МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ   
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«ОРЛОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИМЕНИ И.С. ТУРГЕНЕВА»

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

по направлению подготовки

09.03.04 Программная инженерия

Направленность (профиль) Промышленная разработка программного обеспечения

Студента Шорина Владислава Дмитриевича шифр 171406

Институт приборостроения, автоматизации и информационных технологий

Тема выпускной квалификационной работы

«Разработка серверной подсистемы сервиса обмена сообщениями в ЕОИС ОГУ им. И.С. Тургенева»

Студент В.Д. Шорин

Руководитель А.И. Фролов

Нормоконтроль А.Ю. Ужаринский

Зав. кафедрой

программной инженерии А.И. Фролов

Орёл 2020

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ   
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«ОРЛОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИМЕНИ И.С. ТУРГЕНЕВА»

Институт приборостроения, автоматизации и информационных технологий

Кафедра программной инженерии

Направление 09.03.04 Программная инженерия

Направленность (профиль) Промышленная разработка программного обеспечения

УТВЕРЖДАЮ:

Зав. кафедрой

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.И. Фролов

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_ г.

ЗАДАНИЕ

на выполнение выпускной квалификационной работы

студента Шорина Владислава Дмитриевича шифр 171406

1 Тема ВКР «Разработка серверной подсистемы сервиса обмена сообщениями в ЕОИС ОГУ им. И.С. Тургенева»

Утверждена приказом по университету от «13» ноября 2020г. № 2-3142

2 Срок сдачи студентом законченной работы «26» июня 2020г.

3 Исходные данные к работе

Теоретический материал; информация о предметной области

4 Содержание пояснительной записки (перечень подлежащих разработке вопросов)

5 Перечень демонстрационного материала

Презентация, отображающая основные этапы и результаты выполнения ВКР

Дата выдачи задания « 20 » мая 2020 г.

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.И. Фролов

(подпись)

Задание принял к исполнению \_\_\_\_\_\_\_\_\_\_\_\_\_\_ В.Д. Шорин

(подпись)

КАЛЕНДАРНЫЙ ПЛАН

|  |  |  |
| --- | --- | --- |
| Наименование этапов работы | Сроки выполнения  Этапов работы | Примечание |
| Анализ задачи | 27.05.2020 |  |
| Определение спецификаций | 29.05.2020 |  |
| Проектирование программного обеспечения | 10.06.2020 |  |
| Реализация программного обеспечения | 19.06.2020 |  |
| Оформление ВКР | 26.06.2020 |  |

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_ В.Д. Шорин

(подпись)

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.И. Фролов

(подпись)

**АННОТАЦИЯ**

ВКР 98 с., 22 рис., 1 табл., 8 источников, 1 прил.

*КЛЮЧЕВЫЕ СЛОВА*.

Выпускная квалификационная работа на тему «Разработка серверной подсистемы сервиса обмена сообщениями в ЕОИС ОГУ им. И.С. Тургенева».

В первой главе ВКР проводится анализ задачи разработки

Во второй главе

В третьей главе

В четвертой главе

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 6](#_Toc59713580)

[1 АНАЛИЗ ЗАДАЧИ РАЗРАБОТКИ СПЕЦИАЛИЗИРОВАННОГО СЕРВИСА ОБМЕНА СООБЩЕНИЯМИ В ЕОИС ОГУ им. И.С. Тургенева 7](#_Toc59713581)

[1.1 Структура сервисов личных кабинетов студента и преподавателя ЕОИС ОГУ им. И.С. Тургенева 7](#_Toc59713582)

[1.2 Функциональные возможности существующей системы обмена сообщениями 9](#_Toc59713583)

[1.3 Анализ аналогичных решений и возможностей применения существующих программных средств 9](#_Toc59713584)

[1.4 Постановка задачи и формулировка требований 16](#_Toc59713585)

[1.4.1 Возможности сервиса обмена сообщениями 16](#_Toc59713586)

[1.4.2 Требования к серверной подсистеме 16](#_Toc59713587)

[2 ОПРЕДЕЛЕНИЕ СПЕЦИФИКАЦИЙ К СЕРВЕРНОЙ ПОДСИСТЕМЕ СЕРВИСА ОБМЕНА СООБЩЕНИЯМИ 18](#_Toc59713588)

[2.1 Разработка архитектуры 18](#_Toc59713589)

[2.2 Разработка функциональных спецификаций 18](#_Toc59713590)

[2.3 Разработка информационных спецификаций 18](#_Toc59713591)

[2.4 Разработка поведенческих спецификаций 18](#_Toc59713592)

[3 ПРОЕКТИРОВАНИЕ СЕРВЕРНОЙ ПОДСИСТЕМЫ СЕРВИСА ОБМЕНА СООБЩЕНИЯМИ 19](#_Toc59713593)

[3.1 Проектирование структуры программного обеспечения 19](#_Toc59713594)

[3.2 Проектирование основных компонентов программного обеспечения 19](#_Toc59713595)

[3.3 Проектирование API 19](#_Toc59713596)

[3.4 Проектирование БД 19](#_Toc59713597)

[3.5 Разработка .. 19](#_Toc59713598)

[4 РЕАЛИЗАЦИЯ, ТЕСТИРОВАНИЕ И ОТЛАДКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ 20](#_Toc59713599)

[4.1 Реализация алгоритма (контроллеры?)… 20](#_Toc59713600)

[4.2 Реализация АПИ?… 20](#_Toc59713601)

[4.3 Тестирование и отладка программного обеспечения 20](#_Toc59713602)

[ЗАКЛЮЧЕНИЕ 21](#_Toc59713603)

[СПИСОК ЛИТЕРАТУРЫ 22](#_Toc59713604)

[ПРИЛОЖЕНИЕ А 23](#_Toc59713605)

[Удостоверяющий лист № 160662 24](#_Toc59713606)

[ИНФОРМАЦИОННО-ПОИСКОВАЯ ХАРАКТЕРИСТИКА ДОКУМЕНТА НА ЭЛЕКТРОННОМ НОСИТЕЛЕ 25](#_Toc59713607)

# ВВЕДЕНИЕ

Важной частью жизни человека является общение. В современном мире невозможно представить жизнь человека без использования какого-либо гаджета: телефона, планшета, компьютера и т.д. Поэтому, большая часть общения между людьми перешла именно в наши технические средства и сервисы обмена сообщениями, или мессенджеры, стали неотъемлемой их частью. Данные сервисы позволяют человеку поддерживать контакт с важными в его жизни людьми в практически любой момент и на любом расстоянии и обмениваться с ними почти любой информацией.

Сервисы обмена сообщениями играют важную роль не только в потребительской сфере, но также и в деловой. Это позволяет упростить рабочий процесс посредством ускорения обмена необходимой информацией между соответствующими группами людей.

Также, сервисы обмена сообщениями являются основными средствами коммуникации в процессе образования, особенно дистанционного. Они позволяют упростить процессы общения между студентом и преподавателем, выполнения различных контрольных работ, лабораторных работ и т.п.

Целью данной работы является разработка серверной подсистемы сервиса обмена сообщениями в единой образовательной информационной системе (ЕОИС) ОГУ им. И.С. Тургенева.

Для достижения поставленной цели необходимо решить следующие задачи:

1. проанализировать задачу разработки специализированного сервиса обмена сообщениями;
2. выполнить анализ требований и определение спецификаций программного обеспечения специализированного сервиса обмена сообщениями;
3. спроектировать архитектуру серверной подсистемы и базы данных сервиса обмена сообщениями;
4. изучить API (прикладной программный интерфейс веб-сервера университета, который предоставляет возможность авторизации пользователя);
5. спроектировать собственное API сервера;
6. реализовать серверную подсистему.

# АНАЛИЗ ЗАДАЧИ РАЗРАБОТКИ СПЕЦИАЛИЗИРОВАННОГО СЕРВИСА ОБМЕНА СООБЩЕНИЯМИ В ЕОИС ОГУ им. И.С. Тургенева

## Структура сервисов личных кабинетов студента и преподавателя ЕОИС ОГУ им. И.С. Тургенева

Вслед за развитием информационных технологий во всем мире и расширений перечня возможностей использования различных технических средств претерпевает существенные изменения и сфера образования. Немаловажную роль играют и непредвиденные обстоятельства, которые вынуждают образовательные учреждения использовать дистанционные технологии. Вследствие совокупности данных факторов вытекает необходимость внедрения современных и удобных средств реализации всех образовательных процессов и технологий. К таким средствам можно отнести: официальные сайты образовательных учреждений со всей необходимой информацией, личные кабинеты студентов и преподавателей, средства обмена информацией между участниками образовательного процесса.

Для создания современных и практичных условий проведения образовательного процесса в ОГУ им. И.С. Тургенева создан соответствующий официальный сайт. На данном сайте для участников образовательного процесса создан личный кабинет, для студента и преподавателя соответственно, необходимый для того, чтобы:

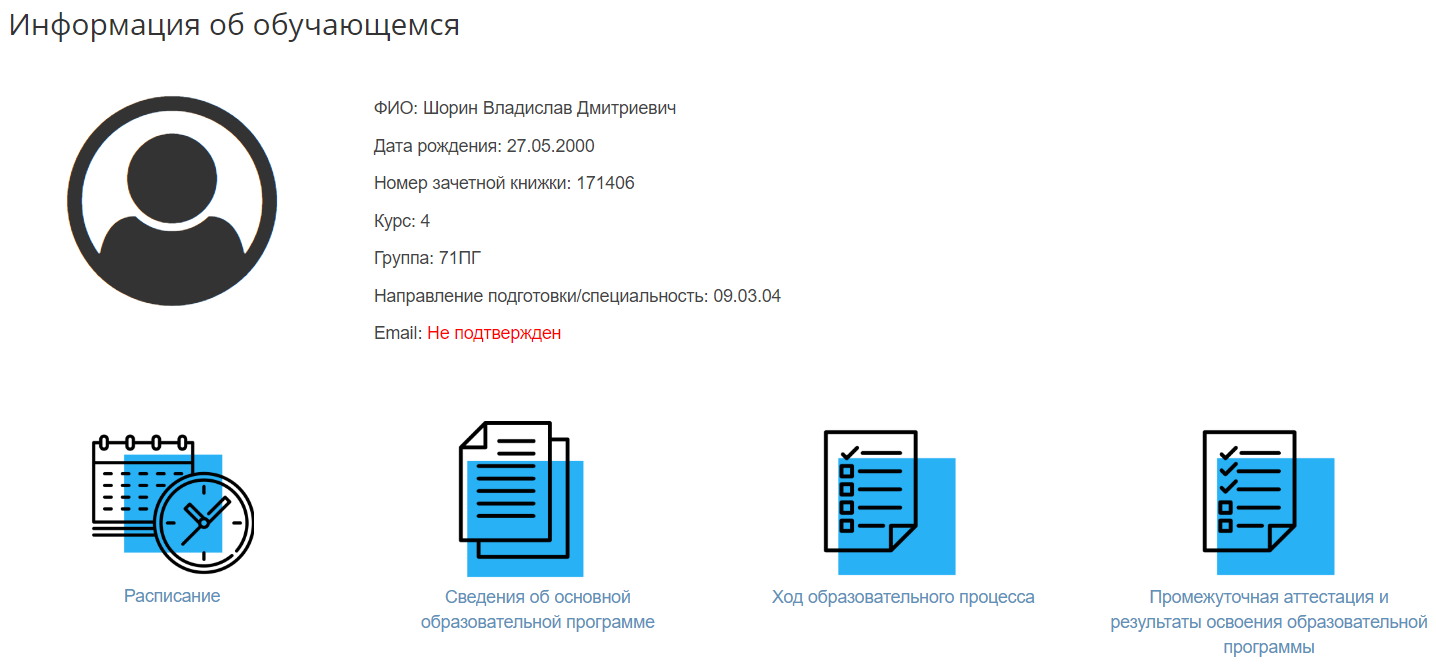
* получать актуальную информацию об учебном процессе в короткий промежуток времени;
* осуществлять взаимодействие студента с профессорско-преподавательским составом;
* иметь доступ к информации в электронных библиотечных системах и электронных образовательных ресурсах, связанный с высшим учебным заведением и многие другие функции.

Для, того, чтобы воспользоваться всем доступным функционалом личного кабинета, в первую очередь, необходимо являться участником образовательного процесса в ОГУ им. И.С. Тургенева (т.е. быть студентом или членом профессорско-преподавательского состава). При соблюдении данного условия, получение доступа личному кабинету студента или преподавателя сводится к авторизации в личном кабинете, т.е. ввод персональных данных участника в соответствующей графе: для студента это номер зачетной книжки, фамилия, имя и пароль. Для преподавателя данный процесс аналогичен.

При входе в учетную запись, пользователю предоставляются поля с его личной информацией, а также список доступных функциональных возможностей. Структура сервисов личных кабинетов студента и преподавателя будет иметь некоторые различия.

На рисунке 1.1.1 представлен интерфейс веб-версии личного кабинета студента ОГУ им. И.С. Тургенева. Вначале представлена личная информация, идентифицирующая студента (фотография (при наличии), ФИО, курс обучения и т.д.). Далее идет список доступных ему функциональных возможностей:

* расписание – обеспечивает доступ к актуальному расписанию занятий, отображая для каждого ФИО преподавателя, название предмета, тип занятия, время проведения, место проведения;
* сведения об основной образовательной программе – включает в себя сведения об образовательной программе, рабочие программы по предметам, программы практик, ГИА, методические материалы;
* ход образовательного процесса – показывает текущие результаты освоения рабочей программы, выраженные в баллах балльно-рейтинговой системы;
* промежуточная аттестация и результаты освоения образовательной программы – представляет студенту результаты промежуточной аттестации за каждый прошедший семестр и текущей сессии;
* электронные библиотечные системы и электронные образовательные ресурсы – предоставляет доступ к электронной версии научной библиотеки ОГУ им. И.С. Тургенева;
* электронное портфолио – содержит портфолио студента как в учебной и научно-исследовательской, так и во внеучебной деятельности (дипломы, грамоты, курсовые работы и т.д.);
* анкетирование – позволяет студенту пройти анкетирование для оценки качества учебного процесса в высшем учебном заведении;
* записаться на спортивную секцию – дает доступ студенту к возможности записи на интересующую его спортивную секцию (только для тех, у которых в учебном плане на следующий семестр предусмотрены занятия по физической культуре);
* итоговая аттестация – содержит результаты государственной итоговой аттестации;
* система сообщений – сервис обмена сообщениями между студентами и преподавателями внутри университета;
* оценка/рецензия – позволяет оставить отзыв или рецензию о ходе образовательного процесса;
* настройка личного кабинета – предоставляет возможность смены электронной почты и пароля личного кабинета;
* информация о руководителях института/факультета – предоставляет информацию о руководителях института или факультета (ФИО, должность, номер телефона, почта);
* IPR BOOKS – позволяет перейти на портал IPR BOOKS.



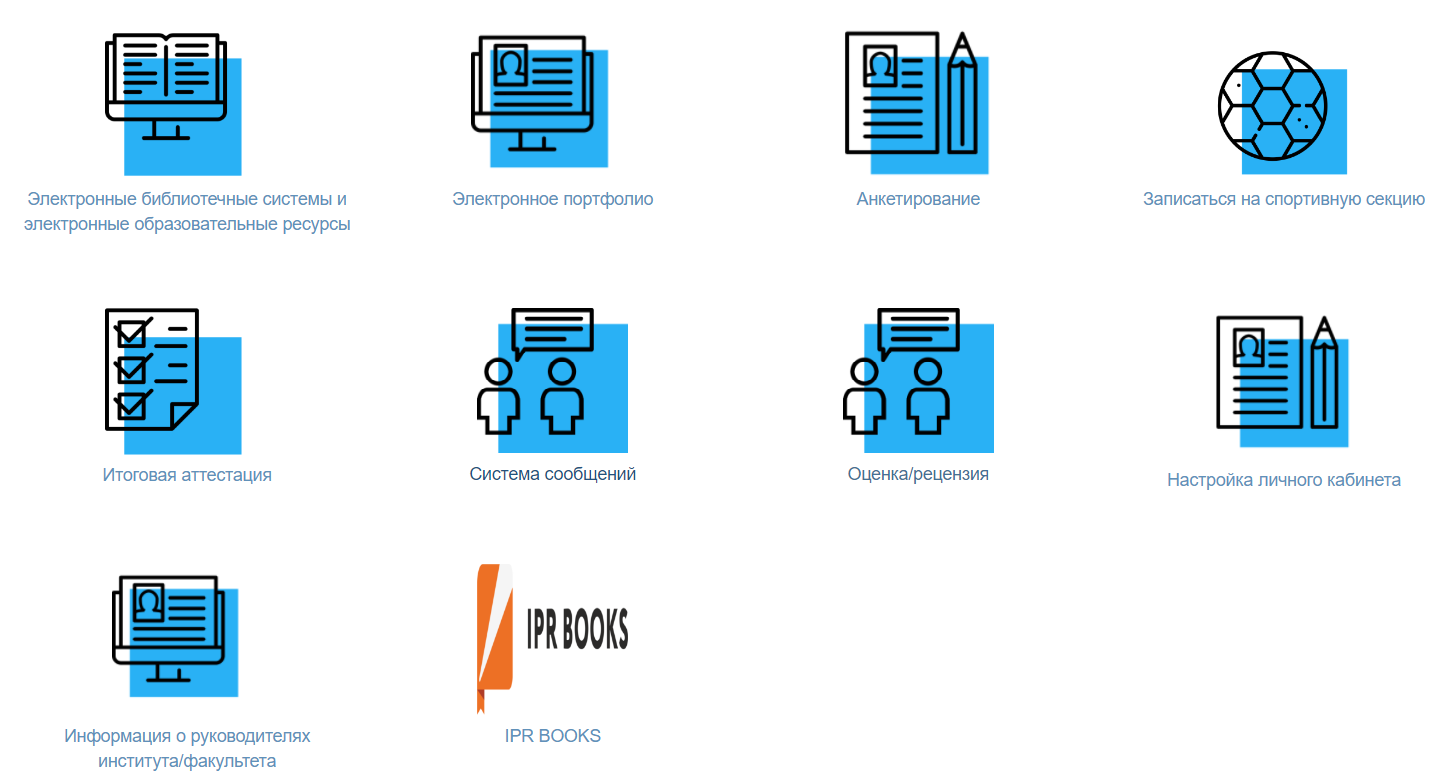


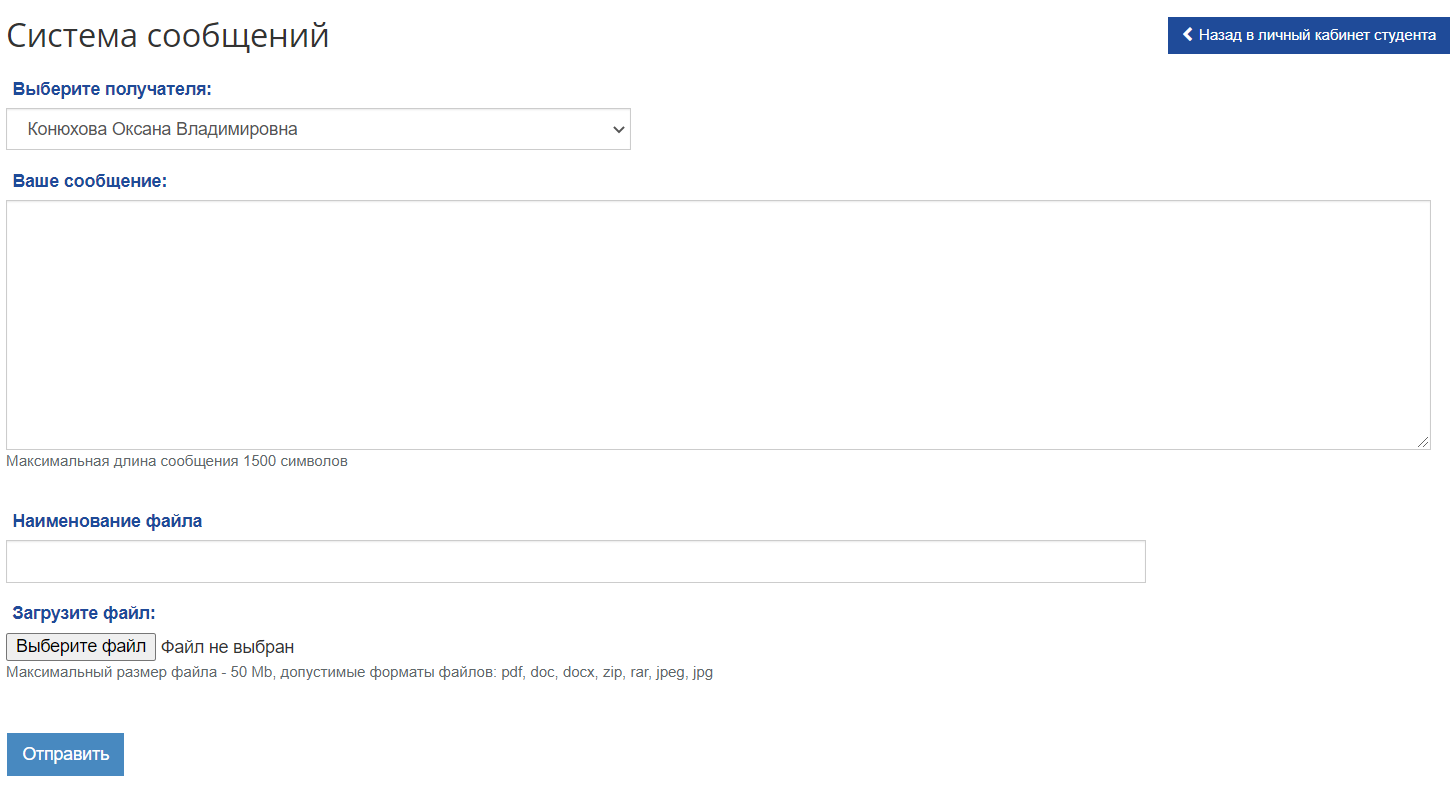
Рисунок 1.1.1 – Интерфейс веб-версии личного кабинета студента

## Функциональные возможности существующей системы обмена сообщениями

У студента и преподавателя отличаются не только личные кабинеты в целом, но и некоторые их части, в том числе система обмена сообщениями. У каждой стороны образовательного процесса имеются как общие функциональные возможности, так и индивидуальные (они в большей степени присущи профессорско-преподавательскому составу).

Студент в своем личном кабинете выбирает пункт «Система сообщений», после чего попадает на страницу сервиса обмена сообщениями (рисунок 1.2.1). В существующей системе обмена сообщениями данному участнику образовательного процесса доступны следующие функциональные возможности:

* выбор получателя сообщения из списка преподавателей, которые закреплены за предметами в текущем семестре;
* набор текстового сообщения получателю;
* прикрепление файла с дополнительными материалами в одном из доступных форматов (pdf, dox, docx, zip, rar, jpeg, jpg);
* просмотр списка входящих сообщений;
* сортировка входящих сообщений (от всех отправителей; от конкретного преподавателя)
* просмотр списка отправленных сообщений;
* сортировка отправленных сообщений (для всех получателей; для конкретного преподавателя)



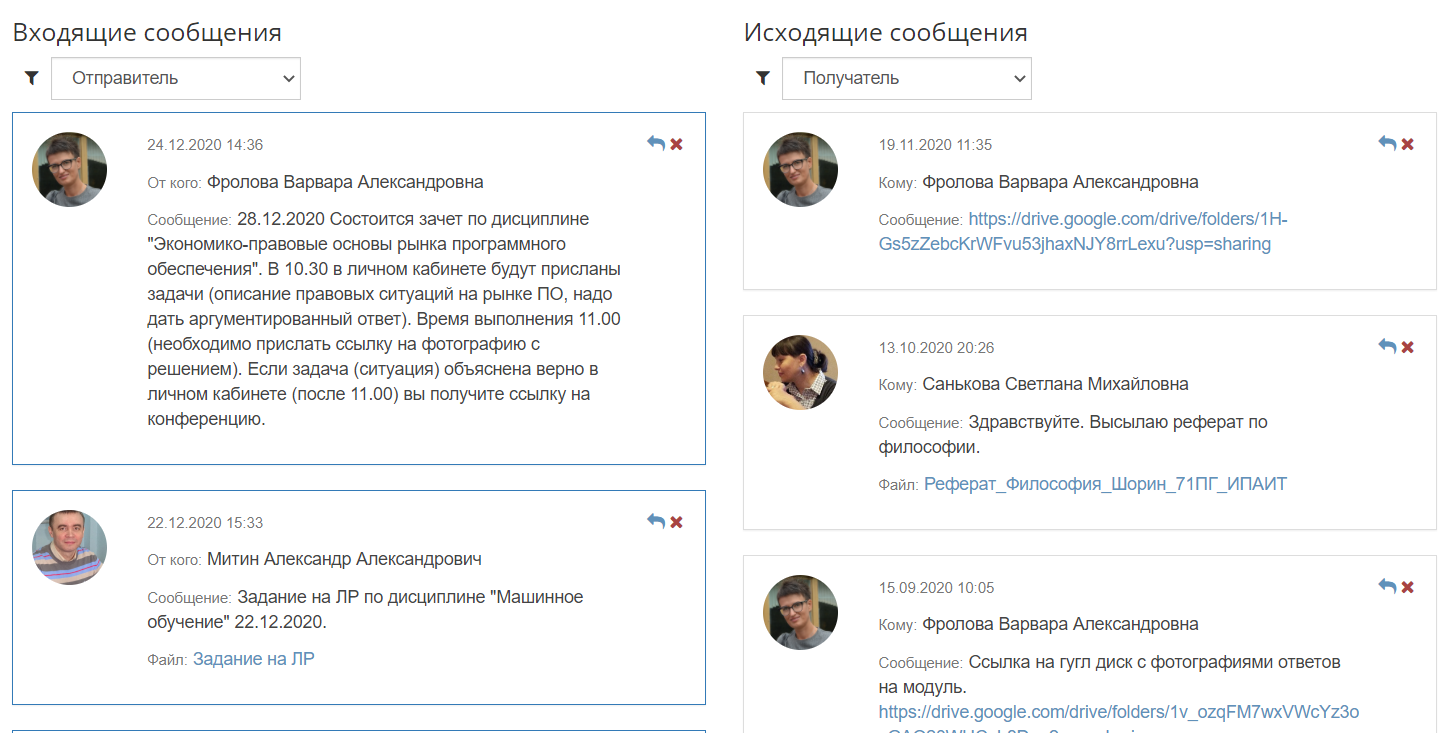


Рисунок 1.2.1 – Интерфейс веб-версии системы сообщений студента

## Анализ аналогичных решений и возможностей применения существующих программных средств

В целях изучения актуальности разрабатываемой программной системы, перед началом разработки требуется провести поиск и сравнение уже существующих решений по данной тематике, а также выделить необходимые критерии сравнения.

Исходя из того факта, что программное обеспечение разрабатывается непосредственно для нужд ОГУ им. И.С. Тургенева, конкурентоспособных аналогов, соответствующих выявленным функциональным и нефункциональным требованиям в полной мере, не существует. Однако, мы можем рассмотреть программное обеспечение, чей функционал наиболее близок к требуемому.

В открытых источниках не имеется информации о сервисах обмена сообщениями, функционал которых в большей степени предназначен для удовлетворения потребностей высших учебных заведений, а не широкого круга лиц, потому что данные сервисы входят в состав функциональных возможностей личных кабинетов соответствующих высших учебных заведений, доступ к которым достаточно проблематично получить, не являясь участником образовательного процесса соответствующего университета. Поэтому для изучения и сравнения с разрабатываемой программной системой будут выбраны такие популярные мессенджеры, как «WhatsApp», «Viber» и «Telegram».

Данный анализ позволит, в первую очередь, выявить наиболее востребованные направления как в функциональной части современных сервисов обмена сообщениями, так и в их пользовательском интерфейсе (от удобства и простоты использования до дизайна). Результаты проведенного изучения аналогов можно использовать для последующей возможной корректировки функциональных и нефункциональных требований, выявленных ранее.

Выделим необходимые критерии, по которым будет производиться сравнительный анализ в рамках серверной подсистемы сервиса обмена сообщениями:

1. возможность создания индивидуального чата;
2. возможность создания группового чата;
3. возможность объединения чатов в совокупности по каким-либо критериям;
4. возможность обмена материалами (аудио- и видеоформата, изображения, электронными документы и т.д.);
5. возможность взаимодействия с ЕОИС какого-либо выбранного высшего учебного заведения (в данном случае, ОГУ им. И.С. Тургенева);
6. стек технологий, применяемых для реализации серверной подсистемы;
7. архитектура сервера.

Первым предметом рассмотрения выберем популярный мессенджер «WhatsApp», который доступен на различных платформах (в том числе, и мобильных).

Основной функционал данного приложения заключается в следующем:

* возможность обмена текстовыми сообщениями между пользователями;
* возможность обмена аудио- и видеоматериалами, а также электронными документами;
* создание как индивидуальных, так и групповых чатов;
* аудио- и видеозвонки;
* добавление контактов с контактной книги телефона в приложение;

На рисунке 1.3.1 представлен интерфейс мобильного приложения «WhatsApp», скачанного с мобильного сервиса «Google Play», на котором можно заметить достаточно удобный и интуитивно понятный интерфейс.

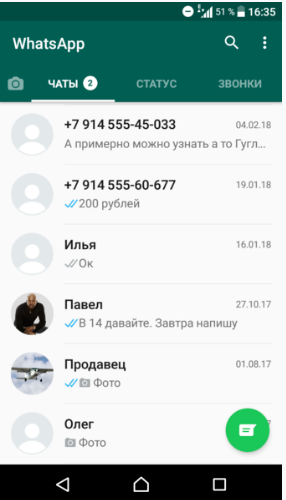


Рисунок 1.3.1 – Интерфейс приложения «WhatsApp»

Однако, в данном мессенджере отсутствует какая-либо возможность объединения чатов в группы по какому-либо критерию. Также, приложение направлено на предоставление услуг связи широкому кругу лиц, поэтому там отсутствует возможность взаимодействия с той или иной ЕОИС.

Основываясь на исследовании данных, доступных в открытых источниках сети Интернет, получены следующие сведения об архитектуре «WhatsApp»:

* языки программирования: Erlang, PHP;
* база данных: Mnesia;
* операционная система: FreeBSD;
* веб-серверы: Yaws, lighttpd;
* протокол обмена сообщениями: свой XMPP;

Следующим косвенным аналогом рассматривается «Viber». Данный мессенджер имеет аналогичный с предыдущим основной функционал. Отличия между ними заключаются на немного более глубоким уровне рассмотрения и выражаются в следующем:

* в «WhatsApp» присутствует веб-версия приложения, а в «Viber» нет;
* в «Viber» можно создавать опросы в групповом чате;
* возможность создания секретных чатов в «Viber» и т.д.

На рисунке 1.3.2 представлен интерфейс мессенджера «Viber». Как можно заметить, все достаточно удобно и понятно: представлен список чатов, в которые можно перейти просто выбрав нажатием необходимый, можно создать новый чат посредством нажатия на соответствующую иконку и т.д.

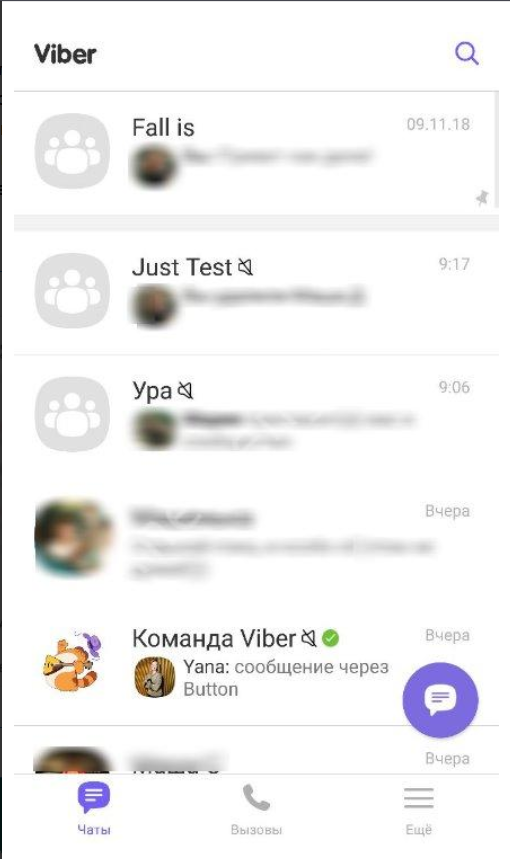


Рисунок 1.3.2 – Интерфейс приложения «Viber»

Однако, в мессенджере «Viber» также, как и в «WhatsApp», ожидаемо отсутствует возможность взаимодействия с ЕОИС высших учебных заведений, а также нет функции группировки чатов.

Проведя исследование доступной информации об архитектуре серверной части мессенджера «Viber», получился следующий стек технологий:

* + языки программирования: С++, Python, Java, Shell script, PHP;
  + база данных: Couchbase, несколько кластеров NoSQL баз данных;
  + операционная система: нет данных;
  + веб-сервер: nginx
  + протокол обмена сообщениями: WebRTC.

Помимо основного функционала, представленного в мессенджерах «WhatsApp» и «Viber», «Telegram» обладает также некоторыми следующими преимуществами:

* удаление любых сообщений не только у себя, но и у собеседника без временных ограничений;
* таймер на удаление сообщения в секретном чате (сообщение или файл удаляются через определенное время после их открытия или прочтения);
* наличие «сохранения сообщений» – отдельный диалог с «самим собой», куда можно пересылать сообщения и загружать неограниченное количество файлов (по сути, бесконечное облачное хранилище);
* группировка чатов – возможность создания тематических папок, куда пользователь может помещать неограниченное количество чатов по интересам.

На рисунке 1.3.3 представлен интерфейс приложения «Telegram». В верхней части экрана можно заметить список тематических папок («Инстаграм», «Друзья» и т.д.). Сам пользовательский интерфейс мессенджера, как и у других, достаточно прост и понятен интуитивно: список чатов, в каждый из которых можно перейти по нажатию, поиск по чатам, создание чата и т.д.

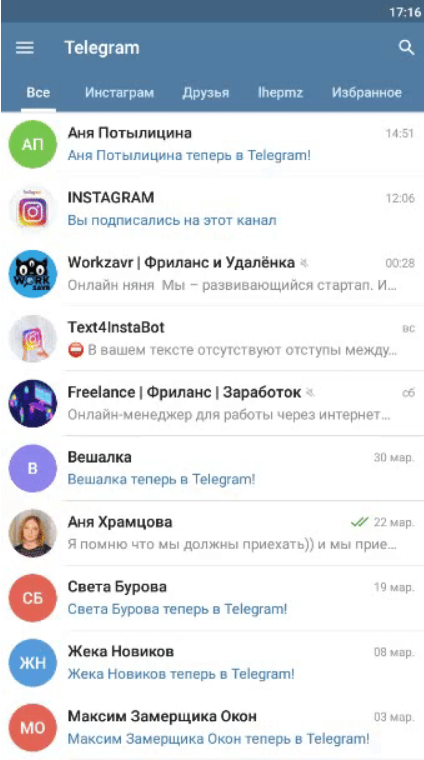


Рисунок 1.3.3 – Интерфейс приложения «Telegram»

Однако, поскольку данный сервис обмена сообщениями и медиафайлами также, как и вышеупомянутые, направлен на предоставление услуг обмена информацией широкому кругу лиц, здесь также отсутствует функциональная возможность взаимодействия с ЕОИС высшего учебного заведения.

Ниже представлена информация о стеке применяемых технологий серверной части мессенджера «Telegram», имеющаяся в открытом доступе:

* языки программирования: AspectJ, Swift, Objective-C, C#, Java;
* база данных: TDLib;
* операционная система: нет данных;
* веб-сервер: nginx
* протокол обмена сообщениями: MTProto.

Архитектура всех представленных аналогов, также как и архитектура разрабатываемого сервиса обмена сообщениями, построены по принципу MVC. Архитектурный паттерн MVC (Model-View-Controller, Модель-Представление-Контроллер) представляет собой схему разделения создаваемого приложения, состоящую из трех частей:

* модель – предоставляет данные и осуществляет отклик на команды, приходящие от контроллера (при этом изменяет свое состояние);
* представление – реагирует на изменения в модели и отображает ее данные удобным для пользователя способом;
* контроллер – получает действия от пользователя и, используя модель и представление, реализует их.

Сервис, который разрабатывается в рамках данной работы имеет следующие преимущества относительно аналогов:

* взаимодействие с ЕОИС высшего учебного заведения (ОГУ им. И.С. Тургенева в данном случае);
* возможность группировки чатов по интересам (в данном случае, по дисциплинам, курсам, типам предмета и т.д.);
* отсутствие необходимости хранения информации о других участниках учебного процесса. Поскольку сервис взаимодействует с ЕОИС ВУЗа, то вся информация берется из запросов на сервер ЕОИС, благодаря чему у пользователя отсутствует необходимость хранения на своем устройстве большого количества телефонных номеров, контактных данных и остальной необходимой информации о других участниках).

Серверная подсистема системы обмена сообщениями будет разрабатываться с применением следующих технологий:

* язык программирования: PHP (с использованием фреймворка Laravel);
* база данных: Microsoft SQLServer;
* операционная система: Windows;
* веб-сервер: Apache;
* протокол обмена сообщениями: HTTP.

Изучив существующие аналоги, для упрощения процесса анализа преимуществ и недостатков каждого приложения необходимо построить сравнительную таблицу, в которой будут четко представлены результаты их изучения. В таблице 1.4.1 представлен сравнительный анализ аналогов разрабатываемого программного обеспечения.

Таблица 1.3.1 – Сравнение выбранных аналогов с разрабатываемым приложением

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | WhatsApp | Viber | Telegram | Разрабатываемый сервис |
| возможность создания индивидуального чата | + | + | + | + |
| возможность создания группового чата | + | + | + | + |
| возможность объединения чатов в совокупности по каким-либо критериям | - | - | + | + |
| возможность обмена материалами (аудио- и видеоформата, изображения, электронными документы и т.д.) | + | + | + | + |
| удобный и понятный интерфейс | + | + | + | + |
| возможность взаимодействия с ЕОИС какого-либо выбранного высшего учебного заведения (в данном случае, ОГУ им. И.С. Тургенева) | - | - | - | + |
| язык(и) программирования, фреймворки | Erlang, PHP | С++, Python, Java, Shell script, PHP | AspectJ | PHP, Laravel |
| база(ы) данных | Mnesia | Couchbase, несколько кластеров NoSQL баз данных | TDLib | Microsoft SQLServer |
| операционная система | FreeBSD | нет данных | нет данных | Windows |
| веб-сервер(ы) | Yaws, lighttpd | nginx | nginx | Apache |
| протокол обмена сообщениями | свой XMPP | WebRTC | MTProto | HTPP |

На основе представленных в таблице 1.3.1. данных, можно сделать вывод о том, что все сервисы обмена сообщения помимо прямых функций по обмену сообщениями и созданию индивидуальных и групповых чатов, предоставляют также возможности по обмену электронными документами, аудио- и видеоматериалами. К тому же, каждый из аналогов имеет удобный и понятный интерфейс, что является одним из самых важных критериев популярности приложения. Однако, только «Telegram» позволяет объединять чаты в группы по интересам. Но в тоже время, с ростом количества групп, использование данной функциональной возможности становится все менее удобным, поскольку необходимо будет достаточно долго листать список групп в поисках необходимого чата.

Отсюда можно сделать вывод о том, что на текущий момент на рынке программного обеспечения отсутствует открытое решение, которое поможет максимально упростить процесс взаимодействия студента и преподавателя. Исходя из проведенного анализа выявленных аналогов можно сделать вывод об актуальности разрабатываемого сервиса обмена сообщениями.

## Постановка задачи и формулировка требований

Изучив структуры сервисов личных кабинетов студента и преподавателя, а также проведя анализ функциональных возможностей существующей системы обмена сообщениями, мы можем сделать вывод о том, что имеющаяся в данный момент система имеет некоторое количество недостатков, а именно:

1. сложный процесс доступа к сервису – необходимо сначала авторизироваться в личном кабинете, затем выбрать соответствующий пункт меню и затем уже имеется возможность отправки сообщения. Помимо этого, также накладывается ограничение на продолжительность сессии авторизации, что влечет за собой периодические повторения вышеуказанных действий для повторного доступа к сервису обмена сообщениями;
2. неудобный процесс формирования и отправки сообщения – для студента и преподавателя: необходимо сначала выбрать преподавателя из списка, затем набрать сообщение в соответствующей поле и прикрепить при необходимости файл с дополнительными материалами;
3. отсутствие уведомлений о сообщениях – если участнику образовательного процесса приходит новое сообщение, то он никаким образом не сможет об этом узнать, кроме личной проверки системы сообщений (что включает также первый пункт о сложности процесса доступа);
4. неудобный интерфейс отображения сообщений – все сообщения отражаются в том порядке, в котором были получены/отправлены на нескольких страницах;
5. отсутствие какой-либо группировки сообщений по темам – все сообщения можно отфильтровать только по их отправителю или получателю, что накладывает определенные трудности по нахождению какого-либо определенного сообщения по интересующей теме в случае наличия длинной переписки с конкретным пользователем;
6. может еще что то дописать

На основе выявленных недостатков, а также проведенного анализа выбранных аналогов можем сделать вывод о том, что главной задачей данной работы является разработка сервиса обмена сообщениями, который упростит процесс обмена сообщениями между преподавателем и студентом, т.е. позволит им мгновенно обмениваться сообщениями, получать уведомления о получении нового сообщения, группировать чаты по тематикам и т.д.

Разрабатываемый сервис обмена сообщениями будет состоять из серверной части и мобильного приложения. Мобильное приложение соответствует представлению в архитектурном паттерне MVC, а сервер содержит в себе модель и контроллер, соответственно.

## Возможности сервиса обмена сообщениями

Разрабатываемый сервис обмена сообщениями должен реализовывать следующие возможности:

* создание индивидуальных и групповых чатов;
* группировка чатов по типам занятий (лабораторная, практика и т.д.);
* просмотр вложенных документов в текущем чате;
* возможность отправки в чат электронных документов различных типов;
* отображение количества непрочитанных сообщений в каждом чате;
* управление сообщениями (удаление, редактирование, отложенная отправка(?));
* управление чатами администратором подсистемы управления чатами.

## Требования к серверной подсистеме

Также, перед тем как приступить к разработке сервиса обмена сообщениями, необходимо решить немаловажную задачу по определению требований к серверной подсистеме. Выделяют функциональные и нефункциональные требования. Суть функциональных требований сводится к выявлению поведения программы в определенных ситуациях, реакции на различные действия и входные данные, которые будут доступны пользователям и других функциональные возможности разрабатываемого сервиса. В свою очередь, нефункциональные требования определяют то, как должен работать программный продукт или система, и какими свойствами и характеристиками они должны обладать, так называемые атрибуты качества (производительность, надежность, качество и т.д.).

Среди функциональных требований можно выделить следующие:

* разделение чатов по типам занятий;
* работа с электронными документов различных типов;
* управление сообщениями (удаление, редактирование, отложенная отправка(?));

К нефункциональным требованиям можно отнести:

* построение архитектуры сервера таким образом, чтобы имела место максимальная автоматизация действий администратора подсистемы управления чатами;
* разработка удобного API, которое позволит упростить доступ к серверу (а также соответствующая документация на него);
* удобная архитектура, позволяющая с минимальными временными и трудовыми затратами производить действия с сервером (расширять, использовать, редактировать и т.д.).

# ОПРЕДЕЛЕНИЕ СПЕЦИФИКАЦИЙ К СЕРВИСУ ОБМЕНА СООБЩЕНИЯМИ

## Разработка архитектуры

Наиболее распространенной архитектурой для построения веб-приложений является клиент-серверная архитектура. Оба компонента представляют собой программное обеспечение, в большинстве случаев расположенные на разных вычислительных машинах, которые взаимодействуют между собой используя различные сетевые протоколы, но также они могут находиться и на одном устройстве. Клиентские программы являются потребителями информации и делают соответствующие запросы на сервер. Серверные программы, в свою очередь, являются поставщиками данной информации и предоставляют клиентам данные в соответствии с полученным запросом.

Применяемый для разработки серверной подсистемы веб-фреймворк Laravel предназначен для разработки приложений с использованием архитектурной модели MVC (рисунок 2.1.1).

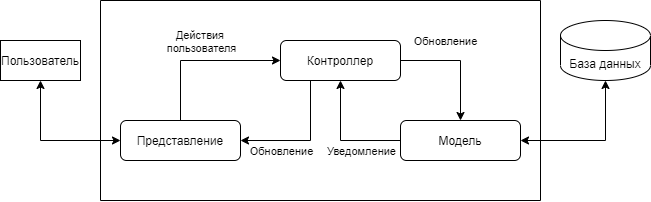


Рисунок 2.1.1 – Схема архитектуры MVC

Сама серверная подсистема сервиса обмена сообщениями будет представлять собой контроллер и модель, а в качестве представления будет выступать параллельно разрабатываемое мобильное приложение Взаимодействие между представлением (мобильным приложением) и контроллером и моделью (сервером) будет осуществляться с применением API. Для реализации данного взаимодействия в веб-фреймворке Laravel предусмотрены отдельные контроллеры – REST-контроллеры, представляющие собой дополнительный слой разделения логики обработки GET- и POST-запросов.

Представление (мобильное приложение), в зависимости от действий пользователя, делает запрос на сервер по соответствующему API (application programming interface, программный интерфейс приложения. Программный интерфейс приложения исходя из полученной команды выбирает необходимый контроллер. Контроллер использует связанную с ним модель (модели) для выполнения поставленной задачи и обновляет ее данные. Модель, в свою очередь, производит все необходимые действия над своими данными (производит запросы в базу данных, проверяет их на корректность и т.д.), уведомляет контроллер о результатах произведенных действий и возвращает измененные данные. Контроллер производит свои действия над полученными от модели данными при необходимости и передает их в качестве ответа представлению, которое посредством своего независимого функционала визуализирует полученные результаты.

Именно использование API и позволяет делать клиент независимо от сервера.

Таким образом, архитектура разрабатываемого сервиса обмена сообщениями будет соответствовать архитектурному паттерну MVC с применением API для обеспечения взаимодействия ее компонентов (рисунок 2.1.2)

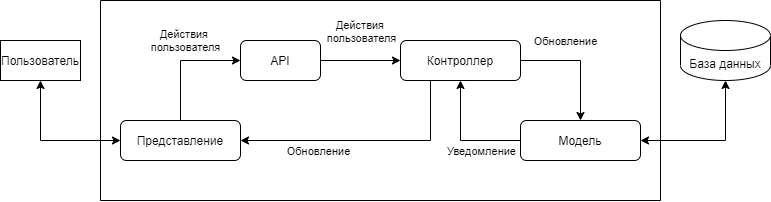


Рисунок 2.1.1 – Схема архитектуры MVC с использованием API

## Разработка функциональных спецификаций

Для упрощения процесса разработки и описания функциональных спецификаций сервиса обмена сообщениями можно воспользоваться диаграммой вариантов использования (use case diagram) UML. Данная диаграмма необходима для того, чтобы описать отношения между пользователями и возможностями моделируемой системы или части ее возможной функциональности («актерами» и «прецендентами», соответственно, в контексте языка UML).

Диаграмма всех вариантов использования разрабатываемого сервиса обмена сообщениями представлена на рисунке 2.2.1.

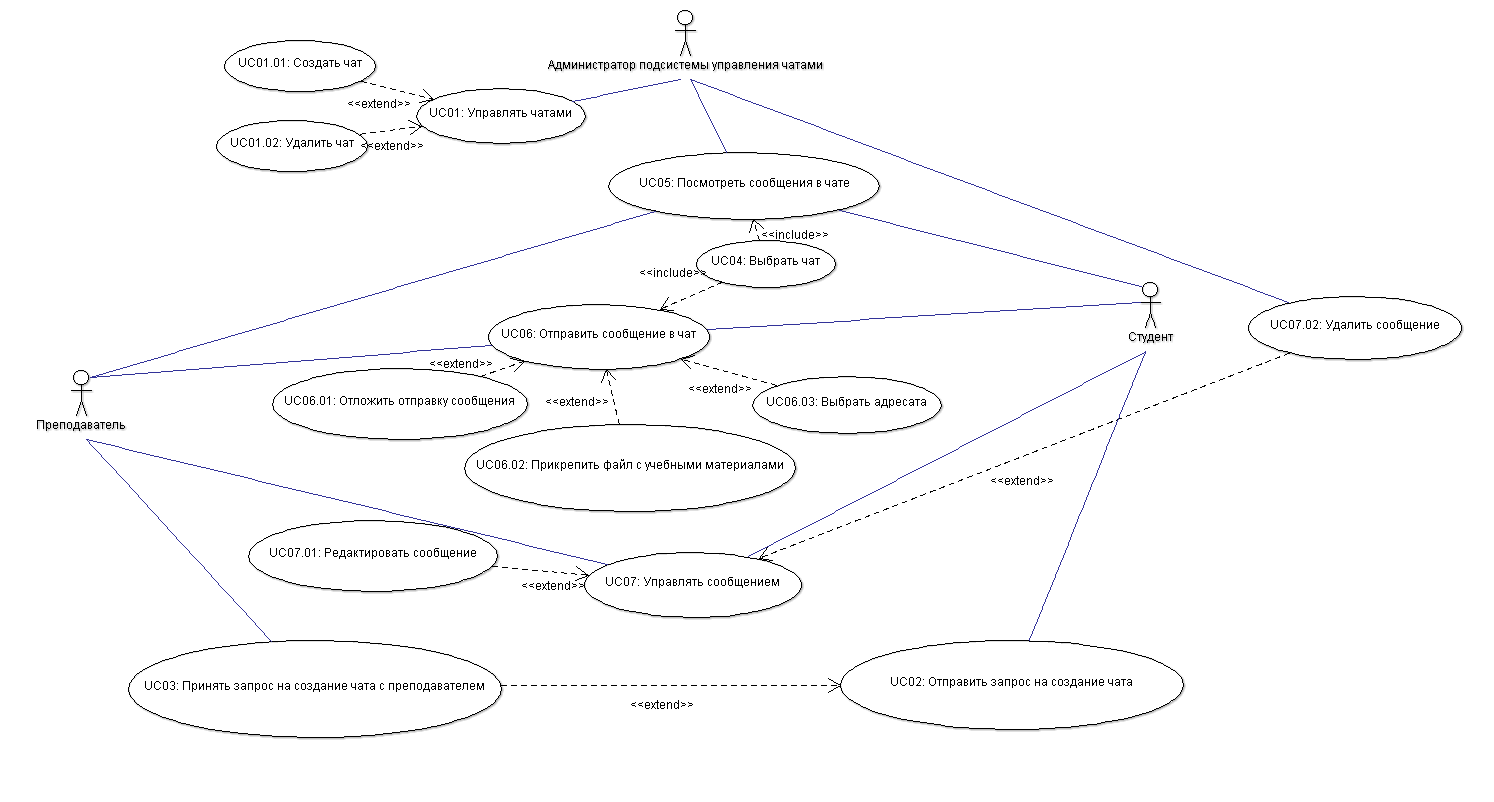


Рисунок 2.2.1 – Диаграмма вариантов использования приложения

Подробное описание каждого варианта использования представлена в таблицах 2.2.1 – 2.2.7.

Таблица 2.2.1 – UC01: Управлять чатами

|  |  |
| --- | --- |
| **Вариант использования** | **UC01: Управлять чатами** |
| Описание (основной поток) | 1. Система выводит список существующих чатов в режиме управления 2. Администратор выбирает чат для управления 3. Администратор выбирает функцию «Управлять чатом» 4. Система выводит сведения о чате из БД 5. Администратор добавляет / удаляет пользователей в чат и выбирает одно из действий:   5а: Сохранить изменения: система сохраняет изменения в БД, добавляет / удаляет чат в списках чатов пользователей. Возврат к п.1  5б: Отмена: выводится сообщение для подтверждения действия; при подтверждении изменения не сохраняются. Возврат к п.1 |
| Триггер | Выбрана функция «Управлять чатами» |
| Предусловия | PRE-1: Авторизоваться в системе в роли администратора подсистемы управления чатами |
| Постусловия | POST-1: Измененный список чатов сохранен в БД  POST-2: В списке чатов пользователя отображаются актуальные чаты |
| Альтернативные потоки | A1: Удаление чата:   1. Шаг 1 основного потока 2. Шаг 2 основного потока 3. Администратор выбирает функцию «Удалить чат» 4. Выводится сообщение для подтверждения действия 5. Администратор выбирает одно из действий:   5а: Удалить чат: система удаляет чат из списков чатов пользователей, очищает сведения о выбранном чате в БД и сохраняет изменения в БД. Возврат к п.1  5б: Отмена: выводится сообщение для подтверждения действия; при подтверждении изменения не сохраняются. Возврат к п.1  А2: Создание чата:   1. Шаг 1 основного потока 2. Администратор выбирает функцию «Создать чат» 3. Система выводит список пользователей из БД для создания чата 4. Администратор выбирает пользователей из списка и создает новый чат 5. Система добавляет чат в список чатов, добавляет чат в чаты выбранных пользователей, сохраняет изменения в БД. Возврат к п.1 |
| Исключения | Нет |

Таблица 2.2.2 **–** UC02: Отправить запрос на создание чата

|  |  |
| --- | --- |
| **Вариант использования** | **UC02: Отправить запрос на создание чата** |
| Описание (основной поток) | 1. Система отображает список чатов студента 2. Студент выбирает функцию «Создать чат с преподавателем» 3. Система выводит список преподавателей из БД 4. Студент выбирает преподавателя из списка для создания чата 5. Система отправляет преподавателю запрос на создание чата со студентом. Возврат к п.1 |
| Триггер | Выбрана функция «Отправить запрос на создание чата» |
| Предусловия | PRE-1: Авторизоваться в системе в роли студента |
| Постусловия | POST-1: В списке чатов преподавателя отображается запрос на создание чата со студентом |
| Альтернативные потоки | Нет |
| Исключения | Нет |

Таблица 2.2.3 – UC03: Принять запрос на создание чата с преподавателем

|  |  |
| --- | --- |
| **Вариант использования** | **UC03: Принять запрос на создание чата с преподавателем** |
| Описание (основной поток) | 1. Преподаватель выбирает раздел «Запросы на создание чатов» 2. Система отображает существующие запросы 3. Преподаватель отвечает на запрос 4. Система создает чат, добавляет чат в список чатов, добавляет чат в чаты выбранных пользователей, сохраняет изменения в БД. Возврат к п.1 |
| Триггер | Выбрана функция «Принять запрос на создание чата с преподавателем» |
| Предусловия | PRE-1: Авторизоваться в системе в роли преподавателя |
| Постусловия | POST-1: Измененный список чатов сохранен в БД  POST-2: В списке чатов пользователя отображаются актуальные чаты |
| Альтернативные потоки | Нет |
| Исключения | Нет |

Таблица 2.2.4 – UC04: Выбрать чат

|  |  |
| --- | --- |
| **Вариант использования** | **UC04: Выбрать чат** |
| Описание (основной поток) | 1. Система отображает список чатов пользователя 2. Пользователь выбирает чат для работы |
| Триггер | Выбрана функция «Выбрать чат» |
| Предусловия | PRE-1: Авторизоваться в системе |
| Постусловия | POST-1: Отображение материалов выбранного чата на экран |
| Альтернативные потоки | Нет |
| Исключения | Нет |

Таблица 2.2.5 – UC05: Посмотреть сообщения в чате

|  |  |
| --- | --- |
| **Вариант использования** | **UC05: Посмотреть сообщения в чате** |
| Описание (основной поток) | 1. Система отображает сообщения чата с разделением на прочитанные и непрочитанные 2. Пользователей просматривает сообщения чата |
| Триггер | Выбрана функция «Посмотреть сообщения в чате» |
| Предусловия | PRE-1: Авторизоваться в системе  PRE-2: Выбрать чат |
| Постусловия | POST-1: Непрочитанные сообщения отображаются как прочитанные |
| Альтернативные потоки | Нет |
| Исключения | Нет |

Таблица 2.2.6 – UC06: Отправить сообщение в чат

|  |  |
| --- | --- |
| **Вариант использования** | **UC06: Отправить сообщение в чат** |
| Описание (основной поток) | 1. Пользователь выбирает функцию «Написать сообщение» 2. Система отображает окно ввода сообщения 3. Пользователь вводит текст сообщения и выбирает оно из действий:   3а: Отправить сообщение: система отправляет сообщение в чат, сохраняет сообщение в БД. Возврат к чату  3б: Отмена: система очищает окно ввода сообщения. Изменения не сохраняются. Возврат к чату |
| Триггер | Выбрана функция «Отправить сообщение в чат» |
| Предусловия | PRE-1: Авторизоваться в системе  PRE-2: Выбрать чат |
| Постусловия | POST-1: Обновленный чат сохранен в БД  POST-2: В чате пользователям отображаются непрочитанные сообщения |
| Альтернативные потоки | А1: Отложить отправку сообщения:   1. Шаг 1 основного потока 2. Шаг 2 основного потока 3. Шаг 3 основного потока (до выбора действия) 4. Пользователь выбирает отложенную отправку сообщения. 5. Система отображает поле для ввода даты и времени отправки. 6. Пользователь вводит дату и время и выбирает одно из действий:   6а: Отправить сообщение: система отправляет сообщение в чат в указанную дату и время, сохраняет сообщение в БД. Возврат к чату  6б: Отмена: система очищает окно ввода сообщения и поле ввода даты и времени. Изменения не сохраняются. Возврат к чату  А2: Прикрепить файл с учебными материалами   1. Шаг 1 основного потока 2. Шаг 2 основного потока 3. Шаг 3 основного потока (до выбора действия) 4. Пользователь выбирает функцию «Прикрепить файл с учебными материалами» 5. Система отображает список возможных типов файлов для вложения, открывает файловую систему устройства для выбора файла 6. Пользователь выбирает файл и выбирает одно из действий:   6а: Прикрепить файл: система прикрепляет файл к отправляемому сообщению, сохраняет файл в БД, закрывает файловую систему устройства. Возврат к п.3 основного потока  6б: Отмена: система закрывает файловую систему устройства, файл не сохраняется в БД. Возврат к п.3 основного потока  А3: Выбрать адресата:   1. Шаг 1 основного потока 2. Шаг 2 основного потока 3. Шаг 3 основного потока (до выбора действия) 4. Пользователь выбирает функцию «Отправить сообщение адресату» 5. Система отображает список пользователей чата 6. Пользователь выбирает адресата и одно из действий:   6а: Отправить сообщение: система отправляет сообщение в выбранному пользователю, сохраняет сообщение в БД. Возврат к чату  3б: Отмена: система очищает окно ввода сообщения и выбора адресата. Изменения не сохраняются. Возврат к чату |
| Исключения | Нет |

Таблица 2.2.7 – UC07: Управлять сообщением

|  |  |
| --- | --- |
| **Вариант использования** | **UC07: Управлять сообщением** |
| Описание (основной поток) | 1. Пользователь выбирает функцию «Редактировать сообщение» 2. Система отображает сообщение в окне редактирования 3. Пользователь вводит изменения в сообщение и выбирает одно из действий:   3а: Сохранить изменения: система сохраняет изменения в сообщении в БД. Возврат к чату  3б: Отмена: выводится сообщение для подтверждения действия; при подтверждении изменения не сохраняются. Возврат к чату |
| Триггер | Выбрана функция «Управлять сообщением» |
| Предусловия | PRE-1: Авторизоваться в системе  PRE-2: Выбрать чат  PRE-3: Выбрать сообщение |
| Постусловия | POST-1: Отредактированный чат сохранен в БД  POST-2: В чате пользователям отображаются актуальные сообщения |
| Альтернативные потоки | А1: Удалить сообщение:   1. Пользователь выбирает функцию «Удалить сообщение» 2. Выводится сообщение для подтверждения действия 3. Пользователь выбирает одно из действий:   3а: Удалить сообщение: система отправляет запрос на удаление сообщения из БД администратору, удаляет сообщение из чата, обновляет чат. Возврат к чату  3б: Отмена: выводится сообщение для подтверждения действия; при подтверждении изменения не сохраняются. Возврат к чату |
| Исключения | Нет |

## Разработка информационных спецификаций серверной подсистемы сервиса обмена сообщениями

Модели данных, формат обмена данными(json), какие данные будут приходить/уходить?

## Разработка поведенческих спецификаций

Диаграммы состояний?

## Разработка программного интерфейса приложения(АПИ)

Применительно к разрабатываемому сервису обмена сообщениями, архитектурный паттерн MVC будет дополнен применением API ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) Application Programming Interface, программный интерфейс приложения). API, по своей сути, является неким «контрактом», который заключается между разными приложениями или между частями одного и того же приложения. Применительно к концепции клиент-сервер, API чаще всего реализуется на стороне сервера, а клиент уже обращается к нему, вызывая соответствующие функции, то есть делает запросы на сервер.

Данный контракт определяет то, каким образом стороны контракта могут взаимодействовать друг с другом, то есть:

* какие функции доступны для использования клиентом и их описание;
* какие данные принимает сервер от клиента и в каком формате;
* какие данные отдает сервер клиенту и в каком формате;
* ??

Существует два вида разработки API: CodeFirst и ContractFirst. Первый подход – CodeFirst – состоит в том, что сначала происходит разработка кода сервера и только после того, как его функционал готов (в том числе и API, по которому будет возможно к нему обратиться), формируется документация по данному API.

Преимущества данного подхода:

* минимальные усилия по генерации контракта, поскольку API уже написано и его остается только задокументировать (на данный момент существуют средства, позволяющие автоматизировать этот процесс);
* актуальность контракта, поскольку он генерируется из кода.

Недостатки данного подхода:

* + отсутствие возможности параллельной разработки. Из-за того, что сначала необходимо разработать сервис, а затем сгенерировать контракт, разработка клиентской части откладывается на неопределенное время до получения контракта. Невозможно нормально разработать потребителя не имея договора.
  + нет общего направления работы команд. Этот недостаток вытекает из предыдущего пункта и означает, что до момента получения контракта, заинтересованные стороны, разрабатывающие клиентскую и серверную части, могут по-разному представлять себе как конечные результаты работы, так и функционал приложения.

Суть второго подхода – ContractFirst – сводится к тому, что сначала определяется контракт, по которому будут общаться стороны, и только после этого команды могут приступать к разработке клиента и сервера соответственно.

Преимущества данного подхода:

* возможность параллельной разработки. Исходя из того, что контракт разработан, команды могут спокойно приступать к разработке своих частей;
* точное представление об ожиданиях обеих сторон. Если функционал обеих сторон разрабатывается в разных темпах, то для проведения его тестирования нет необходимости ждать другую сторону. Поскольку контракт уже готов, то всем известно, что и в каком виде отправлять и получать, поэтому можно использовать заглушки.

Недостатки данного подхода:

* наличие дополнительных затрат в начале разработки. Необходимо потратить большое количество времени и сил, чтобы определиться с контрактом, чтобы он был максимально четким и соответствовал поставленным задачам. При этом, он не должен будет часто меняться.
* сложность обновления контракта. Изменение контракта влияет на все заинтересованные стороны. Поэтому, необходимо иметь механизм, который поможет упростить процесс оповещения и внедрения изменений разработчиками.

В данной работе будет использоваться второй подход (контракт в первую очередь), поскольку разработка клиентской и серверной частей ведется параллельно разными разработчиками.

Также, будет применяться архитектурный стиль REST API. Его суть состоит в следующем: клиент вызывает удаленную функцию, которая представляет собой обычный HTTP-запрос (GET или POST), передавая данные в качестве параметров запроса. Каждому ресурсу на сервере назначается свой идентификатор URI (универсальный идентификатор ресурса). Например, если мы хотим получить всех пользователей – GET/users. Также, в запросе указывается метод, описывающий действие, которое необходимо произвести с данными:

* GET – получить информацию о ресурсе;
* POST – создать новый ресурс;
* PUT(UPDATE) – обновить существующий ресурс;
* DELETE – удалить ресурс.

Кроме того, немаловажной частью являются и метаданные, возвращаемые в ответе сервером. К ним могут относиться: сообщения об ошибках, различные свойства передаваемого ресурса (например, общее число записей) и т.д.

Для каждой функции API, которая будет у сервиса, необходимо определить метод работы с данными, URI, данные, получаемые при запросе и отправляемые в ответе, сам формат этих данных (в каком виде), метаданные.

При проектировании API, в первую очередь, необходимо определиться с ресурсами, которые будет принимать и отдавать сервер. Для нашей задачи необходимо определить структуру следующих компонентов:

* пользователь;
* чат;
* сообщение;
* метаданные о документе;
* документ

На рисунке 2.5.1 представлена разработанная схема для ресурса пользователя.

Рисунок 2.5.1 – Схема ресурса «Пользователь»

Следующим шагом является конструирование URL, для каждого из которых определяется и описывается метод работы (тип метода, его описание, тело запроса и т.д.), а для каждого метода также описывается код и структура получаемого ответа. Основными URL будут являться:

* /api/v1/chats – получение информации обо всех чатах, доступных текущему пользователю, а также добавление нового чата;
* /api/v1/chats/{chatId} – получение информации о конкретном чате, ее изменение и удаление данного чата;
* /api/v1/chats/{chatId}/users – получение информации обо всех пользователях чата, а также добавление нового пользователя в чат;
* /api/v1/chats/{chatId}/documents – получение всех документов, вложенных в чат, а также добавление новой метаданных о документе;
* /api/v1/chats/{chatId}/documents/{documentId} – получение метаданных о конкретном документе чата, а также ее изменение;
* /api/v1/chats/{chatId}/documents/{documentId}/data – получение конкретного документа в чате;
* /api/v1/chats/{chatId}/users/{userId}/documents – получение всех документов пользователя в чате.

На рисунках 2.5.2 – 2.5.5 представлена структура и описание URL на примере получения списка чатов (/api/v1/chats). Полная информация о данной функции, а также об остальных представлена в Приложении Б.

Рисунки АПИ

# ПРОЕКТИРОВАНИЕ СЕРВЕРНОЙ ПОДСИСТЕМЫ СЕРВИСА ОБМЕНА СООБЩЕНИЯМИ

## Проектирование структуры программного обеспечения

Какие Интерфейсы, классы

## Проектирование основных компонентов программного обеспечения

Описание контроллеров (3.2.1, 3.2.2, …)

## Проектирование API

## Проектирование БД

Файлы хранить на сервере, а в бд хранить ссылки (пути) на них. Путь делать из названия чата и имени файла. Сделать «триггер»: если удаляется файл, то удалять путь и наоборот.

https://ru.stackoverflow.com/questions/207738/Как-хранить-фотографии-в-БД

## Разработка ..

# РЕАЛИЗАЦИЯ, ТЕСТИРОВАНИЕ И ОТЛАДКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

## Реализация алгоритма (контроллеры?)…

## Реализация АПИ?…

## Тестирование и отладка программного обеспечения

# ЗАКЛЮЧЕНИЕ

В ходе выполнения данной работы были решены следующие задачи:

1. Рассмотрены общие сведения серверной подсистемы сервиса обмена сообщениями и проанализирована задача … .
2. Выполнен анализ требований и определение спецификаций программного обеспечения серверной подсистемы сервиса обмена сообщениями.
3. Спроектирована структура, алгоритмы и интерфейс серверной подсистемы сервиса обмена сообщениями.
4. Реализованы компоненты и интерфейс.

Таким образом, поставленные задачи выполнены, а цель работы достигнута.

# СПИСОК ЛИТЕРАТУРЫ

# ПРИЛОЖЕНИЕ А

**(обязательное)**

**ФРАГМЕНТЫ ИСХОДНОГО ТЕКСТА ПРОГРАММЫ**

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ   
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«ОРЛОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИМЕНИ И.С. ТУРГЕНЕВА»

Удостоверяющий лист № 160662

К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ

на демонстрационный материал, представленный в электронном виде

Студента Шорина Владислава Дмитриевича шифр 171406

Институт приборостроения, автоматизации и информационных технологий

Кафедра программной инженерии

Направление 09.03.04 Программная инженерия

Направленность (профиль) Промышленная разработка программного обеспечения

Наименование документа: Демонстрационные плакаты к выпускной квалификационной работе

Документ разработал:

Студент Шорин В.Д. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Документ согласован:

Руководитель Фролов А.И. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Нормоконтроль Ужаринский А.Ю. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Документ утвержден:

Зав. кафедрой Фролов А.И. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Орел 2020

ИНФОРМАЦИОННО-ПОИСКОВАЯ ХАРАКТЕРИСТИКА  
ДОКУМЕНТА НА ЭЛЕКТРОННОМ НОСИТЕЛЕ

|  |  |  |
| --- | --- | --- |
| Наименование | | Характеристики документа на электронном носителе |
| группы атрибутов | Атрибута |
| 1. Описание документа | Обозначение документа (идентификатор(ы) файла(ов)) | \Презентация\_Шорин.pptx |
| Наименование документа | Демонстрационные плакаты к выпускной квалификационной работе |
| Класс документа | ЕСКД |
| Вид документа | Оригинал документа на электронном носителе |
| Аннотация | Демонстрационный материал, отображающий основные этапы выполнения выпускной квалификационной работы |
| Использование документа | Операционная система Windows 7, Microsoft PowerPoint 2007 |
| 2. Даты и время | Дата и время копирования документа | 11.06.2020 |
| Дата создания документа | 11.06.2020 |
| Дата утверждения документа | 26.06.2020 |
| 3. Создатели | Автор | Шорин В.Д. |
| Изготовитель | Шорин В.Д. |
| 4. Внешние ссылки | Ссылки на другие документы | Удостоверяющий лист  № 160662 |
| 5. Защита | Санкционирование | ОГУ имени И.С. Тургенева |
| Классификация защиты | По законодательству РФ |
| 6. Характеристики содержания | Объем информации документа | 73 824 090 Б |

|  |  |  |
| --- | --- | --- |
| 7. Структура документа(ов) | Наименование плаката (слайда) №1 | Титульный лист |
| Наименование плаката (слайда) №2 | Назначение и архитектура  имитатора закабинного пространства |
| Наименование плаката (слайда) №3 | Цель и задачи работы |
| Наименование плаката (слайда) №4 | Анализ альтернативных решений |
| Наименование плаката (слайда) №5 | Анализ методов моделирования ландшафтов |
| Наименование плаката (слайда) №6 | Функциональная модель программного обеспечения |
| Наименование плаката (слайда) №7 | Диаграмма состояний интерфейса пользователя |
| Наименование плаката (слайда) №8 | Структура пакетов подсистемы |
| Наименование плаката (слайда) №9 | Структура базовых классов |
| Наименование плаката (слайда) №10 | Проектирование алгоритмов: математический аппарат вычисления угла наклона поверхности |
| Наименование плаката (слайда) №11 | Проектирование алгоритмов: общая схема алгоритма генерации растительности |
| Наименование плаката (слайда) №12 | Примеры работы подсистемы |