МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»

Кафедра теории функций и стохастического анализа

РЕОРГАНИЗАЦИЯ СИСТЕМЫ ДОКУМЕНТООБОРОТА НА ПРИМЕРЕ ВЫСШЕГО УЧЕБНОГО ЗАВЕДЕНИЯ

АВТОРЕФЕРАТ МАГИСТЕРСКОЙ РАБОТЫ

студента 2 курса 248 группы
направления 09.04.03 — Прикладная информатика

механико-математического факультета
Алексеева Дмитрия Андреевича

Научный руководитель
доцент, к. ф.-м. н. _____ М. Г. Плешаков

Заведующий кафедрой
д. ф.-м. н., доцент _____ С. П. Сидоров

Саратов 2019

ВВЕДЕНИЕ

В настоящее время университет активно представляет одну из сил, которая влияет на формирование общественного мнения, занимается политическим прогнозированием, а также занимается научными исследованиями.

Университеты играют важную роль в развитии экономики, государства, гражданского общества, предоставляя знания, навыки, идеи и фундаментальные исследования, необходимые любой стране для обеспечения экономического, социального, политического развития и роста.

В современном мире количество информации неуклонно растет, что приводит к всеобщей компьютеризации данных. В связи с этим также растет потребность в быстром получении и анализе необходимой информации. Для повышения эффективности работы организаций, они оптимизируют протекающие внутри процессы посредством веб-сайтов, которые позволяют облегчить взаимодействие с пользователями.

Актуальность темы исследования. Повышенный интерес проявляется к возможностям своевременного получения полной и необходимой информации пользователям. Определяющими факторами наличия такого желания является технологический прогресс, а также доступность информации. Такие условия приводят к необходимости быстро принимать решения. Это становится возможным при наличии единого информационного пространства, неотъемлемой частью которого в современном обществе стала глобальная информационная сеть – Интернет.

Поднимаемая тема в данной работе является актуальной в силу переплетения в ней нескольких наиболее востребованных тематик. Во-первых, структура, которая существует в настоящее время в рассматриваемом высшем учебном заведении, уже имеется в большинстве вузов. А те вузы, которые все еще погрязли в бумажном документообороте стремятся к ней, в силу этого фактора, разрабатываемое решение поможет решить более общую задачу, показать так сказать некий шаблон решения подобной задачи.

Во-вторых, в ходе исследования проблематики работы и создания прототипа пришлось проанализировать большое количество данных, представленных в различных форматах. На основе анализа были разработаны алгоритмы обработки и преобразования данных из одного формата в другой

необходимый для предлагаемого решения. Этот процесс исследования, фильтрации и преобразования данных с целью извлечения необходимой и интересуемой информации представляет собой подмножество одной из наиболее популярной и быстро растущей области — анализ данных.

В-третьих, архитектура предлагаемой информационной системы базируется на микросервисной архитектуре, которая представляет собой разновидность распределенных систем. Развитие сетевых технологий предоставило удобную инфраструктуру для разработки больших распределенных программных комплексов. Необходимость разработки распределенных решений возникла по мере роста критической массы объемов информации, которыми оперируют учреждения. Область распределенных вычислительных систем в настоящее время характеризуется быстрыми темпами изменений идеологий и подходов, что свидетельствует о популярности и актуальности.

Объект исследования - высшие учебные заведения. Современные университеты — это огромные учреждения, обладающие своими сложными и запутанными внутренними процессами. Каждый университет обладает своими уникальными особенностями, но все они опираются на некоторую шаблонную структуру, которая присуща каждому университету.

Предмет исследования - система документооборота высших учебных заведений, с которой взаимодействуют пользователи напрямую. Структура системы документооборота высших учебных заведений рассматривалась на примере Саратовского Национального Исследовательского Государственного Университета имени Н.Г. Чернышевского. Рассматриваемая в работе часть системы документооборота упрощает доступ пользователя к внутренним процессам, протекающим в высшем учебном заведении.

Целью данной магистерской работы является создание информационной системы, которая позволит объединить все разрозненные части системы документооборота в едином месте. Помимо этого среди основных целей создания данной системы можно выделить: добавление гибкости и расширяемости системе документооборота для последующего внедрения новых функциональностей и предоставление открытого и понятного программного интерфейса для доступа к функциональностям системы. На основе принципов, заложенных в разрабатываемой информационной системе впоследствии

может быть развернута система документооборота для любого высшего учебного заведения. Основным мотивом данной работы является рассмотрение совершенного нового подхода для разработки системы документооборота ориентированной на пользователя.

Для достижения заявленной темы, необходимо решение ряда связанных **подзадач**, на основе которых можно будет впоследствии оценивать степень достижения поставленной цели. В итоге было решено обозначить следующие подзадачи:

1. Рассмотреть структуру текущей системы документооборота высших учебных заведений на примере СГУ имени Н.Г. Чернышевского;
2. Сформировать и систематизировать перечень требований для разрабатываемого приложения, а также представить их в едином и понятном формате;
3. Проанализировать существующие архитектуры для распределенных систем;
4. Спроектировать архитектуру приложения и систему документации;
5. Отобрать и исследовать перечень технологий среди существующих решений;
6. Реализовать перечисленные в техническом задании модули.

Апробация результатов исследования. Работа прошла апробацию на различных конференциях, в частности, январь 2018 года на ежегодной студенческой конференции «Актуальные проблемы математики и механики», которую проводил механико-математический факультет СГУ в апреле 2019 года, в секции «Анализ данных», в VII Международной молодежной научно-практической конференции «Математическое и компьютерное моделирование в экономике, страховании и управлении рисками», ноябрь 2018 года.

Структура работы. Дипломная работа состоит из введения, пяти глав, заключения, списка использованных источников и приложения.

ОСНОВНОЕ СОДЕРЖАНИЕ РАБОТЫ

Во введении обосновывается актуальность темы исследования, раскрывается степень ее разработанности, определяются цель, задачи, объект и предмет, научная новизна исследования, выявляется научная и практическая значимость исследования.

В главе «Предметная область» рассматриваются основные аспекты и механизмы, протекающие внутри рассматриваемой предметной области. Дается описание текущему положению высших учебных заведений в рамках общества. Приводится детальное описание внутренней структуры документооборота высшего учебного заведения, который брался за основу. Были рассмотрены основные структурные элементы системы документооборота, а также основные методы поступления информации в систему.

Основной акцент в этой главе делался на проблемы, с которыми может столкнуться система документооборота в скором времени, и чем это может обернуться для нее. Среди таких проблем отмечалось наличие огромного числа неструктурированной и бумажной информации, а также очень медленные темпы трансформирования такой информации в электронный и структурированный вид. Источником данных проблем на наш взгляд является использование монолитной архитектуры в рамках существующих электронных ресурсов.

Среди минусов использования монолитной архитектуры в работе выделяются: разобщенность ресурсов и перенасыщенность системы функциональностями. Также в этой главе приводится краткое описание решения, которое на наш взгляд призвано решить все эти проблемы.

В рамках главы «Техническое задание» рассматривалось несколько видов стандартов для составления технического задания, а также было сформировано само техническое задание для предложенного в рамках работы решения. Обуславливалась важность технического задания, а также наличие у него четкой и понятной структуры.

В данной главе были перечислены и рассмотрены различные виды стандартов составления технического задания. Сюда можно отнести как государственные стандарты, так и стандарты, основанные на процессах разработки. Для каждого из рассматриваемых стандартов было приведено его описание,

а также плюсы и минусы его использования в рамках текущей работы.

Изходя из цели работы, которая предполагает реализацию прототипа был выбран стандарт IEEE STD 830-1998. Этот стандарт не требует глубокой детализации требований к объекту разработки. Единственным условием создания требований на основе этого стандарта является разработка оптимальных требований, достаточных для общего понимания идеи.

В рамках главы «Проектирование» документировался процесс проектирования будущего приложения, а также некоторых важных процессов для этапа разработки. Здесь на основе составленного технического задания произошел отбор различных видов архитектур, были сформированы и оптимизированы некоторые процессы разработки, а также был выбран стандарт создания спецификаций для разрабатываемых решений.

Система документооборота уже на текущий момент представляет сложную и многогранную систему, которая постоянно растет и развивается. Следовательно, для того, чтобы добавить системе масштабируемости, необходимо реализовать распределенное решение. Это позволит системе увеличить производительность не только за счет вертикального масштабирования, но и за счет горизонтального масштабирования. В рамках веб-сайтов вертикальное масштабирование подразумевает наращивание производительности серверов, на которых расположен сайт, что не всегда возможно из-за ограниченности вычислительных возможностей. Горизонтальное масштабирование позволяет разбить систему на более мелкие структурные единицы, чтобы потом распределить их по различным физическим машинам.

В работе было рассмотрено несколько подходов к построению приложений. Рассматривались проблемы монолитного решения, а также каким образом произошел переход к распределенным архитектурам. Среди распределенных архитектур были рассмотрены сервис-ориентированная и микросервисная архитектуры. Обе архитектуры очень схожи между собой по принципам и концепциям, заложенным внутри них. Предпочтение было отдано микросервисному подходу, потому что в сервис-ориентированной архитектуре имеются недостатки, связанные с проблемами отказоустойчивости. Так при выходе из строя даже самого незначительного модуля может неожиданно завершиться все приложение.

Актуальным решением для высоконагруженных приложений в настоящее время является приложение, построенное на основе микросервисов. В таком приложении общая бизнес-задача разбита на отдельные части, каждая из которых имеет отдельное приложение(микросервис) с отдельной кодовой базой.

В соответствии с требованиями, полученными на этапе составления технического задания была спроектирована архитектура системы на базе микросервисного подхода. Высокоуровневая схема разрабатываемой системы представлена на рисунке 1.

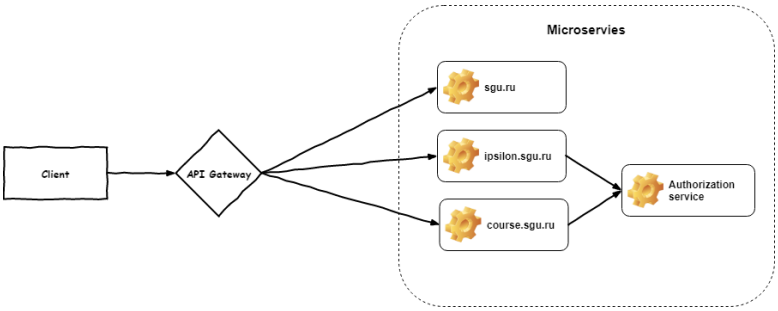

Рисунок 1 – Архитектура приложения

На этой схеме видно, что пользователь взаимодействует с системой с помощью промежуточного узла. Промежуточный узел будет реализован на основе принципов, заложенных в паттерне API Gateway. Функциональные блоки было решено сгруппировать на данном этапе на основе бизнес-задач. Поэтому для каждого ресурса СГУ был сформирован отдельный блок, а также в отдельный блок был вынесен сервис аутентификации для избежания дублируемости кода.

В рамках микросервисного подхода каждый из сервисов представляет собой самостоятельный и изолированный продукт. Помимо того, что цельное решение, которое будут представлять собой все сервисы вместе, должно обладать хорошей гибкостью и интегрируемостью новых функциональностей. Также каждый отдельный сервис внутри себя должен обладать гибкой структурой, которая позволит легко и просто поддерживать и расширять существующий код.

Для этого было решено разделить каждый сервис на две части: первая представляет собой слой представления с которым непосредственно работает

пользователь, вторая — это уровень бизнес-логики, здесь хранится вся бизнес-логика данного сервиса. Обе эти части разделены по разным проектам. Такой подход позволяет вести разработку цельного сервиса параллельно, а также улучшает процесс тестирования сервиса.

Каждый сервис на данный момент представляет лишь интерфейс прикладного программирования и пользователь может общаться с ним только в формате запрос/ответ без удобного и понятного обычному пользователю интерфейса. Поэтому в рамках данной работы использовался единый шаблон архитектуры при разработке слоя представления. Диаграмма классов для используемого решения приведена на рисунке 2.

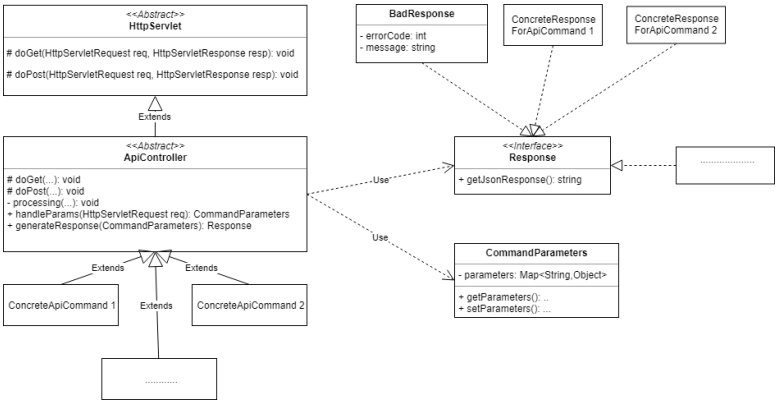

Рисунок 2 – Диаграмма классов

Следующим этапом в процессе проектирования был процесс проектирования документации. В процессе разработки любого программного продукта, особенное внимание уделяется процессу документирования разрабатываемого продукта. Документация является органической, составной частью программного продукта и требует значительных ресурсов для ее создания и применения.

Исходный код представляет собой продукт только в совокупности с комплексом документов, которые содержат полную и достаточную информацию о механизмах работы продукта. С помощью этого комплекса документов происходит оказание помощи пользователям в освоении продукта, его использовании и применении. Для этого вся сопутствующая исходному коду документация должна быть корректной, структурированной и понятно изложенной. Качество и полнота отображения существующего решения в документах должна ранжироваться в зависимости от целевого пользователя продукта.

Так для разработчиков, которые будут использовать возможности данного продукта, стоит детально описать интерфейс. Среди деталей стоит указать особенности работы с предоставляемым интерфейсом, но при этом можно опустить некоторые несущественные детали реализации.

Микросервисная архитектура подразумевает собой набор самостоятельных сервисов, которые общаются друг с другом, а на самом деле представляют один большой продукт. В данной работе для общения между микросервисами используется наиболее популярный и часто используемый протокол REST. Данный протокол не является самоописательным, что говорит о том, что клиент какого-либо сервиса должен знать точный формат запроса и ответа от данного сервиса. Поэтому процессу составления спецификации для каждого конкретного API требуется уделить особое внимание, так как понятность и простота использования разработанного решения сильно зависит от него. Спецификация позволит пользователю создать полную картину того, как необходимо взаимодействовать с данным продуктом. Спецификация — это документ, который точно, полно и в нужной форме определяет требования, устройство, поведение и другие особенности продукта.

Одним из самых популярных на данный момент способов документирования API является спецификация OpenAPI. The OpenAPI Specification (изначально известная как Swagger Specification) - это формализованная спецификация, а также целая экосистема, в которой существует множество инструментов для решения разнообразных задач по документированию API. Спецификация построена таким образом, что не зависит от языков программирования, и удобна в использовании как человеком, так и машиной.

Основная мощность данной спецификации заключается в ее экосистеме, если разобраться в существующих инструментах, то можно легко и просто интегрировать эту экосистему в жизненный цикл продукта, что позволит поддерживать документацию всегда в актуальном состоянии, уменьшить затраты на тестирование, а также снизить потребность в генерации человеком рутинных частей клиентского кода.

Спецификация OpenAPI поддерживает два формата написания документации JSON и YAML. Несмотря на то, что JSON-формат, представляет собой легковесный формат обмена данными и выигрывает в производитель-

ности обработки файла данного формата. Для документирования сервисов, разрабатываемых в данной работе было решено использовать формат YAML. YAML - это формат сериализации данных, концептуально близкий к языкам разметки, но ориентированный на удобство ввода-вывода типичных структур данных многих языков программирования. Язык в первую очередь предназначен для удобства чтения и редактирования.

В рамках главы «Реализация» описывались результаты реализации предлагаемого в работе решения. Для каждого разработанного сервиса приводился пример документации методов его конечных точек. На основе этой документации более подробно описывались сигнатуры имеющихся методов, примеры результатов выполнения этих методов. Помимо этого в этой главе был сформирован и описан оптимальный алгоритм разработки новых сервисов в рамках текущей работы.

Прототип, разрабатываемый в данной работе опирается на микросервисную архитектуру. При таком подходе внутренняя структура приложения делится на самостоятельные, изолированные сервисы. Каждый такой сервис представляет какую-нибудь одну цельную бизнес-задачу для данной предметной области. Так как процесс создания каждого отдельного сервиса схож, то для формализации процесса создания сервиса был сформирован наиболее оптимальный алгоритм разработки новых сервисов.

Первым делом, перед началом разработки сервиса — проектировалась спецификация к сервису в формате YAML. В спецификации подробно описывались методы, которые будет способен сервис обработать, а также параметры необходимые для корректной работы соответствующих методов. Затем на основе разработанной спецификации к сервису реализовывался базовый интерфейс без работающих функциональностей. После тестирования данного интерфейса, следовал процесс реализации соответствующих функциональностей.

На этапе составления технического задания были отобраны на наш взгляд наиболее часто используемые пользователями функциональности для каждого из существующих ресурсов СГУ. В итоге получилось 5 самостоятельных микросервисов. Три из которых взаимодействуют каждый с конкретным ресурсом СГУ соответственно. Один представляет собой служебный сервис.

Последний является промежуточным слоем между пользователем системы и реализованными функциональностями системы.

Служебный сервис или сервис аутентификации предоставляет файл Cookie после успешной аутентификации на выбранном сервисе. В текущей реализации с ним взаимодействуют ipsilon.sgu.ru и course.sgu.ru сервисы, функциональности которых подразумевают предварительную аутентификацию на соответствующем сервисе. Оба ресурса ipsilon.sgu.ru и course.sgu.ru при аутентификации пользователя на сервисе записывают в Cookie уникальный идентификатор сессии, для того чтобы впоследствии на основе него идентифицировать пользователя.

Сервис, который взаимодействует с сайтом ipsilon.sgu.ru обладает лишь одной функциональностью — возвращает список дисциплин для обращающегося пользователя и соответствующей этим дисциплинам подробной информацией. Эта функциональность дополнительно содержит фильтр, для возможности отобрать дисциплины на основе желаемого семестра.

Помимо этого сервис обладает служебным методом, который призван повысить безопасность за счет уменьшения количества ввода конфиденциальной информации для пользователя. Метод генерирует токен аутентификации для пользователя на основе его электронной почты и пароля, которые используются для доступа к ресурсу. В итоге пользователь для использования функциональностей сервиса, которые требуют аутентификации, использует токен аутентификации.

Сервис course.sgu.ru взаимодействует с одноименным ресурсом СГУ. Ресурс course.sgu.ru представляет собой площадку, целью которой является размещение курсов для проведения учебных занятий, практик, а также контроля успеваемости. Внутренняя реализация сервиса схожа с ipsilon.sgu.ru сервисом. Здесь также для получения доступа к бизнес-методам сервиса необходимо сначала пройти процесс аутентификации путем получения токена.

Данный сервис представляет лишь две функциональности. В рамках первой он возвращает список названий курсов, на которые подписан пользователь. Вторая функциональность позволяет получить список событий, который произошли или запланированы. События отбираются в рамках курсов на которые подписан пользователь. На сайте course.sgu.ru данная функциональ-

ность представлена в виде календаря, где дата на которую запланировано событие выделена и содержит описание деталей события.

Следующий сервис опирается на официальный сайт СГУ. Сайт sgu.ru не требует аутентификации пользователя для получения каких-либо данных. На текущий момент разработанный сервис на основе данного ресурса предоставляет две функциональности: получение расписания занятий и сессии.

При составлении промежуточного слоя было рассмотрено два наиболее простых и интуитивно понятных способа разделения пользователя и функциональностей. Было решено реализовать его на базе паттерна API Gateway. Это не что иное, как сервис, который является точкой входа в приложение доступной для внешнего мира. Помимо выполнения своих функциональностей этот слой представляет удобный и понятный интерфейс для пользователя. При обращении пользователь видит страницу, на которой представлены все сервисы и поддерживаемые ими методы. Макет главной страницы сайта представлен на рисунке 3. Макет формируется динамически на основе спецификаций к сервисам в формате OpenApi.

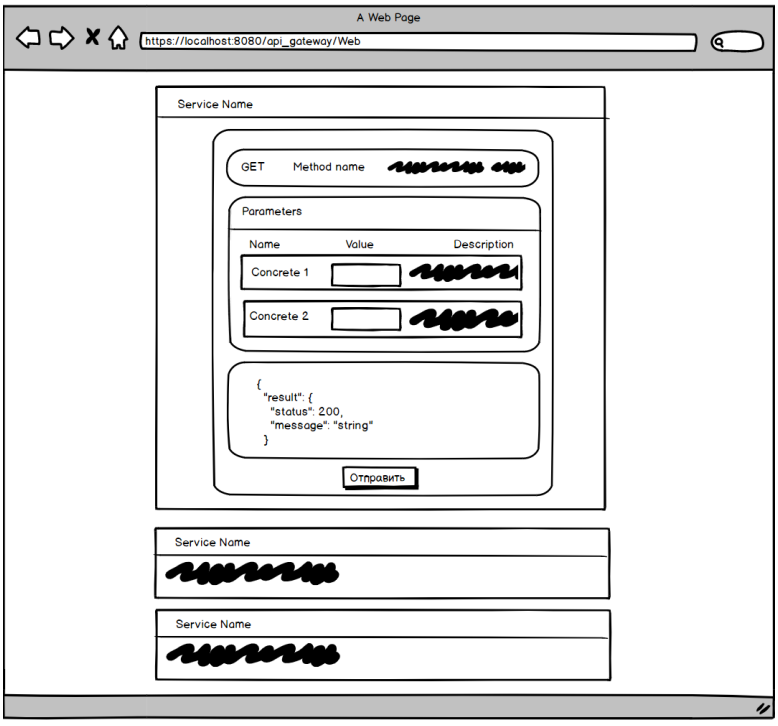

Рисунок 3 – Макет сайта после отправки пользователем запроса

В рамках главы «Используемые технологии» описывался перечень технологий, который был выбран для реализации прототипа предлагае-

мого решения. Помимо этого, описывалось почему выбор пал именно на такой набор инструментов, а также подчеркивалась важность этапа выбора списка инструментов для реализации в процессе проектирования программного продукта.

Прототип, который разрабатывался в данной работе был написан при помощи высокоуровневого, объектно-ориентированного и кросплатформенного языка Java. Каждый отдельный микросервис представляет собой отдельное веб-приложение, которое может обрабатывать HTTP-запрос и возвращать ответ пользователю.

Одним из ключевых компонентов разработки веб-приложений на языке Java является спецификация Java Servlet. Все разработанные сервисы были созданы в соответствии с этой спецификацией. Она представляет собой небольшое подключаемое расширение для сервера, которое расширяет функциональные возможности сервера. Servlet - это динамически загружаемый модуль, который обслуживает запросы от веб-сервера. Он полностью работает внутри виртуальной машины Java.

Спецификация Java Servlet подразумевает расширение возможностей сервера, поэтому запустить такое приложение отдельно от сервера невозможно. Поэтому в качестве веб-сервера в работе использовался Apache Tomcat. Он представляет собой контейнер веб-приложений на основе Java. Он бесплатный, наиболее популярный и написан на Java, что облегчает интеграцию приложений, написанных на Java.

При разработке приложения возникало множество однотипных и рутинных операций, таких как создание базовой структуры проекта, сборка проекта и развертывание проекта на тестовом сервере. Для того, чтобы упростить данные процессы, а также сделать их в универсальной форме было решено воспользоваться фреймворком для автоматизации сборки проектов Apache Maven.

ЗАКЛЮЧЕНИЕ

Ценность данной работы состоит в том, что она призвана обратить свое внимание на текущее положение дел в системе документооборота в высших учебных заведениях. Рассматриваемые проблемы хоть и не имеют острого характера на текущий момент, но впоследствии могут оказать негативное влияние на систему.

Предлагаемое в работе решение позволит системе документооборота быть готовой к различному роду проблемам. Оно позволит создать некоторую инфраструктуру для системы документооборота, которая будет обладать высокой отказоустойчивостью, гибкостью, а также хорошей расширяемостью. Еще одним немаловажным преимуществом может стать ее открытость, что позволит увеличить количество предоставляемых функциональностей за счет сторонних разработчиков.

Основным мотивом написания данной работы было рассмотрение совершенного нового подхода к разработке системы документооборота высших учебных заведений, с которой взаимодействуют пользователи. Целью было создание информационной системы. Она призвана решить проблемы текущей версии системы, а также снять ряд ограничений с системы. На основе этого решения можно будет, как реорганизовать существующую систему документооборота, так и создать новую на основе принципов, заложенных в этой работе.

В ходе работы удалось сформировать список требований к данной системе, а также определить набор функциональных блоков из которых будет состоять прототип для демонстрации возможностей системы. На основе полученных требований удалось решить наиболее важные вопросы, возникшие в процессе разработки такой системы. Среди них стоит отметить:

— удалось определиться с архитектурой информационной системы, чтобы она отвечала всем требованиям;
— сформировать и оптимизировать процессы по созданию и документации функциональностей в данной системы, что приведет к быстрому росту количества функциональностей в системе.

Основная цель и сформированные на основе нее подзадачи в рамках данной работы можно считать достигнутыми. Предложенный подход на те-

кущий момент не представляет полного решения, которое может быть легко и просто внедрено в систему высшего учебного заведения из-за огромного обилия тонкостей реализации распределенных приложений.

Код проекта представлен по ссылке https://bitbucket.org/account/user/grinchteam/projects/SGUAPI. Здесь для каждого сервиса был создан отдельный репозиторий. Внутри каждого репозитория содержатся папки Service и documentation. В папке Service лежит кодовая база соответствующего сервиса, а в папке documentation - спецификация к сервису в формате OpenAPI.