числе с новейшими, относительно менее трудоемка при использовании. Недостатком можно было бы считать относительно

меньшую скорость обработки событий приложения, но в рамках реализации нашего модуля эта скорость не является существенным критерием.

Второй раздел «Разработка графического интерфейса» состоит из двух подразделов.

Перед началом разработки графического интерфейса в подразделе 2.2.1 был рассмотрен один из паттернов проектирования пользовательских интерфейсов - Диаграмма Гутенберга, с помощью которой могут быть созданы оптимальные возможности зрительного восприятия разрабатываемого модуля.

Диаграмма Гутенберга описывает модель поведения пользователя при просмотре информации, выведенной на экран.

Страница условно делится пользователем на 4 зоны:

1. Левая верхняя - зона приоритетного просмотра.В эту зону пользователь всегда смотрит в первую очередь. Поэтому именно здесь размещается самая важная информация — логотип и слоган, либо, если речь идет мы о текстовом контенте, первые 2-3 слова заголовка.
2. Правая верхняя — хорошо просматриваемая зона. Как правило, из зоны приоритетного просмотра взгляд смещается по горизонтали именно сюда. Внимание уже ослаблено, но пользователь все еще достаточно сконцентрирован, поэтому вдоль линии движения взгляда и в самом секторе размещается другая важная информация: контакты, форма обратного звонка, адрес, оффер, а в случае с «голым» текстом - заголовок целиком.
3. Левая нижняя — наименее исследуемая зона. Сюда после просмотра правого верхнего сектора взгляд смещается буквально на доли секунды, поэтому здесь не рекомендуется размещать значимую информацию.
4. Правая нижняя — зона выхода. Здесь пользователь принимает решение о совершении целевого действия - продолжении чтения или уходе с сайта. Именно в этой зоне целесообразно размещать призыв к действию или кнопку заказа.

Далее, в подразделе 2.2.2 рассматриваются три основных вида окон, различающихся по типу вносимой информации: окно, содержащее только текстовую информацию; окно, предполагающее отображение информации из базы

данных; окно, предполагающее построение таблицы. Для каждого вида приведен пример программного кода и результат работы кода.

В третьей главе «Программная реализация» подробно описывается процесс реализации модуля «Рабочая программа».

Рассмотрена реализация Главного окна модуля, в котором отражены все пункты, подлежащие заполнению пользователем в ходе разработки рабочей программы.

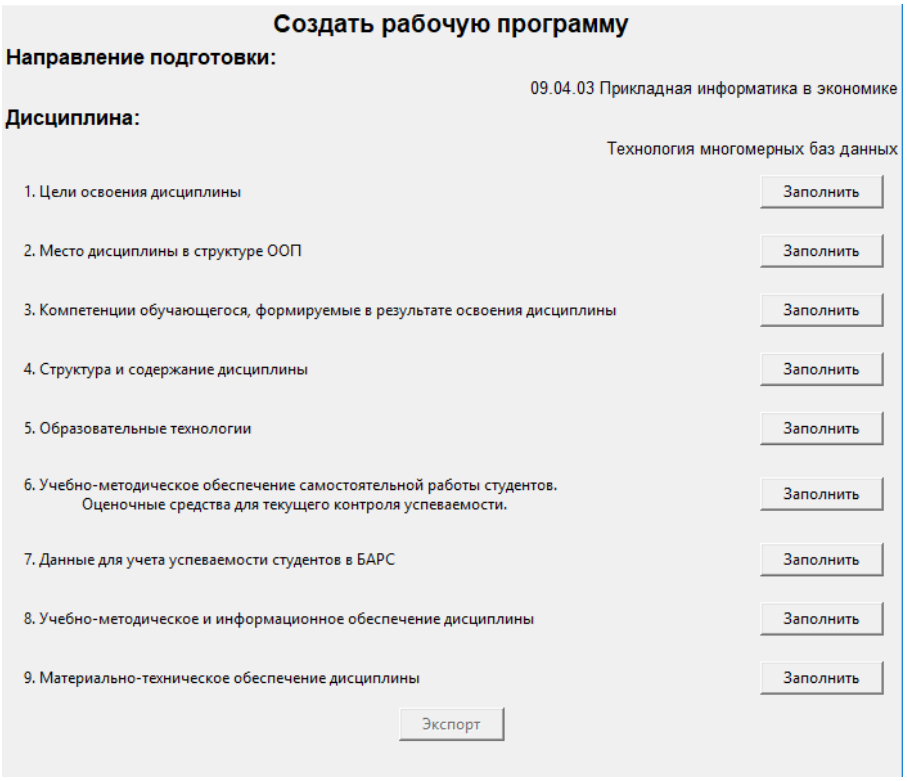

Рисунок 1 — Главное окно модуля

Далее описывается реализация окон, принадлежащих каждому виду, выделенному в подразделе 2.2.2 соответственно, «образовательные технологии» (первый вид), «Учебно-методическое и информационное обеспечение дисциплины (модуля)» (второй вид) и «Компетенции обучающегося, формируемые в результате освоения дисциплины» (третий вид).

Также рассмотрено экспортирование готовой рабочей программы в текстовый документ с помощью отдельного окна «Экспорт».

В заключении говорится о том, что в ходе работы были представлены и решены установленные перед началом исследования задачи, а именно:

- были получены теоретические знания о документации, регламентирующей разработку и реализацию учебного процесса в организациях высшего образования в Российской Федерации,
- изучен существующий инструментарий для проектирования и реализации модулей в информационных системах, собрана и проанализирована информация об их практическом применении проведено сравнение существующих инструментов с целью выбора оптимальных для решения поставленной задачи,
- спроектирован и реализован модуль «Рабочая программа» (изначально поставленная задача разработки отдельного модуля «Фонд оценочных средств» утратила актуальность в связи с изменениями ФГОС ВО, согласно которым ФОС ныне является составной частью Рабочей программы).

Разработанный модуль может быть включен в действующую систему «Цифровой ВУЗ» и взаимодействовать с другими её модулями по желанию пользователя.