УТВЕРЖДАЮ

должность _генеральный директор____
_____ООО «EdZSoft Design»_____

_____ /Никитин Сергей Васильевич

«_10_»_____сентября_____ 2022 г.

Функциональная схема и техническая документация

наименование вида ИС

«Сервис просмотра бланков ГИА-9»

Сокращенное наименование ИС

Техническая документация (для развёртывания АИС)

RU.48358662.00001-03 11 (согласно ГОСТ 19.103-77)

СОГЛАСОВАНО

должность _____

подпись_____/_____

«_10_»__сентября_____ 2022 г.

РАЗРАБОТЧИК

должность _генеральный директор__

подпись_____/Никитин С.В._

«_10_»__сентября_____ 2022 г.

Екатеринбург, 2022

Подп. и дата

Взам. инв. №

Инв. № дубл.

Подп. и дата

Инв. № подл.

СОДЕРЖАНИЕ

1 ВВЕДЕНИЕ

1.1 Область применения

Автоматизированная информационная система «Сервис просмотра изображений бланков ответов участников ГИА-9» (специализированная версия программного комплекса «Сервис публикации результатов оценочных процедур, ответов, бланков ответов, подачи апелляций, регистрации участников оценочных процедур») предназначена для публикации результатов экзаменов, бланков ответов, результатов распознавания бланков ответов.

Модуль подготовки бланков и сведений для экспорта в веб-сервис АИС «Просмотр бланков ГИА-9» взаимодействует непосредственно с РИС ГИА, получая оттуда сведения о бланках, участниках, результатах участников, их кратких ответов. Модуль выполнен в виде Desktop-приложения, и должен располагаться на АРМ с доступом к базе данных и к папке сервера пакетов. Рекомендуется размещение непосредственно на сервере пакетов.

1.2 Краткое описание возможностей

АИС «Сервис просмотра изображений бланков ответов участников ГИА-9» позволяет участнику ГИА-9:

– авторизоваться с использованием собственной Фамилии и номера документа (номер документа не хранится в открытом виде, только его преобразование в md5-хэшсумму с добавлением «соли»);

– увидеть/просмотреть перечень загруженных результатов;

– увидеть подробные сведения о первичной отметке участника (до перевода в 5-балльную шкалу), выставленных экспертами отметках по критериям, распознавании кратких ответов участника с указанием правильности/не правильности ответа, краткую структуру заданий;

– увидеть/просмотреть бланки ответов участника (без персональных данных), скачать их при необходимости.

Модуль выгрузки данных для АИС «Сервис просмотра изображений бланков ответов участников ГИА-9» позволяет выгрузить:

– бланки ответов участников с удалением персональных данных участников;

– сведения о кратких ответах участников ГИА;

– сведения о результатах участников ГИА.

Модуль работает под управлением Windows 10, не требует дополнительного программного обеспечения, поставляется в виде exe-файла, который следует поместить в любое место на жёстком диске.

1.3 Уровень подготовки пользователя

Для работы с веб-сервисом и модулем выгрузки от пользователя не требуется особой подготовки.

Администратору системы требуется знать устройство файловой системы и иметь на диске место для сохранения данных, знать логин и пароль доступа к базам данных ГИА на уровне запросов SQL.

Рекомендуется знание/умение работать с электронными таблицами Calc или Excel.

1.4 Перечень эксплуатационной документации

В перечень эксплуатационной документации входит:

– руководство для участника ГИА;

– руководство для администратора веб-сервиса;

– руководство для ответственного за подготовку бланков и выгрузку сведений.

Данная часть документации предназначена для технических работников, определяющих возможную схему функционирования сервиса.

2 НАЗНАЧЕНИЕ И УСЛОВИЯ ПРИМЕНЕНИЯ

2.1 Назначение системы

Две части указанной системы взаимодействуют с базой данных, формат которой соответствует единой распределённой базе данных ГИА-9, утверждённый ФГБУ «ФЦТ», опубликованный http://rustest.ru/gia/technological-solutions/federalnaya-baza-dannyh-gia-9/, сама обработка должна производиться в программном обеспечении, включающем Ixora TestReader для распознавания бланков (рекомендуются АИС ГИА «ОГЭ 1.0» либо АИС ГИА «ОГЭ 2.0» от ФГБУ «ФЦТ»).

2.2 Условия применения

Веб-сервис развёртывается на серверах с операционной системой Linux. В данном документе будут приведены примеры для Ubuntu 20.04 LTS.

Модуль должен иметь доступ к файловой системе на сервере пакетов Ixora TestReader для получения с неё бланков, а также к базе данных РИС ГИА. Модуль работает только под управлением Windows 10, требования к компьютеру аналогичны требованиям к АРМ для сервера пакетов Ixora TestReader.

3 ПОДГОТОВКА К РАБОТЕ

3.1 Состав и содержание необходимого программного обеспечения

Для развёртывания веб-сервиса требуется доступ по SSH, для функционирования — доступ по портам 80 и 443. На сервере базы данных (при создании кластера) понадобится открыть порт 5432. Возможна работа по FTP- или SMB-протоколу, если это будет целесообразным на файловом сервере.

Модуль следует расположить на сервере пакетов, либо на компьютере, имеющем доступ как к файловой структуре сервера пакетов, так и к базе данных РИС ГИА.

Файл extractgia9_v3.exe (v3 означает 3-ю ветку разработки модуля) следует поместить в любую папку на жёстком диске.

Файл работает только в Windows 10. При появлении сообщений о содержащемся в файле вирусе, следует понизить безопасность системы. Файл не содержит вирусов, дополнительно можно проверить его любым другим антивирусом.

Для выполнения запросов к базе данных РИС ГИА компьютер должен иметь доступ к базе данных, в чём следует убедиться, например, с использованием любого Transact SQL редактора, в частности, Microsoft SQL Server Management Studio.

Один комплект бланков занимает от 80 кб до 1 Мб, в связи с чем, следует убедиться перед запуском в наличии требуемого места.

3.2 Функциональная схема работы сервиса

Сервис построен по схеме взаимодействия Phusion Passenger с Nginx и Rails-приложениями. На. рисунке 1 приведена структурная схема взаимодействия.

Полученный http-запрос передаётся веб-серверу (реализован на nginx), который статичные файлы передаёт сразу в качестве ответа на запрос, а остальные файлы — через вспомогательный агент. Вспомогательный агент пересылает запросы к rails-исполнителю (ruby необходимой версии с надстройкой rails) и журналирует свои действия, а за его работой следит модуль watchdog, контролирующий зависания вспомогательного агента и агента журналирования. Подробное описание работы модуля можно найти на https://www.phusionpassenger.com/documentation/Design%20and%20Archi/tecture.html. Для работы сайта выбраны модули: Ruby 3-ей ветки, Rails 6-ой ветки, PostgreSQL 12-й ветки.

Для развёртывания сервиса в отказоустойчивом кластере, отдельно выносятся: файловый сервер со статичными файлами, сервер с базой данных. Балансировщик нагрузки настраивается таким образом, чтобы перераспределял нагрузку на серверы-исполнители, созданные в соответствии со схемой из рисунка №1.

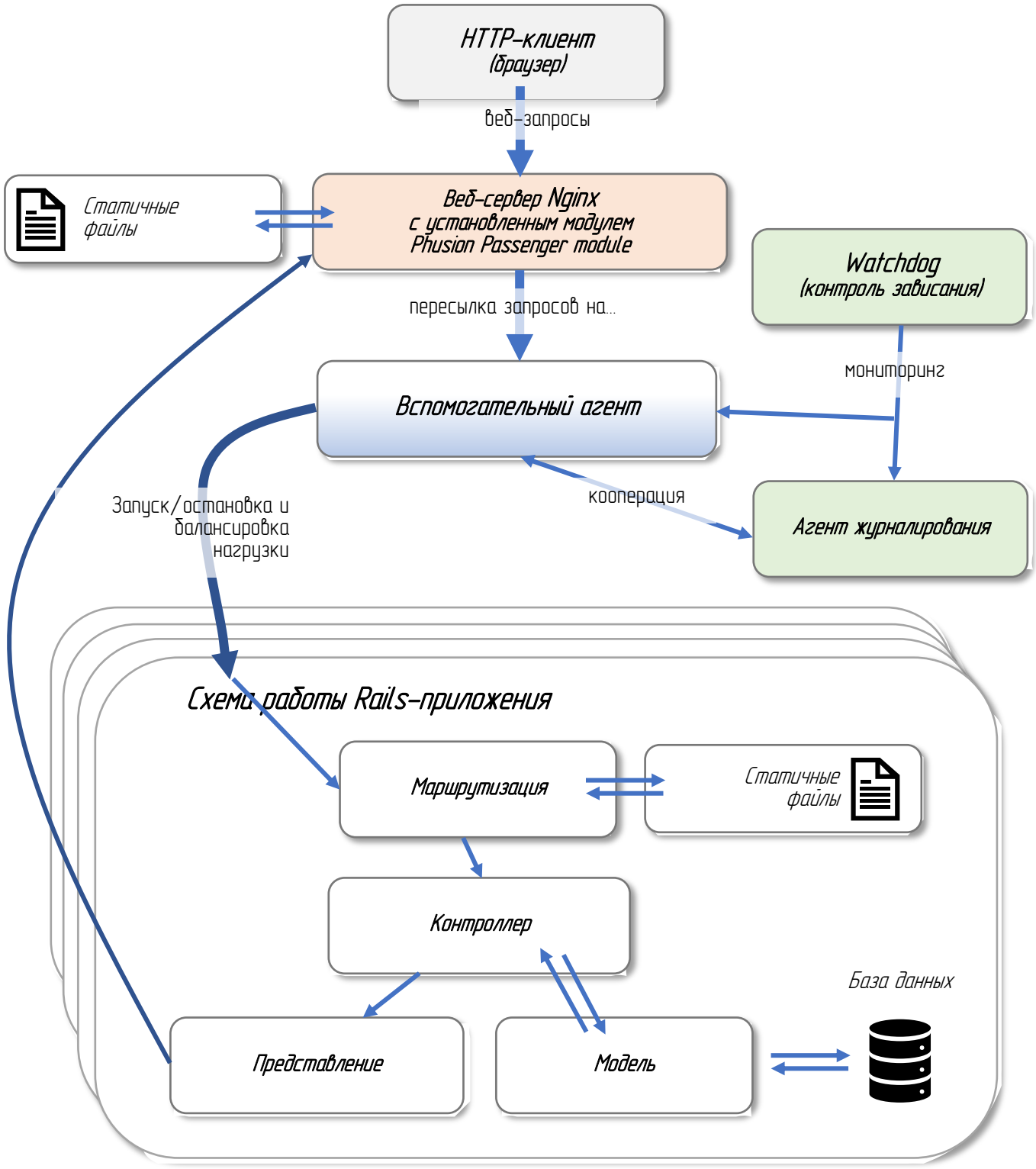


Рисунок 1. Функциональная схема работы веб-сервера

Rails-проекты строятся по схеме MVC: model, viewer, controller. Как показано на рисунке №1, полученный запрос поступает в систему маршрутизации, которая выбирает куда его далее направить: в контроллер или просто выдать статичный файл. Отметим, что при правильном построении системы статичные файлы открытого доступа могут выдаваться сразу nginx без запуска rails-процессов. Подробнее о схеме MVC можно прочитать https://guides.rubyonrails.org/getting_started.html.

3.3 Подготовка к развёртыванию системы

Для развёртывания системы на удалённом сервере понадобится отладочная машина с Linux, на котором необходимо установить git, ssh-клиент, сгенерировать ключи, получить доступ к bitbucket.org-проекту через ssh-ключ, получить доступ к github через ssh-ключ.

Удалённый сервер должен быть также добавлен в ключи github и bitbucket. Для этого у человека, производящего деплоинг должен быть аккаунт на bitbucket.org и github.com.

Полный исходный код программы для ЭВМ «Программный комплекс для реализации Сервиса просмотра бланков ГИА-9» (специальной версии программного обеспечения «Сервис публикации результатов оценочных процедур, ответов, бланков ответов, подачи апелляций, регистрации участников оценочных процедур»), включающий Автоматизированную информационную систему «Сервис просмотра изображений бланков ответов участников ГИА-9», расположен на сервисе bitbucket.org.

Для его клонирования используется команда:

```
git clone git@bitbucket.org:NikitinC/lks.git
```

Клонировать могут только участники bitbucket.org, добавленные в репозиторий (производится владельцем репозитория, для связи с которым можно использовать адрес электронной почты admin@edzsoft.ru). Участник должен добавить в свои ssh-ключи в сервисе ключи с сервера, на котором развёртывается проект и ключи с Linux, с которого управляется развёртывание (здесь и далее развёртывание и слово деплоинг будут использоваться равнозначно).

3.3.1 Добавление в bitbucket.org ключей серверов

Войдите в командную строку Linux. По умолчанию ключи будут добавлены в папки/файлы /Users/<имя_пользователя>/.ssh в macOS и /home/< имя_пользователя >/.ssh в Linux.

Шаг 1. Создать индивидуальную индентификацию.

1. Ввести в терминал и выполнить:

```
$ ssh-keygen
```

Появятся сообщения «Generating public/private rsa key pair» («Генерация публичной/приватной пары rsa-ключа») и «Enter file in which to save the key (/Users/<имя_пользователя>/.ssh/id_rsa):» («Введите, имя файла, в который будет произведено сохранение ключа»).

2. Нажать Enter или Return для подтверждения места по умолчанию.

Рекомендуется использовать настройки по умолчанию, если у вас нет веской причины для их изменения. Если вам необходимо использовать другой файл с другим путём, к примеру, «my-new-ssh-key», тогда вам следует ввести в терминал путь к этому файлу:

```
$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/Users/<имя_пользователя>/ssh/id_rsa):
/Users/<имя_пользователя>/.ssh/my-new-ssh-key
```

3. Ввести дважды парольную фразу.

Команда создает идентификатор по умолчанию с его открытым и закрытым ключами.

4. Вызвать содержимое ~/.ssh для просмотра ключевого файла.

```
$ ls ~/.ssh
```

В папке должно содержаться два файла: id_rsa и id_rsa.pub. Команда выше должна их показать. Если что-то не так, то сгенерируйте заново. В файле .pub содержится публичный ключ, а в файле без .pub — закрытый (личный) ключ.

Шаг 2. Добавление публичного ключа аккаунт Bitbucket

На сайте bitbucket.org выберите «**Personal settings**» («Личные настройки») из меню, появляющегося после щелчка по аватару. В появившемся слева меню выберите «**SSH keys**» («SSH-ключи»). В открывшемся окне с полным перечнем добавленных ключей, нажмите на «**Add key**» («Добавить ключ»).

В терминале следует выполнить команду:

```
$ cat ~/.ssh/id_rsa.pub
```

На macOS можно сразу скопировать содержимое в буфер обмена:

```
$ pbcopy < ~/.ssh/id_rsa.pub
```

Выберите и скопируйте в буфер обмена ключ. Вставьте его в раздел «Key*» («Ключ*») окна добавления ключа bitbucket. Выше напишите метку ключа (например, «Удалённый сервер» или «Домашний Linux»). Добавьте ключ в копилку, нажав снизу справа кнопку «Add key» («Добавить ключ»).

Эти действия необходимо выполнить для всех машин, связанных с деплоингом (как минимум, сервер и Linux для развёртывания).

3.3.2 Добавление в github.com ключей серверов

В логин на github.com добавляются те же ключи, что были добавлены в логин на bitbucket.org. Для добавления справа вверху следует нажать на аватар пользователя, и выбрать «Settings» («Настройки»).

В настройках слева в меню выбрать пункт «SSH and GPG keys» («SSH и GPG ключи»), в котором нажать на кнопку «New SSH key» («Новый SSH ключ»).

В терминале следует выполнить команду:

```
$ cat ~/.ssh/id_rsa.pub
```

На macOS можно сразу скопировать содержимое в буфер обмена:

```
$ pbcopy < ~/.ssh/id_rsa.pub
```

Выберите и скопируйте в буфер обмена ключ. Вставьте его в раздел «Key» («Ключ») окна добавления ключа github. Выше напишите метку ключа (например, «Удалённый сервер» или «Домашний Lunux»). Добавьте ключ в копилку, нажав снизу справа кнопку «Add SSH key» («Добавить SSH ключ»).

Эти действия необходимо выполнить для всех машин, связанных с деплоингом (как минимум, сервер и Linux для развёртывания).

3.4 Развёртывание системы на удалённом сервере

Для развёртывания системы должен быть подготовлен:

– машина с «локальным Linux» (например, VirtualBox с развёрнутым Ubuntu 20.04);

– сервер с «серверным Lunux» (например, сервер внутри сети VipNet с ip-адресом 1.2.3.4).

В дальнейшей части инструкции вместо реального ip-адреса будет использоваться 1.2.3.4, а машина для развёртывания будет называться «локальной».

Здесь и далее будем считать, что пользователь умеет пользоваться операционной системой Ubuntu 20.04 и устанавливать необходимые утилиты.

Шаг 1. Создаём пользователя для деплоинга

Для запуска программного обеспечения на сервере не следует использовать root-пользователя (суперадминистратор), поэтому первым шагом мы создаём пользователя, под которым будет запускаться rails-приложение, и ограничиваем его права. Это не даст хакерам даже в случае обнаружения им какой-то уязвимости получить доступ ко всеми серверу.

Подсоединяемся к удалённому серверу:

Локальная машина

```
$ ssh root@1.2.3.4
```

На удалённом сервере:

root@1.2.3.4

```
root$ adduser deploy
root$ adduser deploy sudo
root$ exit
```

Сразу же добавим ssh-ключи сервера в локальную машину, чтобы в последующем осуществлять «быстрый вход». Для этого используем утилиту **ssh-copy-id**. На Mac данная утилита устанавливается через homebrew, для чего используется команда «brew install ssh-copy-id».

Запускаем команды ниже и авторизуемся:

Локальная машина

```
$ ssh-copy-id root@1.2.3.4
$ ssh-copy-id deploy@1.2.3.4
```

Теперь доступ к серверу должен осуществляться без ввода пароля для пользователей root и deploy. Это важно, поскольку далее в процессе деплоинга через cap не придётся по многу раз вводить эти логины/пароли.

Итак, переходим к непосредственной настройке Linux.

Шаг 2. Настройка «локальной машины» для дальнейшего развёртывания АИС

Клонируем программные компоненты веб-сервиса на локальную машину:

Локальная машина

```
$ git clone git@bitbucket.org:NikitinC/lks.git
$ cd lks
$ curl -sL https://deb.nodesource.com/setup_12.x | sudo -E bash -
$ curl -sS https://dl.yarnpkg.com/debian/pubkey.gpg | sudo apt-key add -
$ echo "deb https://dl.yarnpkg.com/debian/ stable main" | sudo tee
/etc/apt/sources.list.d/yarn.list
$ sudo add-apt-repository ppa:chris-lea/redis-server
$ sudo apt-get update
$ sudo apt-get install git-core curl zlib1g-dev build-essential libssl-
dev libreadline-dev libyaml-dev libsqlite3-dev sqlite3 libxml2-dev
libxslt1-dev libcurl4-openssl-dev software-properties-common libffi-dev
dirmngr gnupg apt-transport-https ca-certificates redis-server redis-
tools nodejs yarn
$ git clone https://github.com/rbenv/rbenv.git ~/.rbenv
$ echo 'export PATH="$HOME/.rbenv/bin:$PATH"' >> ~/.bashrc
```

```
$ echo 'eval "$(rbenv init -)"' >> ~/.bashrc
$ git clone https://github.com/rbenv/ruby-build.git
~/.rbenv/plugins/ruby-build
$ echo 'export PATH="$HOME/.rbenv/plugins/ruby-build/bin:$PATH"' >>
~/.bashrc
$ git clone https://github.com/rbenv/rbenv-vars.git
~/.rbenv/plugins/rbenv-vars
$ exec $SHELL
$ rbenv install 3.0.0
$ rbenv global 3.0.0
$ ruby -v
```

В результате выполнения последних действий должен появиться ответ: «# ruby 3.0.0». Это означает, что Ruby версии 3.0 установлен.

Следующим шагом устанавливаем пакет bundler.

Локальная машина

```
$ gem install bundler
$ bundle -v
```

В результате выполнения последнего действия должно появиться сообщение о версии bundler (например, «# Bundler version 2.0»).

Для дальнейшей установки всех необходимых библиотек, достаточно ввести «bundler install» в каталоге lks и следовать подсказкам:

Локальная машина

```
$ bundler install
```

В результате выполнения могут появиться дополнительные требования об установке взаимозависимых компонентов — от компиляторов c++ и интерпретаторов python до обновления ядра Linux. Необходимо выполнять данные действия пока не появятся сообщения об успешной установке.

Возможно, для выполнения каких-то действий понадобятся права root (пароль от sudo для текущего пользователя). Подобную установку мы будем выполнять на следующем шаге на удалённом сервере.

Шаг 2. Установка Ruby на удалённом сервере

На данном шаге мы установим Ruby с использованием менеджера версий ruby. Менеджер версий чаще всё-таки используется в разработке, так как там требуется переключение между

версиями, но мы используем его и на деплоинговом сервере, так как это позволяет быстро обновлять версию Ruby в «продакшн» (здесь и далее, от английского «production» — реально работающий в текущий момент сервер (набор серверов), обеспечивающий работу сервиса).

Итак, вновь начнём с того, что устанавливаем необходимые для компиляции Ruby зависимости, в первую очередь это Webpacker, для которого надо добавить Node.js и Yarn. Заходим на удалённые сервер по ssh под логином deploy.

Добавляем репозиторий Node.js:

deploy@1.2.3.4

```
deploy$ curl -sL https://deb.nodesource.com/setup_12.x | sudo -E bash -
```

Добавляем репозиторий Yarn:

deploy@1.2.3.4

```
deploy$ curl -sS https://dl.yarnpkg.com/debian/pubkey.gpg | sudo apt-key
add -
deploy$ echo "deb https://dl.yarnpkg.com/debian/ stable main" | sudo tee
/etc/apt/sources.list.d/yarn.list
```

Также установим Redis для использования его в продакшене веб-сокетов ActionCable. Также можно настроить Redis в качестве хранилища кэш.

deploy@1.2.3.4

```
deploy$ sudo add-apt-repository ppa:chris-lea/redis-server
```

Обновим список репозиториев и начнём установку зависимостей для компиляции Ruby вместе с Node.js и Yarn:

deploy@1.2.3.4

```
deploy$ sudo apt-get update
deploy$ sudo apt-get install git-core curl zlib1g-dev build-essential
libssl-dev libreadline-dev libyaml-dev libsqlite3-dev sqlite3 libxml2-
dev libxslt1-dev libcurl4-openssl-dev software-properties-common libffi-
dev dirmngr gnupg apt-transport-https ca-certificates redis-server
redis-tools nodejs yarn
```

По окончании установки всех зависимостей, можно приступить к установке непосредственно Ruby версии 3.0.

deploy@1.2.3.4

```
deploy$ git clone https://github.com/rbenv/rbenv.git ~/.rbenv
deploy$ echo 'export PATH="$HOME/.rbenv/bin:$PATH"' >> ~/.bashrc
deploy$ echo 'eval "$(rbenv init -)"' >> ~/.bashrc
deploy$ git clone https://github.com/rbenv/ruby-build.git
~/.rbenv/plugins/ruby-build
deploy$ echo 'export PATH="$HOME/.rbenv/plugins/ruby-build/bin:$PATH"'
>> ~/.bashrc
deploy$ git clone https://github.com/rbenv/rbenv-vars.git
~/.rbenv/plugins/rbenv-vars
deploy$ exec $SHELL
deploy$ rbenv install 3.0.0
deploy$ rbenv global 3.0.0
deploy$ ruby -v
deploy$ # ruby 3.0.0
```

Если всё прошло успешно, то после команды «ruby -v» должна выйти версия «ruby 3.0.0». Теперь следует установить bundler (устанавливаем 2 версии — для новых и старых gem'ов).

deploy@1.2.3.4

```
deploy$ gem install bundler
deploy$ gem install bundler -v 1.17.3
deploy$ bundle -v
deploy$ # Bundler version 2.0
```

Последняя команда «bundle -v» должна вывести версию bundler. Если же выйдет ошибка о том, что bundler не найден, следует выполнить команду «rbenv rehash» и попытаться установить заново.

Шаг 3. Установка NGINX и Passenger на удалённом сервере

В продакшене используется nginx в качестве веб-сервера для получения HTTP-запросов (см. рис. 1). Затем эти запросы передаются в Passenger, который запускает необходимые приложения Ruby (и завершает их по окончании работы).

Установка производится аналогично предыдущим случаям — сначала добавляем репозиторий, потом устанавливаем пакеты и настраиваем их.

deploy@1.2.3.4

```
deploy$ sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --
recv-keys 561F9B9CAC40B2F7
deploy$ sudo sh -c 'echo deb https://oss-
binaries.phusionpassenger.com/apt/passenger focal main >
/etc/apt/sources.list.d/passenger.list'
deploy$ sudo apt-get update
deploy$ sudo apt-get install -y nginx-extras libnginx-mod-http-passenger
deploy$ if [ ! -f /etc/nginx/modules-enabled/50-mod-http-passenger.conf
]; then sudo ln -s /usr/share/nginx/modules-available/mod-http-
passenger.load /etc/nginx/modules-enabled/50-mod-http-passenger.conf ;
fi
deploy$ sudo ls /etc/nginx/conf.d/mod-http-passenger.conf
```

На текущем этапе nginx и Passenger установлены. Осталось указать верную версию Ruby в конфигурации Passenger. Для этого открываем файл конфигурации Passenger в редакторе:

deploy@1.2.3.4

```
deploy$ sudo vim /etc/nginx/conf.d/mod-http-passenger.conf
```

В этом файле нужно поменять только строку с «passenger_ruby» на:

```
passenger_ruby /home/deploy/.rbenv/shims/ruby;
```

Следует сохранить файл и перезапустить nginx.

deploy@1.2.3.4

```
deploy$ sudo service nginx start
```

На текущем этапе следует убедиться, что nginx работает, посетив обычный общедоступный ip-адрес (или по доменному имени), или адрес внутри сети VipNet с помощью браузера. В браузере должно появиться «стандартное приветствие NGINX» с надписью «Добро пожаловать в NGINX» (или «Welcome to NGINX» в зависимости от конфигурации операционной системы). На данном этапе уже можно проверить правильность «прокидывания портов» наружу центра обработки данных.

Далее мы удалим этот сервис nginx, подставив вместо него устанавливаемое приложение Ruby.

deploy@1.2.3.4

```
deploy$ sudo rm /etc/nginx/sites-enabled/default
deploy$ sudo vim /etc/nginx/sites-enabled/lks
```

Приступаем к настройке nginx под приложение lks:

deploy@1.2.3.4

```
server {
   listen 80;
   listen [::]:80;

   server_name _;
   root /home/deploy/lks/current/public;

   passenger_enabled on;
   passenger_app_env production;

   location /cable {
      passenger_app_group_name lks_websocket;
      passenger_force_max_concurrent_requests_per_process 0;

   } # Позволяем приложения вплоть до 100MB

   client_max_body_size 100m;

   location ~ ^/(assets|packs) {
      expires max;
      gzip_static on;
   }
}
```

Сохраняем и перезапускаем nginx.

deploy@1.2.3.4

```
deploy$ sudo service nginx reload
```

С этого момента сервер работать перестанет до окончания всех настроек.

Шаг 4. Установка PostgreSQL и создание базы данных сервиса

Система использует базу данных PostgreSQL. Для этого понадобится установить сервер postgres и libpq., что позволит нам скомпилировать pg rubygem.

Затем мы добавим deploy в пользователи баз данных postgres с полным доступом к базе данных, и используем учётную запись нового пользователя для создания баз данных приложения. Пользователь будет называться deploy.

В конце мы создадим базу данных с наименованием lks, которая будет использоваться для нашего приложения, сделаем её владельцем пользователя deploy.

deploy@1.2.3.4

```
deploy$ sudo apt-get install postgresql postgresql-contrib libpq-dev
deploy$ sudo su - postgres
postgres$ createuser --pwprompt deploy
postgres$ createdb -O deploy lks
postgres$ exit
```

После создания базы данных, к ней всегда можно присоединиться при помощи команды «psql -U deploy -W -h 127.0.0.1 -d lks». Проверяйте, что вы используете 127.0.0.1 при присоединении к базе, а не localhost.

В случае, если базу данных предполагается «поднимать» на другом сервере, следует сделать всё перечисленное на другом сервере, а также перенастроить PostgreSQL. Для этого:

Сервер для PostgreSQL

```
root# su - postgres -c "psql -c 'SHOW config_file;'"
```

В результате выполнения получим путь к файлу настроек, например, «/db/pgsql/postgresql.conf».

Запускаем редактор файла настроек:

Сервер для PostgreSQL

```
root# vim /db/pgsql/postgresql.conf
```

Находим и редактируем (по умолчанию строка закомментирована):

```
listen_addresses = '*'
```

Сохраняем, и открываем следующий файл:

Сервер для PostgreSQL

```
root# vim /db/pgsql/pg_hba.conf
```

Внизу файла добавляем строку:

```
host      all      all      192.168.0.10/32      password
```

В примере разрешён доступ с адресов 192.168.0.10/32, всем учётным записям (all). При желании можно ограничить запись только учётной записью deploy.

Перезапускаем сервер и пробуем к нему подсоединиться, указав вместо 127.0.0.1 адрес PostgreSQL-сервера.

Шаг 5. Установка приложения при помощи Capistrano

Приложение в bitbucket.org уже подготовлено для установки с локальной машины на сервер при помощи Capistrano. Требуется лишь изменить несколько файлов.

На этом этапе возвращаемся на локальную машину.

В папке lks/config/deploy открываем файл **production.rb**.

Находим в нём адрес сервера, на котором необходимо «развернуть» систему:

```
server '1.2.3.4', user: 'deploy', roles: %w{app db web}
```

Подсоединяемся к удалённому серверу и настраиваем переменные окружения:

Локальная машина

```
$ ssh deploy@1.2.3.4
deploy$ mkdir /home/deploy/lks
deploy$ vim /home/deploy/lks/.rbenv-vars
```

И вводим параметры:

```
DATABASE_URL=postgresql://deploy:PASSWORD@127.0.0.1/lks

RAILS_MASTER_KEY=содержимое_файла_lks/config/master.key
SECRET_KEY_BASE=1234567890
STRIPE_PUBLIC_KEY=x
STRIPE_PRIVATE_KEY=y
```

Соответственно, необходимо скопировать и заполнить содержимое файла lks/config/master.key, а также пароль (PASSWORD) и сервер базы данных PostgreSQL.

Сохраняем этот файл, и теперь переменные среды будут автоматически подгружаться каждый раз, когда на сервере будут запускаться команды Ruby.

Можно начинать разворачивать продакш-среду. Для этого возвращаемся в папку lks/ и запускаем из-под неё Capistrano.

Локальная машина

```
$ cap production deploy
```

Если по какой-то причине установка прервётся, следует перейти на удалённый сервер, зайти в папку /home/<имя_пользователя>/lks/, найти в нём папку releases и в ней вручную запустить bundle install, указав среду production:

deploy@1.2.3.4

```
$ RAILS_ENV=production bundle exec
```

Во время установки может понадобиться ввести пароль sudo, согласиться на скачивание с github или bitbucket, а также дополнительно установить какие-то библиотеки или зависимости. Данная задача выполняется один раз, и зависит только от дистрибутива Linux на удалённом сервере и выбора компонентов по умолчанию. Затем следует вернуться на локальную машину и вновь выполнить команду «cap production deploy».

По окончании сервер nginx будет перезапущен, и при переходе по ip-адресам сервера в браузере мы должны увидеть приложение.

Если вы видите ошибки, то следует подключиться к серверу по ssh и изучить журналы:

deploy@1.2.3.4

```
deploy$ # Журнал Rails
deploy$ less /home/deploy/lks/current/log/production.log
deploy$ # Журналы NGINX и Passenger
deploy$ sudo less /var/log/nginx/error.log
```

Как только вы обнаружите свою ошибку (часто это отсутствующая переменная среды или конфигурация для рабочей среды), вы можете исправить ее и перезапустить или повторно развернуть приложение.

Шаг 6. Установка SSL-сертификата Let's Encrypt

Последний шаг выполняется только в случае, если сервер не находится за балансировщиком нагрузки. Другими словами, сервер должен иметь белый ip-адрес.

Итак, переходим на сервер «ssh deploy@1.2.3.4», где устанавливаем и запускаем клиент Let's Encrypt.

deploy@1.2.3.4

```
deploy$ cd ~ git
deploy$ clone https://github.com/letsencrypt/letsencrypt
deploy$ cd letsencrypt/
deploy$ ./letsencrypt-auto
```

После компилирования и настройки клиента, запускаем генерацию SSL-сертификата. Для этого понадобятся домен («example.com»), электронная почта («email@example.com»), путь к программе «/home/deploy/lks/current/public».

deploy@1.2.3.4

```
deploy$ sudo /home/deploy/.local/share/letsencrypt/bin/letsencrypt certonly
--webroot --webroot-path /home/deploy/lks/current/public --renew-by-default
--email email@example.com --text --agree-tos -d example.com -d
www.example.com
```

Еще одна вещь, которую нам нужно сделать, чтобы получить рейтинг A+ по безопасности SSL, — это создать собственную группу Диффи-Хеллмана. Причина этого в том, что он защищает нас от атаки Logjam. Это довольно просто. Нам просто нужно использовать OpenSSL для создания группы Diffie-Hellman, которую мы позже добавим в нашу конфигурацию Nginx SSL.

deploy@1.2.3.4

```
deploy$ cd ~
deploy$ openssl dhparam -out dhparams.pem 2048
```

Реконфигурируем nginx:

```
        listen 443 ssl;

        sl_certificate /etc/letsencrypt/live/example.com/fullchain.pem; s
        sl_certificate_key /etc/letsencrypt/live/example.com/privkey.pem;


        ssl_session_timeout 5m;
        ssl_protocols TLSv1 TLSv1.1 TLSv1.2;
        ssl_ciphers 'ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES128-GCM-
SHA256:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-AES256-GCM-SHA384:DHE-RSA-
AES128-GCM-SHA256:DHE-DSS-AES128-GCM-SHA256:kEDH+AESGCM:ECDHE-RSA-AES128-
SHA256:ECDHE-ECDSA-AES128-SHA256:ECDHE-RSA-AES128-SHA:ECDHE-ECDSA-AES128-
SHA:ECDHE-RSA-AES256-SHA384:ECDHE-ECDSA-AES256-SHA384:ECDHE-RSA-AES256-
SHA:ECDHE-ECDSA-AES256-SHA:DHE-RSA-AES128-SHA256:DHE-RSA-AES128-SHA:DHE-
DSS-AES128-SHA256:DHE-RSA-AES256-SHA256:DHE-DSS-AES256-SHA:DHE-RSA-AES256-
SHA:AES128-GCM-SHA256:AES256-GCM-SHA384:AES128-SHA256:AES256-SHA256:AES128-
SHA:AES256-SHA:AES:CAMELLIA:DES-CBC3-
SHA:!aNULL:!eNULL:!EXPORT:!DES:!RC4:!MD5:!PSK:!aECDH:!EDH-DSS-DES-CBC3-
SHA:!EDH-RSA-DES-CBC3-SHA:!KRB5-DES-CBC3-SHA';

        ssl_prefer_server_ciphers on;
        ssl_session_cache shared:SSL:10m;
        ssl_dhparam /home/deploy/dhparams.pem;
```

После окончания конфигурирования следует перезапустить NGINX:

deploy@1.2.3.4

```
deploy$ sudo nginx -s reload
```

Теперь веб-приложение должно работать через https://адрес_сервера.

Для окончания поставим обновление SSL-сертификата в задачу cron. Поскольку Let's Encrypt предназначен для недолговечных SSL-сертификатов, нам нужно настроить сценарий обновления для периодического запуска, чтобы попытаться обновить SSL-сертификат. Мы добавим это в задание cron пользователя root, чтобы он мог пытаться обновлять сертификат каждый понедельник в 2:30.

deploy@1.2.3.4

```
deploy$ sudo crontab -e
```

И в открывшемся меню добавляем строку:

```
30 2 * * 1 /home/deploy/.local/share/letsencrypt/bin/letsencrypt renew
```

Сохраняем файл. Больше никаких действий не требуется.

4 ОПИСАНИЕ ОПЕРАЦИЙ ПО ОБСЛУЖИВАНИЮ

В разделе будут рассмотрены наиболее частые операции по обслуживанию портала.

4.1 Бэкапирование базы данных

Наиболее частая операция — бэкапирование базы данных, поскольку бэкапирование исходных текстов программы не требуется (его всегда можно взять с bitbucket.org).

Команда для создания бэкапа базы данных со сжатием в gzip-файл.

deploy@1.2.3.4

```
deploy$ pg_dump -U deploy -W -h 127.0.0.1 -d lks | gzip > lks_backup.gz
```

После запуска команды потребуется ввести пароль deploy от сервера баз данных 127.0.0.1.

Чтобы скачать созданный файл, его следует переместить в файлы в каталоге для скачивания. Например, lks/public/uploads/lks_backup.gz.

Для этого:

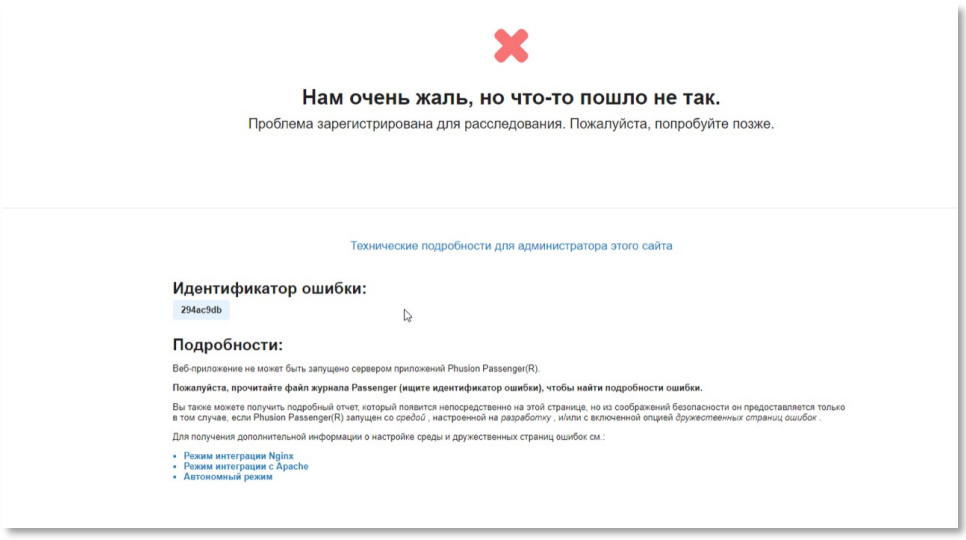
deploy@1.2.3.4

```
deploy$ mv *.gz lks/current/public/uploads/
```

А затем в браузере перейти по адресу: https://адрес_сервера/uploads/lks_backup.gz и использовать скачанный файл.

4.2 Перезапуск сервера nginx

После проблем со связью с сервером баз данных может понадобиться перезапуск программного обеспечения. Чаще всего проблема в браузере выглядит примерно так:



Для перезапуска соединитесь с сервером и изучите файлы:

deploy@1.2.3.4

```
deploy$ # Журналы NGINX и Passenger
deploy$ sudo less /var/log/nginx/error.log
```

В файлах error.log содержится описание ошибки nginx, номер которой (идентификатор) указан в сообщении для пользователя.

Перед изучением следует попробовать перезагрузить систему.

deploy@1.2.3.4

```
deploy$ sudo shutdown -r now
```

Если по окончании перезагрузки система не начала работать, изучайте журналы nginx.

4.3 Ошибка 500 при выполнении веб-приложения Rails

Данная ошибка однозначно свидетельствует об ошибке внутри среды Rails: либо пользователь пытается выполнить операции, которую не имеет права выполнять, либо действия пользователя привели к ошибке в логике операций.

Подробности можно узнать в файле журнала Rails.

deploy@1.2.3.4

```
deploy$ # Журнал Rails
deploy$ tail -n 100 /home/deploy/lks/current/log/production.log
```

Анализ журнала во время технической поддержки производится производителем программного обеспечения, в том числе, с привлечением программ автоматизированного контроля за ошибками на серверах.

При отсутствии технической поддержки со стороны производителя рекомендуется изучить файл production.log и исправить ошибки в базе данных или исходных программных файлах.

4.4 Перезапуск Passenger

После деплоинга или ручного изменения любого из файлов, следует перезапустить passenger следующей командой:

deploy@1.2.3.4

```
deploy$ passenger-config restart-app
```

5 АВАРИЙНЫЕ СИТУАЦИИ

На текущий момент аварийных ситуаций не зафиксировано.

При их появлении, просим дать доступ к отладке программы и связаться с разработчиком по email: admin@edzsoft.ru.

6 РЕКОМЕНДАЦИИ ПО ОСВОЕНИЮ

Для начала изучения структуры работы приложения и его оперативной перенастройки и доработки, следует изучить:

– модель MVC;

– структуру стандартного приложения Rails;

– структуру HAML (весь исходный код использует структурирование HAML).

Что такое MVC

Модель программирования Model–View–Controller (MVC, «Модель–Представление–Контроллер», «Модель–Вид–Контроллер») – схема разделения данных приложения и управляющей логики на три отдельных компонента: модель, представление и контроллер таким образом, чтобы модификацию каждого компонента можно было осуществлять независимо.

При этом

Модель (Model) предоставляет собой данные и реагирует на команды контроллера, изменяя своё состояние (принцип инкапсуляции объектно–ориентированного программирования);

Представление (View) отвечает за отображение данных модели пользователю, реагируя на изменения модели (включает HTML–код).

Контроллер (Controller) интерпретирует действия пользователя, оповещая модель о необходимости изменений (получает сведения от браузера и вызывает необходимые модели и представления).

Структура каталогов Rails

app	каталог с приложением
bin	–
config	каталог конфигурации
db	каталог для базы данных
lib	библиотеки
log	журналы
public	файлы открытого доступа (без обработки)
test	–
tmp	временная папка
vendor	папка для библиотек

В папке app находятся все компоненты приложения:

app/controllers — подкаталог controllers, где Rails ищет классы контроллеров. Контроллер обрабатывает веб–запрос от пользователя;

app/models — подкаталог models содержит модели – объекты, которые моделируют и упаковывают данные, хранящиеся в базе данных приложения;

app/views — подкаталог views содержит шаблоны представления, которые Rails наполняет данными и преобразует в HTML для возврата в браузер.

Каталог config содержит небольшой объем кода конфигурации, который необходим приложению, включая конфигурацию базы данных (в файле database.yml), структуру среды Rails (environment.rb) и маршрутизацию входящих веб-запросов (rout.rb), а также mailer.

Каталог public подобен общедоступному каталогу для веб-сервера, в этом каталоге есть неизменяемые веб-файлы, такие как файлы JavaScript (public / javascripts), графика (public / images), таблицы стилей (public / stylesheets) и файлы HTML (общественности).

В случае сервиса просмотра бланков, в этот каталог помещаются бланки участников ГИА.

Дополнительно можно ознакомиться с актуальной версией инструкции по адресу: https://edzsoft.ru/lks/

Там же можно получить доступ к тестированию платформы.

7 ТЕРМИНЫ И СОКРАЩЕНИЯ

Термин	Полная форма
АИС	Автоматизированная информационная система — совокупность программно-аппаратных средств, предназначенных для автоматизации деятельности, связанной с хранением, передачей и обработкой информации
АРМ	Автоматизированное рабочее место
Апелляция	Процедура по проверке не вступивших в законную силу актов
Бланки ОГЭ	Бланки регистрации устной части, бланки ответов №1 (с регистрационными данными участника и ответами на краткие вопросы), бланки ответов №2 (лист 1 и лист 2), в том числе дополнительные бланки ответов №2, выдаваемые участникам экзаменам при необходимости
Браузер	Прикладное программное обеспечение для просмотра Интернет — сайтов, WEB-страниц, компьютерных документов , а так же для решения других задач.
Верификация	Проверка, подтверждение, в данном случае распознавание с контролем истинности
ГИА	Государственная итоговая аттестация программам основного общего образования
Гиперссылка	Часть гипертекстового документа, ссылающаяся на элемент в этом документе или в пространстве Интернет
ДБО №2	Дополнительный бланк ответов №2
Деперсонализация	Удаление из документов персональных данных
ОГЭ	Основной государственный экзамен
ИК	Индивидуальный комплект участника экзамена
КИМ	Контрольные измерительные материалы ОГЭ
ППЭ	Пункт проведения экзаменов
ПО	Программное обеспечение
Образовательная организация	Организация, осуществляющая образовательную деятельность по имеющим государственную аккредитацию образовательным программам среднего общего образования
Порядок ГИА	Порядок проведения государственной итоговой аттестации по образовательным программам основного общего образования, утвержденный приказом Минпросвещения России и Рособрнадзора от 07.11.2018 г. №189/1513 (зарегистрирован Минюстом России 10.12.2018, регистрационный №52953)
Рособрнадзор	Федеральная служба по надзору в сфере образования и науки
Сеть «Интернет»	Информационно-телекоммуникационная сеть «Интернет»
Соглашения	В данном случае (в тексте) – политика в области обработки персональных данных
Участники ГИА, Участники экзаменов	Обучающиеся по образовательным программам основного общего образования, допущенные в установленном порядке к ГИА
ЭМ	Экзаменационные материалы ОГЭ
Java-Script	Язык программирования, который позволяет создать динамически обновляемый контент, управляет мультимедиа, анимирует изображения и др.
PDF	межплатформенный открытый формат электронных документов с использованием ряда возможностей языка PostScript.

СОСТАВИЛИ

Наименование организации, предприятия	Должность исполнителя	Фамилия, имя, отчество	Подпись	Дата
ООО «EdZSoft»	Генеральный директор	Никитин С.В.		10.09.2022

СОГЛАСОВАНО

Наименование организации, предприятия	Должность исполнителя	Фамилия, имя, отчество	Подпись	Дата

Подп. и дата | Взам. инв. № | Инв. № дубл. | Подп. и дата | Инв. № подл

Лист регистрации изменений

Изм	новых				Всего листов (страниц) в докум.	№ документа	Входящий номер сопроводи-тельного документа и дата	Подпись	Дата
	изменен-ных	заменен-ных	новых	изъятых					

Подп. и дата

Взам. инв. №

Инв. № дубл.

Подп. и дата

Инв. № подл

Ли	Изм.	№ докум.	Подп.	Дата	Лист
					29