Титульный лист

Содержание

ВВЕДЕНИЕ

Все программы, написанные на языке программирования Java, можно разделить на следующие группы:

1. Автономные (самостоятельные) приложения;
2. Апплеты;
3. Сервлеты.

Для запуска автономных приложений требуется предварительная установка виртуальной машины Java (JVM) — программы, обрабатывающей байт-код и передающей инструкции, полученные на его основе, оборудованию. Апплеты — приложения Java, выполнение которых осуществляется веб-браузером, в который встроена JVM (все современные браузеры являются java-совместимыми). Сервлеты, в отличие от апплетов, выполняются на стороне web-сервера. Служат они для формирования динамических html-страниц.

Для упрощения процессов хранения и передачи скомпилированного кода java-программ, он упаковывается в специальные java-архивы, имеющие следующие расширения: \*.jar, \*.war, \*.ear.

Архивы с расширением \*.jar представляют собой библиотеки, предназначенные для хранения классов. Такие библиотеки могут быть подключены и использованы в сторонних java-приложениях. В war-архивы упаковываются веб-приложения, а в ear-архивы, в свою очередь, — специальные бизнес-модули. Ear-архивы обеспечивают одновременное и согласованное развертывание этих модулей на сервере приложений.

Java-приложения, упакованные в архив с расширением \*.jar, могут быть запущены как на локальном, так и на удаленном компьютере. Возможность удаленного запуска достигается использованием таких сетевых протоколов как, например, SSH.

SSH — сетевой протокол прикладного уровня, главной задачей которого является предоставление возможности удаленного управления операционной системой и туннелирование TCP-соединений. Протокол SSH пришел на смену другому сетевому протоколу — Telnet, главным недостатком которого является то, что передаваемые данные никак не кодируются. В этом случае становится возможной кража информации и дальнейшее ее использование в злонамеренных целях.

Протокол SSH, в свою очередь, разрабатывался для предоставления безопасности передаваемых данных путем реализации стойкого алгоритма шифрования, надежной системы аутентификации пользователя и сервера, предоставлением системы контроля целостности передаваемых данных, а также инкапсуляцией приложений, работающих на основе протокола TCP, для установления безопасных туннелей.

Однако, несмотря на все свои достоинства, протокол SSH не решает всех проблем сетевой безопасности. Он лишь фокусирует свое внимание на обеспечении безопасной работы таких приложений, как эмуляторы терминала (однако этими лишь функциями протокол не ограничивается). Использование реализаций протокола SSH на серверах и в клиентских приложениях помогает защитить данные лишь в процессе передачи. Протокол SSH никоим образом не является заменой брандмауэров, систем обнаружения вторжений, сетевых сканеров, систем аутентификации и других инструментов, позволяющих защитить информационные системы и сети от атак.

Для запуска java-приложений может быть использован интерфейс командной строки (CLI - Command Line Interface). Запускаются Java-приложения с помощью инструкции java. В этой инструкции могут быть указаны параметры, которые необходимо передать программе. Точкой входа в java-приложение является метод public static void main(String[] args). Именно в этот метод осуществляется передача параметров, указанных в инструкции java.

1 ПОСТАНОВКА ЗАДАЧИ

* 1. Описательная постановка задачи

В современном мире при разработке программного обеспечения у разработчика нередко возникает необходимость использовать для своих целей сторонние программы и утилиты, в том числе написанные на объектно-ориентированном языке программирования Java. Но не всегда данные программы располагаются на локальной машине. И разработчик в данной ситуации вынужден запускать их на некотором удаленном ресурсе. При этом могут возникать различные проблемы, так как для запуска java-программы необходимо точно знать, где она располагается, и какие входные параметры она может принимать. Таким образом, на запуск программ может затрачиваться большое количество времени, что существенно тормозит процесс разработки программного обеспечения.

Целью данной работы является решение описанной выше проблемы, а именно — разработка библиотеки, позволяющей осуществлять удаленный запуск java-программ с интерфейсом командной строки, а также получать результат запущенной программы для дальнейшего его использования в своих целях.

Работа проводится в рамках задачи по моделированию галактик, для решения которой требуется запуск расчетных задач, располагающихся на удаленных ресурсах.

* 1. Формальная постановка задачи

Требуется разработать клиентское десктопное приложение, позволяющее пользователям подключаться к удаленным ресурсам, регистрировать приложения, хранящиеся на них. Также необходимо предоставить пользователю возможность редактировать информацию об уже зарегистрированных приложениях, удалять их и запускать на удаленном ресурсе.

Регистрация нового приложения предполагает ввод имени приложения, его описания и пути к директории, в которой располагается данное приложение.

При редактировании информации о приложении требуется предоставить возможность изменять введенные ранее данные, а также возможность добавлять, удалять и изменять параметры, которые приложение может принимать при запуске. Параметрам при добавлении необходимо задать имя, описание и указать, является ли добавляемый параметр флагом, то есть должен ли он принимать какое-то значение.

При запуске приложения пользователю должен представляться список принимаемых ею параметров, включающий в себя имя параметров и их описание. Также должна быть предоставлена возможность выбора необходимых параметров из списка представленных.

Для обеспечения более удобного взаимодействия пользователя с системой требуется разработать пользовательский графический интерфейс.

* 1. Описание декомпозиции задачи

Для достижения поставленной цели требуется решить следующие подзадачи:

1. Анализ требований к системе.
2. Анализ и структурирование исходной информации.
3. Анализ функций системы.
4. Разработка и согласование с научным руководителем технического задания.
5. Проектирование системы.
6. Разработка алгоритмов обработки информации.
7. Проектирование пользовательского интерфейса.
8. Реализация компонентов системы.
9. Разработка методов тестирования написанного приложения.
   1. Аналитический обзор существующих методов решения рассматриваемой проблемы

Решение поставленной задачи предполагает удаленное управление компьютером. Эту возможность позволяют обеспечить специальные сетевые протоколы. Например: RDP, Telnet, TSL. Однако наибольшую популярность приобрел сетевой протокол SSH.

Обусловлено это тем, что данный протокол не только шифрует весь передаваемый трафик, но также позволяет безопасно передавать в незащищённой среде практически любой другой [сетевой протокол](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%82%D0%B5%D0%B2%D0%BE%D0%B9_%D0%BF%D1%80%D0%BE%D1%82%D0%BE%D0%BA%D0%BE%D0%BB). Таким образом, можно не только удалённо работать на компьютере через [командную оболочку](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%B0%D0%BD%D0%B4%D0%BD%D0%B0%D1%8F_%D0%BE%D0%B1%D0%BE%D0%BB%D0%BE%D1%87%D0%BA%D0%B0), но и передавать по шифрованному каналу звуковой поток или видео.

Наиболее популярной программой, позволяющей установить удаленное соединение по протоколу SSH, является Putty. Однако данная утилита не позволяет решить поставленной задачи, так как она представляет собой механизм, обеспечивающий возможность удаленного администрирования, в то время как поставленная задача является более узкой и предполагает выполнение только одной команды на удаленном ресурсе: java –jar <путь>. Кроме того, Putty не позволяет хранить информацию о расположенных на удаленных ресурсах java-приложениях.

В настоящее время в лаборатории космических исследований в НИИ Физики ведется работа по создании вычислительного кластера для выполнения расчетных задач по моделированию галактических объектов. В качестве базовой платформы выбрана - платформа Java с использованием технологий JCUDA, JFreeChart и пр. Отличительная особенность выполнения вычислительных задач при этом заключается в длительности их исполнения, которое может быть в диапазоне от нескольких часов до нескольких недель, а также в больших объемах начальных и промежуточных данных составляющих сотни Гб и выше. Наиболее удобным способом запуска и контроля выполнения таких задач является режим удаленного доступа, для которого, как правило, используется shttp, shh.

Таким образом, задача разработки инструментального средства для удаленного запуска java-программ является актуальной и не имеет широко распространенных решений.

2 ПРОЕКТИРОВАНИЕ СИСТЕМЫ

Программный продукт проектируется и реализуется на основании технического задания, согласованного с научным руководителем. Полный текст технического задания представлен в приложении А.

2.1 Выбор инструментальных средств

Для построения UML диаграмм, позволяющих визуально представить функции и структуру проекта, используется приложение Umbrello.

Так как предполагается, что конечный продукт будет выполняться на операционных системах как семейства Windows, так и Unix, для его реализации было решено использовать объектно-ориентированный язык программирования Java. Данный язык был разработан компанией Sun Microsystems. Отличительной особенностью Java является то, что все программы, написанные на нем, транслируются в специальный байт-код, который в дальнейшем выполняется JVM — программой, обрабатывающей байтовый код и передающей инструкции, полученные на его основе, оборудованию. Байт-код не зависит от операционной системы и оборудования, что позволяет запускать одни и те же java-приложения на различных устройствах, для которых написана JVM. Другими словами, приложения являются кроссплатформенными.

В качестве текстового редактора была выбрана интегрированная среда разработки NetBeans. Данная среда поддерживает рефакторинг кода, профилирование, выделение синтаксических конструкций цветом, автодополнение набираемых конструкций на лету и множество предопределённых шаблонов кода. Все предоставляемые данной средой возможности существенно ускоряют процесс написания кода.

Для автоматизации сборки проекта решено использовать систему Maven, так как она является простой в использовании (в отличие от Ant). Неотъемлемой частью системы Maven является pom.xml файл, в котором указывается список подключаемых сторонних библиотек, а также параметры выполнения процессов компиляции, сборки проекта, тестирования и архивирования. Данный сборщик проектов поставляется в стандартном наборе компонентов NetBeans.

Для выполнения логирования решено использовать специальную библиотеку log4j, позволяющую выводить сообщения различной степени важности. Все параметры конфигурации процесса логирования хранятся в специальном файле log4j.properties.

Для разработки графического пользовательского интерфейса решено использовать платформу JavaFx. Данная платформа позволяет обеспечить возможность создания как настольных приложений, запускаемых непосредственно из-под операционных систем, так и для интернет-приложений (RIA), работающих в браузерах.

В качестве вспомогательной утилиты при разработке интерфейса используется SceneBuilder. Она позволяет проектировать формы интерфейса посредством простого перетаскивания необходимых компонентов в рабочую область.

Для тестирования приложения решено использовать JUnit — библиотеку для модульного тестирования программного обеспечения на языке Java.

Для хранения данных решено использовать СУБД MySql. Внутренний механизм многопоточности данной СУБД обеспечивает ее высокое быстродействие. Также достоинствами MySql являются ее надежность и безопасность.

2.2 Описание вариантов использования приложения

Все возможные действия пользователя, а также связи между ними представлены на рис.1. Главными действующими лицами разрабатываемого приложения являются пользователь и система.

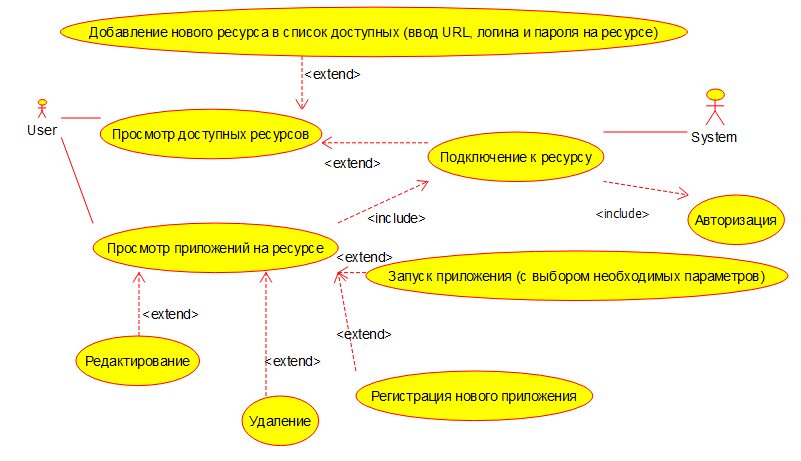


Рисунок 1 — Диаграмма вариантов использования.

Запустив приложение, пользователь может просмотреть список зарегистрированных в системе удаленных ресурсов и выбрать из данного списка тот, к которому необходимо подключиться. Далее пользователь может просмотреть список уже зарегистрированных java-приложений на данном ресурсе и запустить необходимое. Также пользователю предоставлена возможность регистрировать новые приложения, удалять зарегистрированные и редактировать информацию о них и о принимаемых ими параметрах.

Актер System при выборе пользователем требуемого ресурса проводит диагностику данного ресурса на наличие с ним соединения. Также он обеспечивает запуск приложения на выбранном ресурсе.

При авторизации пользователя осуществляется проверка, имеется ли информация о пользователе с таким логином в базе данных. Если нет, устанавливается соединение с удаленным ресурсом, и проверка логина и пароля осуществляется непосредственно на нем. Если логин и пароль введены правильно, они добавляются в базу данных. Таким образом, если пользователь раньше проводил подключение к выбранному ресурсу, соединение с ним при авторизации устанавливаться не будет. Данный подход позволяет сократить время, которое требуется системе, чтобы проверить правильность введенных данных, тем самым обеспечить более комфортное использование разрабатываемого приложения.

2.3 Описание модели данных

На основании системного анализа были выявлены стержневые и характеристические сущности, а также была спроектирована бизнес-модель данных, представленная на рис.2. Все классы связаны между собой отношением «композиция». Это говорит о том, что при удалении экземпляра родительского класса, происходит каскадное удаление всех экземпляров дочерних классов.

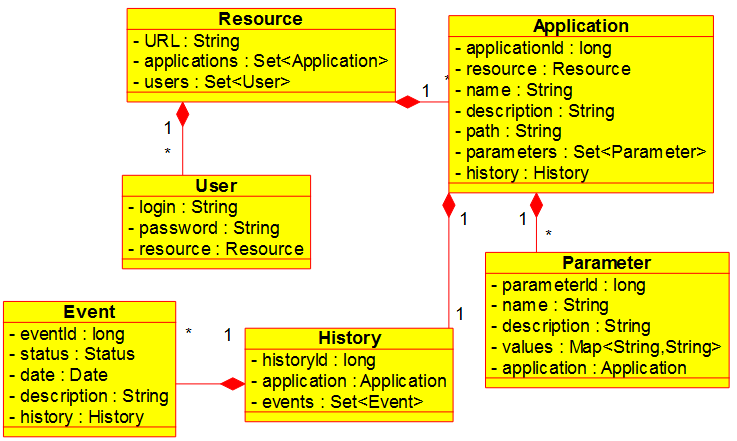


Рисунок 2 — Диаграмма классов.

2.4 Проектирование пользовательского интерфейса

2.5 Методика тестирования

Для тестирования приложения используются модули JUnit. Тестирование приложения проводится в соответствии с таблицей 1.

Таблица 1 — Порядок тестирования разработанного приложения.

|  |  |  |  |
| --- | --- | --- | --- |
| Пункт прогр. испыт. | Наименование показателя | Пункт требований ТЗ | Методика проверки |
| 1 | Просмотр списка зарегистрированных  удаленных ресурсов | 4.1 | Проверка методов, обеспечивающих извлечение информации о ресурсах из БД |
| 2 | Регистрация нового ресурса | 4.2 | Проверка методов, обеспечивающих сохранение информации о регистрируемом ресурсе в БД |
| 3 | Авторизация пользователей | 4.3 | Проверка доступности функций, требующих авторизации |
| 4 | Просмотр списка зарегистрированных приложений | 4.4 | Проверка доступности списка зарегистрированных приложений |
| 5 | Регистрация нового приложения | 4.5 | Проверка методов, обеспечивающих сохранение информации о регистрируемом приложении |
| 6 | Удаление приложения из списка представленных | 4.6 | Проверка методов, обеспечивающих удаление информации о приложении |
| 7 | Редактирование информации о зарегистрированном приложении | 4.7 | Проверка методов, обеспечивающих обновление ранее введенной информации |
| 8 | Добавление параметра | 4.8.1 | Проверка методов, обеспечивающих добавление нового параметра в список возможных принимаемых приложением параметров |
| 9 | Редактирование параметра | 4.8.2 | Проверка методов, обеспечивающих обновление введенной ранее информации о параметре |
| 10 | Удаление параметра | 4.8.3 | Проверка методов, обеспечивающих удаление параметра из списка принимаемых приложением параметров |
| 11 | Удаленный запуск выбранного приложения | 4.9 | Проверка доступности удаленного ресурса, попытка подключения к несуществующему ресурсу, обработка полученного результата |
| 12 | Предоставление списка входных параметров запускаемого приложения | 4.9.1 | Проверка доступности списка возможных параметров |

3 РЕАЛИЗАЦИЯ ПРОГРАММНОГО ПРОДУКТА

3.1 Описание реализаций основных функций приложения.

Разрабатываемый проект состоит из двух модулей: core и javafx. Модуль core является ядром разрабатываемого приложения, в нем реализованы все методы, позволяющие обеспечить требуемые в техническом задании функциональные возможности приложения.

Модуль javafx реализует логику работы пользовательского интерфейса, а также отвечает за его отрисовку. Для обработки данных, полученных от пользователя во время его работы с приложением, вызываются соответствующие методы, реализованные в модуле core.

Все классы, содержащиеся в модуле core, разбиты по пакетам: ru.igorek.core.model, ru.igorek.core.model, ru.igorek.core.utils. В папке src/main/resources хранятся конфигурационные файлы.

В пакете ru.igorek.core.model хранятся классы, реализующие модель данных, представленную на рис.2. Все эти классы являются POJO-бинами, то есть они не имплементируют никаких служебных интерфейсов, имеют конструктор по умолчанию, все атрибуты имеют область видимости private, и для каждого из них реализованы геттеры и сеттеры.

Основными атрибутами класса Resource являются URL типа String, applications типа List<Application> и users типы Set<ResourceUser>. Атрибут URL предназначен для хранения url ресурса, а также порта, слушателем которого является данный ресурс. Атрибут applications хранит список зарегистрированных приложений, а атрибут users — множество пользователей, имеющих доступ к нему.

Класс ResourceUser имеет следующие атрибуты: login типа String и password типа String. Они предназначены для хранения логина и пароля пользователя соответственно.

Основными атрибутами класса Application являются name типа String, description типа String, path типа String, parameters типа List<Parameter>. Атрибут name хранит имя java-приложения. Имя приложения может быть задано пользователем произвольно. Атрибут description хранит описание данного приложения, включающее в себя описание задач, для решения которых оно предназначено. Атрибут path хранит путь к директории на удаленном ресурсе, в которой располагается приложение. Parameters содержит список параметров, которые приложение может принимать при запуске.

Класс Parameter имеет такие атрибуты, как parameterName типа String, parameterDescription типа String, flag типа Boolean. ParameterName хранит имя параметра. Именно это имя будет добавляться в инструкцию java -jar при запуске приложения. ParameterDescription содержит описание параметра, представляемое пользователю при запуске, чтобы он имел представление, как добавление данного параметра повлияет на ход выполнения приложения. Атрибут flag означает, является ли параметр флагом, то есть принимает данный параметр какое-либо значение или нет.

Главным атрибутом в классе History является events типа List<Event>, хранящий список событий, произошедших во время работы приложения. Для каждого пользователя, который осуществил запуск приложения, формируется собственная история.

Основными атрибутами класса Event являются dat типа Date и message типа String. Атрибут dat хранит дату и время, когда произошло событие, а атрибут message — сообщение, представляемое пользователю и относящееся к данному событию.

Объявление данных классов, а также реализация отношений между ними, представлены в приложении Б 1.

Исходя из требований технического задания, все описанные выше сущности имеют свое отображение в базе данных (хост СУБД: mysql14.hostland.ru.). Логика взаимодействия с ней реализована в классе DbApi.

Для программного взаимодействия с базой данных используется ORM-средство Hibernate. Hibernate существенно упрощает, а следовательно и ускоряет, процесс разработки. Он позволяет оперировать данными, хранящимися в СУБД, на уровне объектов. Для этого объектам модели (далее сущностям), представленным на рис.2. проставлены соответствующие аннотации. При сохранении, извлечении, редактировании и удалении сущностей, Hibernate просматривает аннотации данной сущности и на их основе генерирует соответствующий СУБД Mysql SQL код. Таким образом, Hibernate освобождает разработчика от необходимости написания SQL кода, а также позволяет абстрагироваться от конкретной платформы СУБД. И при переходе на другую платформу разработчику не нужно переписывать SQL код, а достаточно лишь поменять параметры подключения к СУБД, которые хранятся в специальном конфигурационном файле с расширением hibernate.cfg.xml, который расположен в папке src/main/resources/hibernate.

Основными методами, реализованными в классе DbApi являются addResource — используется при регистрации ресурса, getAllResources — необходим для вывода списка всех зарегистрированных ресурсов, addUserToResource — используется для занесения логина и пароля, необходимых для авторизации пользователя на ресурсе, в базу данных, addApplicationToResource — вызывается при регистрации нового приложения на удаленном ресурсе, с которым установлено соединение, getApplicationsByResource — используется при формировании списка зарегистрированных приложений, addParameterToApplication — вызывается при добавлении к приложению входного параметра, clearApplicationHistory — используется для очистки истории запуска приложения.

Ниже представлена реализация данных методов.

public void addResource(String url){

Session session = sessionFactory.openSession();

Transaction transaction = session.beginTransaction();

Resource resource = new Resource();

resource.setURL(url);

session.persist(resource);

transaction.commit();

session.close();

}

public List<Resource> getAllResources(){

Session session = sessionFactory.openSession();

List<Resource> list = session.createCriteria(Resource.class).list();

session.close();

return list;

}

public void addUserToResource(String resourceUrl, String login, String password){

Session session = sessionFactory.openSession();

Transaction transaction = session.beginTransaction();

Resource resource = session.get(Resource.class, resourceUrl);

ResourceUser user = new ResourceUser();

user.setLogin(login);

user.setPassword(password);

user.setResource(resource);

resource.getUsers().add(user);

transaction.commit();

session.close();

}

public Application addApplicationToResource(String resourceUrl, String applicationName, String desc, String applicationPath){

Session session = sessionFactory.openSession();

Transaction transaction = session.beginTransaction();

Resource resource = session.get(Resource.class, resourceUrl);

History history = new History();

Application application = new Application();

application.setName(applicationName);

application.setDescription(desc);

application.setPath(applicationPath);

history.setApplication(application);

application.setHistory(history);

application.setResource(resource);

resource.getApplications().add(application);

transaction.commit();

application = resource.getApplications().get(resource.getApplications().size()-1);

session.close();

return application;

}

public List<Application> getApplicationsByResource(String resourceUrl){

Session session = sessionFactory.openSession();

List<Application> applications = new ArrayList<>();

applications.addAll(session.get(Resource.class, resourceUrl).getApplications());

session.close();

return applications;

}

public Parameter addParameterToApplication(long applicationId, String parameterName, String desc, boolean isFlag){

Session session = sessionFactory.openSession();

Transaction transaction = session.beginTransaction();

Application application = session.get(Application.class, applicationId);

Parameter parameter = new Parameter();

parameter.setParameterName(parameterName);

parameter.setFlag(isFlag);

parameter.setParameterDescription(desc);

parameter.setApplication(application);

application.getParameters().add(parameter);

transaction.commit();

parameter = application.getParameters().get(application.getParameters().size()-1);

session.close();

return parameter;

}

public void clearApplicationHistory(long applicationId){

Session session = sessionFactory.openSession();

Transaction transaction = session.beginTransaction();

session.get(Application.class, applicationId).getHistory().getEvents().clear();

transaction.commit();

session.close();

}

Решаемая задача подразумевает управление удаленным ресурсом. Логика взаимодействия с ним реализована. При ее реализации используется API, предоставляемый библиотекой JSch, разработанной компанией JCraft.

При запуске выбранного пользователем приложения вызывается метод startApplication. В данный метод пере

public List<String> startApplication(String resourceUrl, int port, String userLogin, String password, String command){

List<String> lines = new ArrayList<>();

try {

Session session = initSession(resourceUrl, port, userLogin, password);

Channel channel = initChannel(command, session);

channel.connect();

String dataFromChannel = getDataFromChannel(channel);

lines.addAll(Arrays.asList(dataFromChannel.split("\n")));

channel.disconnect();

session.disconnect();

} catch (Exception e) {

log.error(e);

}

return lines;

}

Данный метод возвращает список строк, хранящих результат выполнения запущенного на удаленном ресурсе java-приложения. Данный метод осуществляет подключение к выбранному пользователем ресурсу (initSession) и создает канал передачи данных(initChannel), по которому осуществляется передача команды, необходимой для запуска приложения, а также передача в систему результата ее работы(getDataFromChannel).

В пакете ru.igorek.core.utils хранятся два вспомогательных класса ConfigurationUtil и HibernateUtil.

Класс ConfigurationUtil осуществляет подключение файла configurationProperties.properties, лежащего в папке src/main/resources/etc, и предоставляет метод, позволяющий получить значение того или иного параметра по его имени. Все параметры, вынесенные в данный файл, хранятся в виде «ключ=значение». Данный подход считается хорошим стилем программирования, поскольку он позволяет сделать разрабатываемую систему более гибкой. Это обусловлено тем, что процесс изменения параметров, так или иначе влияющих на ход работы приложения, не требует повторной перекомпиляции и сборки проекта, на что могут тратиться большие временные ресурсы.

Класс HibernateUtil подключает файл hibernate.cfg.xml (расположен в папке src/main/resources/hibernate), в котором заданы параметры работы ORM-средства Hibernate, и на его основе создает экземпляр класса SessionFactory. Содержимое файла hibernate.cfg.xml представлено в приложении Б 1.

* 1. Модуль javafx (Реализация пользовательского интерфейса)

В данном модуле реализована логика работы графического пользовательского интерфейса.

В папке src/main/resources/fxml хранятся все файлы \*.fxml. Каждый файл такого типа описывает на языке XML структуру конкретного окна приложения. Данное описание включает в себя указание всех контейнеров и управляющих элементов, а также их параметры. Кроме того, в каждом файле \*.fxml указано, какой класс-контроллер ему соответствует и в каком \*.css файле описаны стилевые параметры отображения компонентов соответствующего окна интерфейса. Все файлы \*.css размещены в папке src/main/resources/styles.

Графический пользовательский интерфейс реализован в модуле javafx. Данный модуль содержит в себе классы-контроллеры, в которых реализована логика работы графического интерфейса и файлы с расширением \*.fxml, на основе которых строится визуальная составляющая интерфейса. При разработке графического интерфейса была использована вспомогательная утилита SceneBuilder, которая позволяет строить визуальную составляющую посредством простого перетаскивания необходимых компонентов в область разработки макета. Достоинством данной утилиты является то, что она позволяет формировать \*.fxml файл, в котором указывается соответствующий данному окну класс-контроллер. В классах-контроллерах хранятся обработчики различных событий для отдельных компонентов интерфейса, которые наступают во время работы с приложением.

ЗАКЛЮЧЕНИЕ

В ходе выполнения данной работы было разработано десктопное приложение, позволяющее пользователям осуществлять удаленный запуск java-программ с интерфейсом командной строки, а также получать результат выполнения данных программ для дальнейшего его использования в своих целях.

При разработке данного продукта были пройдены следующие этапы:

1. Проведен анализ требований к конечному продукту.
2. Разработано и согласованно с научным руководителем техническое задание.
3. Спроектирована система, а также отдельные ее компоненты.
4. Проведена реализация спроектированной системы.
5. Проведено тестирование конечного продукта.

Разработанное приложение может быть использовано в рамках задачи по моделированию галактик, для решения которой требуется удаленный запуск сложных вычислительных задач.

В дальнейшем в данной системе будет реализована возможность отслеживать в реальном времени ход выполнения запущенного java-приложения.

Список использованных источников

1. Типы архивов: <https://www.slideshare.net/ilialapitan/java-16611045>
2. SSH: <http://www.chin28.narod.ru/d7.3.htm>
3. Javafx https://ru.wikipedia.org/wiki/JavaFX

ПРИЛОЖЕНИЕ А — ТЕХНИЧЕСКОЕ ЗАДАНИЕ

А 1 Введение

Наименование продукта — десктопное приложение, предназначенное для работы на операционных системах семейства Windows и Linux, для удаленного запуска java-приложений с интерфейсом командной строки.

Область применения для приложения — разработка программного обеспечения, требующая запуск, например, расчетных, java-программ, находящихся на удаленных ресурсах.

А 2 Назначение разработки

Обеспечение удаленного запуска java-приложений, а также облегчение ввода и проверки параметров, необходимых для запуска этих приложений.

А 3 Основание для разработки

Основанием для разработки является задание на выпускную квалификационную работу.

А 4 Требования к функциональности продукта

A 4.1 Просмотр списка зарегистрированных в системе удаленных ресурсов

А 4.2 Регистрация нового ресурса

А 4.3 Авторизация пользователя

А 4.4 Просмотр списка зарегистрированных приложений

А 4.5 Регистрация нового приложения

А 4.6 Удаление приложения из списка представленных

А 4.7 Редактирование информации о зарегистрированном приложении

А 4.8.1 Добавление параметра

А 4.8.2 Редактирование параметра

А 4.8.3 Удаление параметра

А 4.9 Удаленный запуск выбранного приложения

А 4.9.1 Предоставление списка входных параметров запускаемого приложения

А 5 Требования к языкам программирования и программным средствам

Для реализации приложения должен использоваться язык программирования Java. Исходный код должен разрабатываться в соответствии со стандартами W3C.

А 6 Порядок контроля и приемки

Приемка и испытания производятся на основании методики тестирования, которая позволит проверить выполнение всех требований к продукту.

ПРИЛОЖЕНИЕ Б — ФРАГМЕНТЫ ПРОГРАММНОГО КОДА

Б 1 Объявление классов, реализующих спроектированную модель данных.

<Сюда вставлю код>

Б 2 Содержимое конфигурационного файла hibernate.cfg.xml

<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE hibernate-configuration PUBLIC "-//Hibernate/Hibernate Configuration DTD 3.0//EN" "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">

<hibernate-configuration>

<session-factory>

<!-- Настройки подключения к БД -->

<property name="connection.driver\_class">com.mysql.jdbc.Driver</property>

<property name="connection.url">jdbc:mysql://mysql14.hostland.ru:3306/host1593007</property>

<property name="connection.username">host1593007</property>

<property name="connection.password">0384a24c</property>

<property name="dialect">org.hibernate.dialect.MySQLDialect</property>

<property name="hibernate.show\_sql">true</property>

<property name="hibernate.connection.pool\_size">10</property>

<property name="hibernate.current\_session\_context\_class">thread</property>

<property name="hibernate.cache.provider\_class">org.hibernate.cache.internal.NoCacheProvider</property>

<property name="hibernate.hbm2ddl.auto">update</property>

</session-factory>

</hibernate-configuration>