Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра автоматизированных систем управления (АСУ)

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ СЕРВЕРНОЙ ЧАСТИ ОБЛАЧНОГО СЕРВИСА ОБУЧЕНИЯ

Отчёт по производственной практике: преддипломной

Студент гр. 434-1
Ю.А. Богомолов

_____ *«_ _» _____ 2018 г.*
Подпись

Руководитель:
Преподаватель каф. АСУ
Доктор технических наук
_____ Профессор
Оценка М.Ю. Катаев

_____ *«___» _____ 2018 г.*
Подпись

Томск 2018

Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра автоматизированных систем управления (АСУ)

ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ НА ПРОИЗВОДСТВЕННУЮ ПРАКТИКУ: ПРЕДДИПЛОМНУЮ ПРАКТИКУ

студенту _____Богомолову Юрию Алексеевичу_____
группа _____434-1_____ факультет ____Систем Управления_____

срок практики с ___30.04.2018___ по _____27.05.2018_____

1. Тема индивидуального задания _____Программное обеспечение серверной части облачного сервиса обучения_____
   _____
2. Перечень вопросов, подлежащих разработке _____
   _____
   _____

Руководитель от университета _____ М.Ю. Катаев
Преподаватель каф. АСУ,
д.т.н., профессов

Задание принял к исполнению _____ Ю.А. Богомолов

«____» _____ 2018 г.

Содержание

Введение

Производственная практика проходила на кафедре. В качестве темы задания была выдана тема «Программное обеспечение серверной части облачного сервиса обучения». Данная работа является продолжением того, что было проделано в рамках проекта ГПО АСУ-1101 «Облачная информационная система обучения студентов». В связи с этим к началу работы уже имелась модель веб-сайта. Поэтому на время практики была поставлена цель начать разработку концепта серверной части системы. Исходя из этого были поставлены задачи продумать концепт серверной части системы и выбрать необходимые инструменты разработки.

1 Описание концепта

1.1 Назначение системы

В настоящее время современные информационные технологии в образовании применяются не в полной мере. Даже в пределах одной страны использование информационных технологий может сильно разниться: где-то активно используют интерактивные доски, а где-то – обычные доски для мела; в некоторых вузах применяются комплексные системы управления обучением, а в других учебных заведениях занимаются только по бумажным книгам.

Поэтому возникла идея разработки системы, целью которой была бы модернизация процесса обучения, будучи при этом простой и удобной в использовании. Удобной как для студентов и преподавателей, так и для целых организаций.

Система, способная достичь указанной цели, должна предоставлять следующие базовые возможности:

1. Для преподавателя:
    а) создавать и размещать учебные материалы;
    б) выдавать учащимся индивидуальные и коллективные задания;
    в) просматривать решения учащихся.
2. Для студента:
    а) просматривать учебные материалы;
    б) решать выданные задания;
    в) отслеживать свои достижения для составления собственного портфолио.
3. Для организации:
    а) отслеживать активность студентов;
    б) предоставлять свои учебные материалы другим организациям.

Однако отличительной особенностью предлагаемой системы, качественно улучшающей вышеописанные функции, является невероятная гибкость, позволяющая автоматизировать многие процессы. Подобная гибкость достигается за счёт возможности разработать собственные приложения, способные функционировать как на серверной, так и на клиентской стороне. Эти приложения могут быть использованы в разработке учебных материалов и других приложений, выдаче и решении заданий, а также проверке решений.

1.2 Структура системы

Перечисленное в прошлом подразделе многообразие функций требует довольно сложной структуры системы, множество компонентов:

1. Веб-сайт – часть системы, непосредственно взаимодействующая с пользователями. Его основное предназначение – обеспечивать удобный интерфейс и доступ ко всем возможностям системы.
2. База данных – хранит структурированные данные системы.
3. Серверное ПО – обеспечивает запуск приложений и их взаимодействие с браузером пользователя.
4. Приложения – производят необходимые обработки данных, позволяя реализовать обозначенные выше гибкость системы и автоматизацию действий.

На рисунке 1.1 изображена схема взаимодействия описанных компонентов системы.

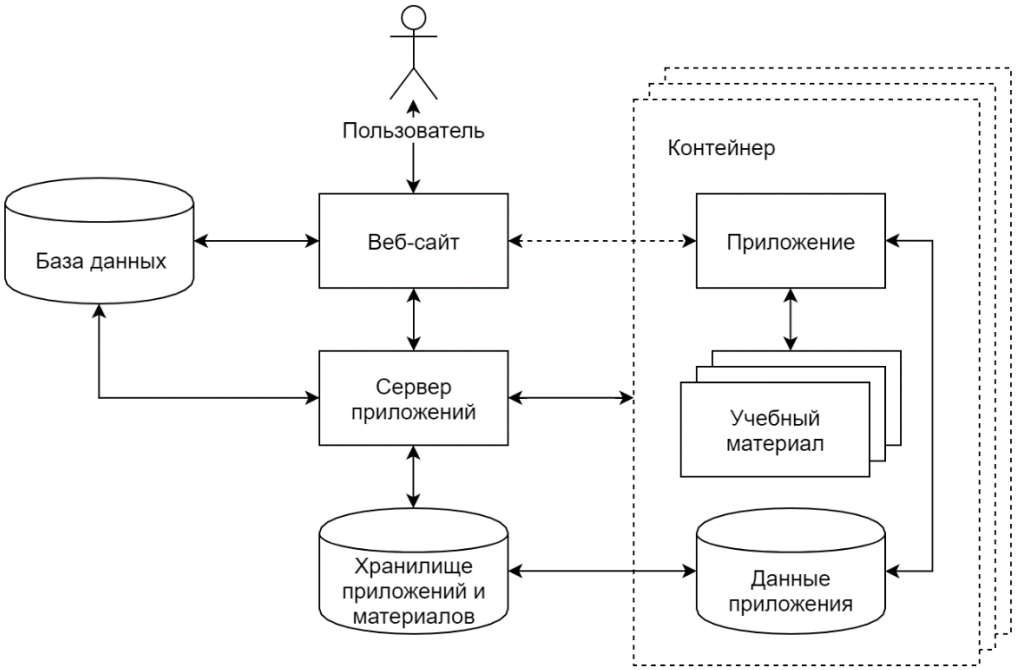

Рисунок 1.1 – Схема взаимодействия компонентов системы

1.3 Примеры приложений

Приложения – очень важный элемент системы. Рисунки 1.2-1.4 демонстрируют примеры различных приложений.

На рисунке 1.2 представлена схема лекционного приложения. Она показывает простейшее взаимодействие различных частей системы приложений.

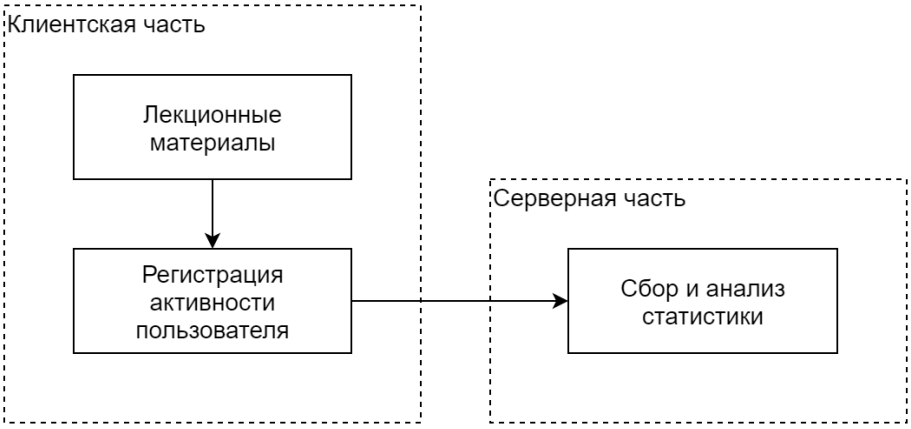

Рисунок 1.2 – Пример лекционного приложения

На рисунке 1.3 представлена схема приложения «Задача вида Дано/Решение». Здесь уже есть несколько приложений, работающий на сервере.

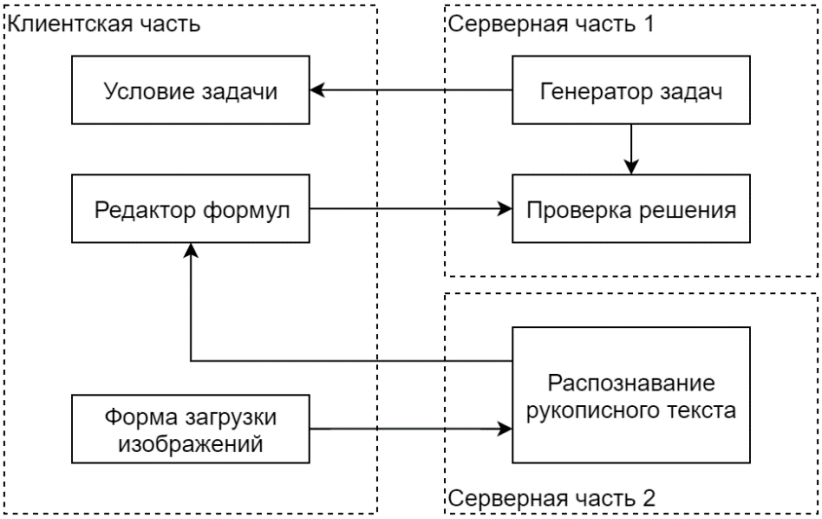

Рисунок 1.3 – Схема приложения «Задача вида Дано/Решение»

На рисунке 1.4 продемонстрирован пример приложения для лабораторных работ по программированию. Кроме того, такое приложение может применяться при проведении олимпиад по спортивному программированию.

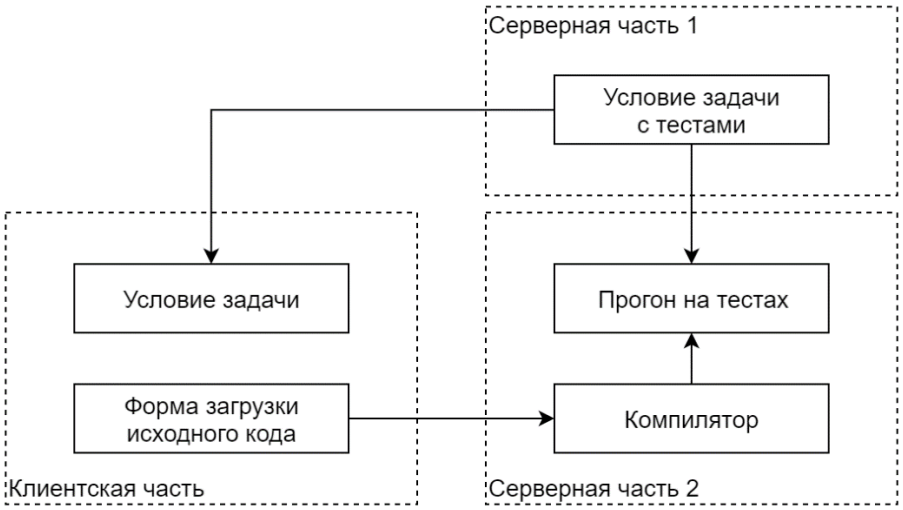

Рисунок 1.4 – Приложение для лабораторных работ по программированию

Как видно на рисунках 1.2-1.4, приложения могут объединяться в целые системы, необходимые для достижения нетривиальных целей. Например, с помощью приложений можно реализовать самостоятельную систему управления обучением с использованием таких популярных стандартов создания электронных курсов, как SCORM, xAPI и CMI-5.

2 Проектирование серверной части системы

2.1 Используемые технологии

В первую очередь было необходимо определиться с используемыми технологиями для реализации серверной части. Во-первых, возникла необходимость решить проблему изоляции отдельных приложений, как в целях безопасности сервера, так и с целью удовлетворения различных требований приложений. Для этих целей идеально подходит Docker – программное обеспечение для автоматизации развёртывания и управления приложениями в среде виртуализации на уровне операционной системы. С его помощью можно быстро запускать любое приложение в легковесном изолированном контейнере с образом требуемой операционной системы, а также необходимыми программными компонентами. При этом контейнер может иметь лишь ограниченный доступ к файловой системе, что не позволит вредоносному приложению каким-либо образом навредить запустившему его серверу. Кроме того, Docker можно будет использовать и для развёртывания самой системы на рабочий (англ. *production*) сервер.

Во-вторых, встал вопрос выбора языка программирования. Очевидным кандидатом для написания подобных вещей является C++ ввиду скорости работы программ, написанных на нём. Однако в целях именно проектирования архитектуры серверной части был выбран язык программирования Python 3 (далее просто Python), у которого есть следующие преимущества перед C++:

1. Простота и малый объём кода.
2. Читаемость кода.
3. Скорость написания кода.
4. Наличие официальной библиотеки для взаимодействия с Docker.

Из альтернатив был также язык Go, обладающий теми же преимуществами перед C++, но предпочтение было отдано языку Python в связи с двумя факторами:

1. Наличие опыта работы с языком.
2. Веб-сервер системы пишется также пишется на языке Python с использованием фреймворка Django.

Тем не менее, выбор языка Python не означает, что придётся отказаться от преимуществ языка C++: когда структура серверного программного обеспечения будет полностью сформирована, код будет переписан на C++, что является несложной задачей.

2.2 Структура серверной части системы

В результате проектирования серверной части системы была предложена схема, изображённая рисунке 1.1. Несмотря на кажущийся небольшим набор компонентов системы, возможны довольно сложные взаимодействия между компонентами. Так, на рисунке 2.1 продемонстрирована схема последовательности действий внутри системы при выполнении студентом задания. В некоторых случаях, когда для выполнения задания не требуется запускать контейнеры (например, нужно просто прочитать теоретический материал), могут опускаться шаги с использованием Docker`а и контейнеров.

Несколько приложений при этом могут обмениваться между собой данными. Для этого есть несколько способов:

1. С помощью файловой системы, когда одна директория монтируется к нескольким контейнерам с приложениями.
2. С помощью сетевого общения: Docker создаёт отдельную виртуальную локальную сеть (VLAN), в которой находятся контейнеры. Так, например, можно общаться с помощью TCP- и UDP-пакетов.
3. Если необходимо общение серверных приложений в контейнерах и клиентского приложения в браузере пользователя, можно организовать его двумя способами:
    а) С помощью ретрансляции сообщений через сервер приложений.
    б) С помощью технологии WebSocket, которая может исключить шаг с ретрансляцией сообщений через сервер приложений и ускорить общение приложений.

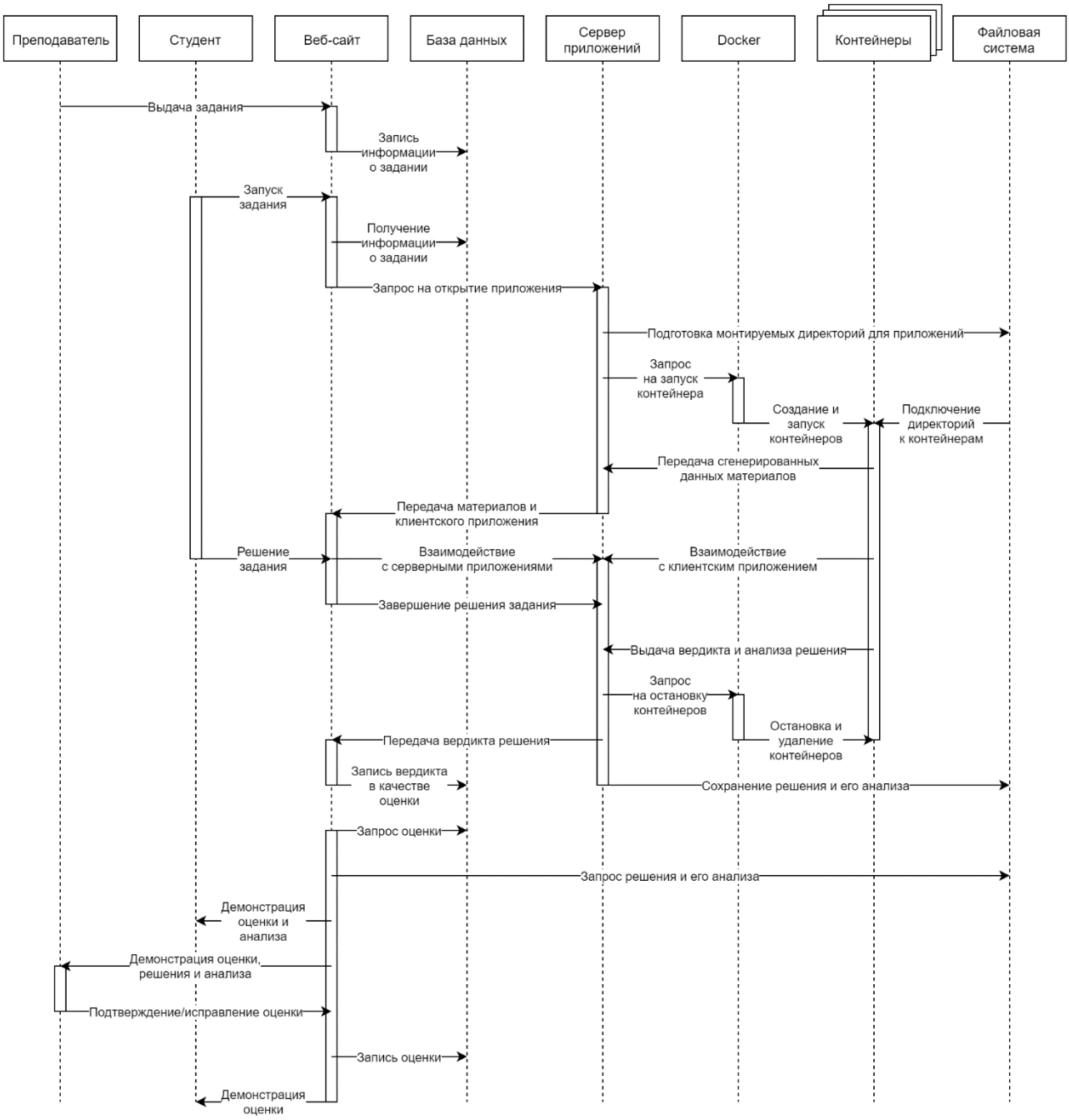

Рисунок 2.1 – Схема последовательности при решении студентом задания

Заключение

В результате прохождения производственной преддипломной практики были выполнены все поставленные задачи: продумать концепт серверной части системы и выбрать необходимые инструменты разработки. Кроме того, было принято участие в Международной научно-технической конференции студентов, аспирантов и молодых учёных «Научная сессия ТУСУР – 2018». Выступление проходило в секции 3.3 «Автоматизация управления в технике и образовании». Доклад опубликован в материалах конференции в томе 3, страницы 330-333.