**ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ**

**ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ**

**«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»**

Факультет компьютерных наук

Образовательная программа «Программная инженерия»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| СОГЛАСОВАНО  Старший преподаватель департамента больших данных и информационного поиска  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В.В. Куренков  «\_\_» \_\_\_\_\_\_\_\_\_\_\_ 2024 г. | |  | УТВЕРЖДАЮ  Академический руководитель образовательной программы «Программная инженерия»  старший преподаватель департамента программной инженерии  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Н.А. Павлочев  «\_\_» \_\_\_\_\_\_\_\_\_\_\_ 2024 г. | |
| |  |  | | --- | --- | | ***Подп. и дата*** |  | | ***Инв. № дубл.*** |  | | ***Взам. инв. №*** |  | | ***Подп. и дата*** |  | | ***Инв. № подл*** |  | | **Система проверки решений по программированию для Лицея НИУ ВШЭ с автоматизированным подсчётом рейтинга**  **Руководство оператора**  **ЛИСТ УТВЕРЖДЕНИЯ**  **RU.17701729.05.10-01 34 01-1-ЛУ** | | | | | | |
|  | |  | | | | |
| Исполнитель  студент группы БПИ219  \_\_\_\_\_\_\_\_\_ / С.О. Котовский /  «\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2024 г. | | | | |
|  | | | | | | |
|  | | | | |  | |

**Москва 2024**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| УТВЕРЖДЕН  RU.17701729.05.10-01 34 01-1-ЛУ |  | |  | |
| |  |  | | --- | --- | | ***Подп. и дата*** |  | | ***Инв. № дубл.*** |  | | ***Взам. инв. №*** |  | | ***Подп. и дата*** |  | | ***Инв. № подл*** |  | | **Система проверки решений по программированию для Лицея НИУ ВШЭ с автоматизированным подсчётом рейтинга**  **Руководство оператора**  **RU.17701729.05.10-01 34 01-1**  **Листов 20** | | | | |
|  | |  | | |
|  | | |
|  | | | | |
|  | | | |  |

**Москва 2024**

ОГЛАВЛЕНИЕ

[1. НАЗНАЧЕНИЕ И ОБЛАСТЬ ПРИМЕНЕНИЯ 4](#_Toc161930915)

[1.1. Функциональное назначение 4](#_Toc161930916)

[1.2. Эксплуатационное назначение 4](#_Toc161930917)

[1.3. Описание функций веб-приложения 4](#_Toc161930918)

[1.3.1. Описание функции “Авторизация” 4](#_Toc161930919)

[1.3.2. Описание функционала навигационного меню 4](#_Toc161930920)

[1.3.3. Описание функционала профиля пользователя 4](#_Toc161930921)

[1.3.4. Описание админ-функционала преподавателя 5](#_Toc161930922)

[1.3.5. Описание страницы создания аккаунта студента 5](#_Toc161930923)

[1.3.6. Описание функционала импорта контестов из Ejudge 5](#_Toc161930924)

[1.3.7. Описание страницы со списком контестов 5](#_Toc161930925)

[1.3.8. Описание страницы с детальной информацией о контесте 5](#_Toc161930926)

[1.3.9. Описание страницы со списком посылок (решений) 6](#_Toc161930927)

[1.3.10. Описание страницы с рейтингом 6](#_Toc161930928)

[2. УСЛОВИЯ ВЫПОЛЕНИЯ ПРОГРАММЫ 7](#_Toc161930929)

[2.1. Требования к составу и параметрам технических средств 7](#_Toc161930930)

[2.2. Минимальный состав программных средств 7](#_Toc161930931)

[2.3. Требования к оператору 8](#_Toc161930932)

[3. ЗАПУСК ПРИЛОЖЕНИЯ 10](#_Toc161930933)

[3.1. Установка программы 10](#_Toc161930934)

[3.2. Запуск и работа программы 10](#_Toc161930935)

[3.2.1. Подготовка к запуску 12](#_Toc161930936)

[3.2.2. Запуск сервиса 12](#_Toc161930937)

[3.2.3. Использование API 12](#_Toc161930938)

[3.2.4. Остановка сервиса 14](#_Toc161930939)

[3.2.5. Обработка ошибок 15](#_Toc161930940)

[4. СООБЩЕНИЯ ОПЕРАТОРУ 16](#_Toc161930941)

[ГЛОССАРИЙ 18](#_Toc161930942)

[ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ 20](#_Toc161930943)

# НАЗНАЧЕНИЕ И ОБЛАСТЬ ПРИМЕНЕНИЯ

## Функциональное назначение

Разработка направлена на внесение технических, логических, визуальных и архитектурных доработок работы, выполненной в рамках курсового проекта в 2023 году «Система проверки решений по программированию для Лицея НИУ ВШЭ с автоматизированным подсчётом рейтинга».

## Эксплуатационное назначение

Мероприятия, проводимые в рамках разработки, направлены на улучшение пользовательского опыта и упрощения дальнейшей поддержки дорабатываемого комплекса программно-технических решений.

Конечные пользователи – учащиеся и преподаватели информатики/программирования Лицея НИУ ВШЭ.

## Описание функций веб-приложения

* + 1. Описание функции “Авторизация”

Функция «Авторизация» предоставляет пользователям следующие возможности:

* Возможность ввести e-mail и пароль для доступа к своему аккаунту.
* Возможность отправки формы для авторизации в системе.
* В случае ввода неверных данных пользователь получает сообщение об ошибке.
* При возникновении проблем соединения с сервером или ошибок на стороне сервера, пользователю отображается соответствующее уведомление.
  + 1. Описание функционала навигационного меню

Навигационное меню предоставляет следующие функции:

* Адаптация меню в зависимости от роли пользователя (студент, преподаватель), предоставляя доступ к соответствующим разделам приложения.
* Для авторизованных пользователей отображаются пункты меню, такие как "Контесты", "Рейтинг", "Решения" и "Профиль". Для преподавателей доступна кнопка “Регистрация пользователей”, “Создать контест”.
* Возможность выхода из аккаунта, что удаляет авторизационные данные пользователя и перенаправляет на страницу входа.
  + 1. Описание функционала профиля пользователя

Профиль пользователя (личный кабинет) предлагает следующие возможности:

* Отображение персональной информации, включая ФИО, организацию, группу и электронную почту.
* Настройка профиля и возможность изменения пароля (если предусмотрено).
* Визуализация прогресса пользователя через интеграцию с системой рейтингов и статистикой решенных задач.
  + 1. Описание админ-функционала преподавателя

Админ-функционал преподавателя обладает следующим функционалом:

* Возможность редактирования информации о преподавателях и студентах (создание пользователей путём ручного ввода/загрузки csv-файла
* Импорт контестов из системы Ejudge по id, а также через форму ручного создания.
  + 1. Описание страницы создания аккаунта студента
* На странице создания аккаунта студента доступны следующие функции:
* Ввод данных студента, включая личную информацию и учебные данные.
* Отправка формы регистрации для создания нового аккаунта студента.
* Получение подтверждения о успешном создании аккаунта или сообщения об ошибке.
  + 1. Описание функционала импорта контестов из Ejudge

Страница импорта контестов предоставляет следующие возможности:

* Ввод данных для импорта контеста по его ID.
* Запуск процесса импорта с отображением статуса выполнения и результатов операции.
  + 1. Описание страницы со списком контестов
* Отображение информации о контестах, включая название, ID, дату окончания приёма решений и доступные языки программирования.
* Предоставление кнопки “Перейти к контесту”, позволяющей пользователю переходить к подробной странице контеста для участия или просмотра задач.
  + 1. Описание страницы с детальной информацией о контесте

На странице конкретного контеста доступны следующие функции:

* Визуализация общей информации о контесте, включая название, ID, даты начала и окончания.
* Отображение списка задач контеста с детальном описанием каждой задачи.
* Предоставление интерфейса для отправки решений, включая выбор языка программирования и загрузку файла с кодом или ввод кода непосредственно в текстовое поле.
* Вывод информации о статусе отправленного решения (принято, ошибка компиляции, неправильный ответ и т.д.).
  + 1. Описание страницы со списком посылок (решений)

Страница “Решения” позволяет пользователям:

* Просматривать список всех решений, отправленных пользователем, с сортировкой по времени отправки.
* Просматривать детальную информацию о каждом решении, включая ID посылки, время исполнения, количество затраченной памяти и вердикт тестирующей системы.
* При нажатии карточки с решением отображается исходный код решения с возможностью его копирования.
  + 1. Описание страницы с рейтингом

На странице рейтинга доступны следующие функции:

* Просмотр таблицы рейтинга пользователей.
* Возможность фильтрации рейтинга по классам, группам или другим параметрам, если это предусмотрено логикой веб-приложения.
* Экспорт таблицы рейтинга в формате CSV для дальнейшего анализа или использования вне приложения.

# УСЛОВИЯ ВЫПОЛЕНИЯ ПРОГРАММЫ

## Требования к составу и параметрам технических средств

**Для сервера:**

* **Процессор**: Двухъядерный или более мощный, с частотой не менее 2.5 GHz обеспечит эффективное выполнение бэкенд-операций и обработку запросов к веб-приложению.
* **Оперативная память**: Минимум 2 ГБ для поддержания стабильной работы приложения и обработки параллельных запросов.
* **Место на диске**: Не менее 5 ГБ для хранения кода приложения, баз данных, логов и временных файлов.
* **Операционная система**: Linux (желательно Ubuntu 18.04 LTS или более новая версия) для лучшей совместимости с разработкой и развёртыванием Node.js приложений и управления сервером через Apache или Nginx.
* **Сеть**: Высокоскоростное подключение к Интернет для быстрого обмена данными с клиентами и своевременного обновления приложения и его зависимостей.

**Для клиентского устройства:**

* **Устройство**: Поддержка современных веб-браузеров на ПК, ноутбуке или мобильном устройстве.
* **Операционная система**: Актуальная версия Windows, MacOS, Linux для ПК или iOS, Android для мобильных устройств.
* **Процессор и оперативная память**: Современный процессор и минимум 2 ГБ оперативной памяти для ПК или 1 ГБ для мобильных устройств для обеспечения плавной работы веб-приложения.
* **Интернет**: Стабильное подключение к Интернет для доступа к серверу приложения и обмена данными.
* **Браузер**: Последние версии Google Chrome, Mozilla Firefox, Safari, или Microsoft Edge с включенным JavaScript.

## Минимальный состав программных средств

**На стороне сервера:**

* **Операционная система**: Совместимость с Node.js и поддержка веб-серверов, таких как Nginx или Apache.
* **Node.js**: Среда выполнения для запуска JavaScript на сервере, необходима для работы серверной части веб-приложения.
* **Веб-сервер**: Nginx или Apache для обработки входящих HTTP-запросов и их маршрутизации к приложению.

**На стороне клиента:**

* **Операционная система**: Совместимость с современными веб-браузерами.
* **Веб-браузер**: Последние версии основных браузеров с включенным JavaScript для корректной работы веб-приложения.
* Для разработки и отладки:
* **IDE**: JetBrains WebStorm или аналогичные среды разработки, поддерживающие JavaScript и Vue.js.
* **Система контроля версий**: Git для управления версиями кода и совместной работы над проектом.

## Требования к оператору

* **Знание основ работы с веб-браузерами** и понимание принципов работы веб-приложений.
* **Базовые навыки работы с командной строкой** и инструментами разработки, если оператор участвует в разработке или поддержке проекта.
* **Основы использования системы контроля версий (Git)** для возможности отслеживания изменений в коде и совместной работы над проектом.

**Профессиональные качества**:

* **Внимательность к деталям**: Критически важно для выявления и исправления мелких ошибок в работе веб-приложения и обеспечения качества пользовательского интерфейса.
* **Адаптивность и готовность к обучению**: Веб-разработка постоянно эволюционирует, поэтому необходимо быть готовым к изучению новых технологий, фреймворков и лучших практик.
* **Проблемно-ориентированность**: Способность быстро находить и эффективно решать возникающие проблемы в работе приложения или при взаимодействии с пользователем.

**Дополнительные требования для операторов, участвующих в разработке:**

* **Знание и опыт работы с Vue.js и сопутствующими технологиями** (например, Vuex для управления состоянием приложения, Vue Router для маршрутизации).
* **Понимание RESTful API** и опыт взаимодействия с backend-системами, а также базовые знания по работе с базами данных.
* **Опыт работы с Node.js**: Понимание основ работы среды выполнения JavaScript на стороне сервера, знание основных модулей и работы с npm.
* **Навыки работы с инструментами разработчика в браузере** для отладки веб-приложений и анализа их производительности.

# ЗАПУСК ПРИЛОЖЕНИЯ

## Установка программы

**Docker**: Docker позволяет упаковывать приложение и его зависимости в контейнер, который может быть легко развернут на любой системе. Установите Docker, следуя инструкциям на официальном сайте: https://docs.docker.com/get-docker/

**Docker Compose**: Docker Compose используется для запуска мульти-контейнерных Docker приложений. Установите Docker Compose, следуя инструкциям на официальном сайте: https://docs.docker.com/compose/install/

**Клонирование репозитория**: Клонируйте репозиторий вашего проекта на сервер, используя

git clone <URL\_репозитория>

**Конфигурация проекта**: В корне проекта создайте файл .env для хранения переменных окружения, необходимых как для веб-приложения, так и для backend-прослойки.

Установка и запуск веб-приложения (Nuxt)

**Установка зависимостей**: В директории веб-приложения запустите команду, чтобы установить все зависимости:

npm install

**Сборка проекта**: Для сборки оптимизированной версии вашего Nuxt приложения для производства, выполните команду:

npm run build

Настройка и запуск backend-прослойки (Node.js)

**Docker Compose**: В директории с файлом docker-compose.yml запустите следующую команду для сборки и запуска контейнеров согласно вашему конфигу:

docker-compose up -d

Эта команда соберет образы Docker для backend-прослойки и других сервисов, указанных в вашем docker-compose.yml, и запустит их в фоновом режиме.

## Запуск и работа программы

**Развертывание веб-приложения (Nuxt) на сервере**

**Nginx или другой веб-сервер**: Настройте веб-сервер так, чтобы он отдавал файлы из директории .nuxt/dist сгенерированной при сборке Nuxt приложения. Пример конфигурации для Nginx:

server {

listen 80;

server\_name yourdomain.com;

location / {

root /path/to/your/nuxt/app/.nuxt/dist;

try\_files $uri $uri/ /index.html;

}

}

Примените изменения, перезапустив Nginx:

sudo systemctl restart nginx

Веб-приложение теперь должно быть доступно через настроенный домен.

**Взаимодействие с backend-прослойкой**

Backend-прослойка запущенная через Docker Compose, автоматически настраивается и связывается с веб-приложением согласно конфигурации в docker-compose.yml.

Убедитесь, что веб-приложение корректно конфигурировано для общения с backend-прослойкой, особенно в части указания правильных эн дпоинтов и токенов доступа, если таковые требуются. В файле конфигурации .env или другом, используемом для хранения настроек окружения, должны быть указаны URL-адреса и порты, по которым frontend часть может обращаться к backend-сервисам.

**Мониторинг и логирование**

**Docker Compose** также позволяет легко настроить логирование и мониторинг за работой всех сервисов в вашем проекте. Используйте команды Docker и Docker Compose для просмотра логов и статусов работы контейнеров.

Для просмотра логов конкретного сервиса, используйте команду:

docker-compose logs -f <имя\_сервиса>

где <имя\_сервиса> — имя сервиса, указанное в файле docker-compose.yml.

**Обновление и масштабирование**

В случае необходимости обновления или масштабирования вашего приложения, Docker Compose облегчает этот процесс. Для обновления образов и перезапуска контейнеров используйте команды:

docker-compose down

docker-compose pull

docker-compose up -d

Эти команды остановят текущие контейнеры, загрузят последние версии образов (если они доступны) и запустят контейнеры заново.

## Подготовка к запуску

**Установка Node.js и NPM**:

Убедитесь, что на вашем компьютере установлен Node.js версии не ниже 14.0.0. Если нет, скачайте и установите его с официального сайта <https://nodejs.org/>.

При установке Node.js, NPM (Node Package Manager) будет установлен автоматически.

**Скачивание и подготовка программы**:

Скачайте и распакуйте архив с программой или клонируйте репозиторий с помощью Git: git clone <URL\_репозитория>.

Откройте командную строку или терминал и перейдите в каталог с программой.

**Установка зависимостей**:

Введите команду npm install для установки необходимых зависимостей, указанных в файле package.json.

## Запуск сервиса

**Запуск сервера**:

В командной строке или терминале, находясь в каталоге с программой, введите команду docker compose up.

Сервис будет запущен и начнет слушать порт, указанный в файле конфигурации (по умолчанию 3000).

После запуска вы увидите сообщение, что сервер успешно запущен и слушает соответствующий порт, при этом должно вывестись сообщение об успешном запуске докер-контейнера.

## Использование API

* Получение информации о пользователе

**Запрос**:

curl --location --request GET 'http://localhost:3000/api/master/get-user' \

--header 'Content-Type: application/json' \

--data '{

"contest\_id" : 1,

"other\_user\_id" : 1

}'

* Получение информации о контесте

**Запрос**:

curl --location 'http://localhost:3000/api/master/contest-status-json?contest\_id=1'

**Ответ**:

{

"message": "Contest status fetched successfully",

"data": {

"ok": true,

...

}

}

* Получение информации о задаче

**Запрос**:

curl --location 'http://localhost:3000/api/master/problem-statement-json?contest\_id=1&problem=1'

**Ответ**:

{

"message": "Problem statement fetched successfully",

"data": {

"title": "Задача A",

...

}

}

* Отправка задачи на проверку

**Запрос**:

curl --location 'http://localhost:3000/api/master/submit-run' \

--form 'contest\_id="1"' \

--form 'problem="1"' \

--form 'lang\_id="23"' \

--form 'file=@"/path/to/your/file.py"'

**Ответ**:

{

"message": "Submission handled successfully",

"data": {

"ok": true,

...

}

}

* Получение информации о посылке

**Запрос**:

curl --location 'http://localhost:3000/api/master/get-submit?contest\_id=1&run\_id=2'

**Ответ**:

{

"message": "Submit info fetched successfully",

"data": {

"ok": true,

...

}

}

* Регистрация пользователей из CSV

**Запрос**:

curl -X POST http://localhost:3000/api/register-users \

-F "file=@/path/to/your/file.csv" \

-F "userType=student"

**CSV Структура файла**:

Для студентов: first\_name,last\_name,middle\_name,email,role,password,department,group

Для преподавателей: first\_name,last\_name,middle\_name,email,role,password

## Остановка сервиса

* Чтобы остановить сервис, воспользуйтесь командой docker ps -a для получения списка контейнеров, найдите id нужного контейнера и пропишите команду docker stop <id>

## Обработка ошибок

* В случае возникновения ошибок во время работы сервиса они будут выведены в командную строку или терминал.
* В зависимости от типа ошибки может потребоваться перезапуск сервиса или дополнительная диагностика проблемы.
* Для более подробного анализа и логирования ошибок рекомендуется использовать встроенные механизмы Node.js и Puppeteer, а также дополнительные библиотеки для логирования, такие как Winston или Morgan.

# СООБЩЕНИЯ ОПЕРАТОРУ

* **Ошибка аутентификации пользователя**: Если при попытке аутентификации пользователя на сайте Ejudge происходит ошибка, оператору будет выведено сообщение: "Ошибка аутентификации. Проверьте введенные данные и попробуйте снова".
* **Ошибка при получении списка задач**: Если при попытке получить список задач из Ejudge происходит ошибка, оператору будет выведено сообщение: "Не удалось получить список задач. Пожалуйста, проверьте доступность Ejudge и попробуйте снова".
* **Ошибка при отправке решения**: Если при попытке отправить решение на проверку происходит ошибка, оператору будет выведено сообщение: "Не удалось отправить решение. Пожалуйста, проверьте соединение с сервером Ejudge и попробуйте снова".
* **Ошибка при получении статуса решения**: Если при запросе информации о статусе решения возникает ошибка, оператору будет выведено сообщение: "Не удалось получить информацию о статусе решения. Пожалуйста, попробуйте позже".
* **Ошибка загрузки файла**: Если при попытке загрузить файл решения или CSV файла для регистрации пользователей возникает ошибка, оператору будет выведено сообщение: "Ошибка загрузки файла. Пожалуйста, проверьте формат файла и попробуйте снова".
* **Ошибка сети**: Если возникают проблемы с интернет-соединением или сервером во время выполнения запроса, оператору будет выведено сообщение: "Ошибка сети. Пожалуйста, проверьте ваше интернет-соединение".
* **Ошибка сервера**: Если при обработке запроса сервером происходит ошибка, оператору будет выведено сообщение: "Ошибка сервера. Пожалуйста, попробуйте позже".
* **Некорректные данные запроса**: Если данные, отправленные в запросе, некорректны или неполны, оператору будет выведено сообщение: "Некорректные данные запроса. Пожалуйста, проверьте введенные данные и попробуйте снова".
* **Неизвестная ошибка**: Если возникает неизвестная ошибка или исключение, оператору будет выведено сообщение: "Произошла неизвестная ошибка. Пожалуйста, обратитесь к разработчику за помощью".

Во всех случаях программа работает корректно и не завершается аварийно. Если возникают проблемы, оператору следует обратиться к соответствующим инструкциям или запросить помощь у администрации ресурса.

# ГЛОССАРИЙ

* + 1. **Авторизация** - Процесс проверки удостоверения пользователя, который позволяет системе узнать, кому предоставлять доступ к определенным ресурсам или функциям. В данном контексте это включает в себя ввод электронной почты и пароля.
    2. **Puppeteer** - Библиотека Node.js, которая предоставляет высокоуровневое API для управления браузером Chrome или Chromium через DevTools Protocol. Используется для автоматизации действий в браузере, таких как парсинг веб-страниц.
    3. **Ejudge** - Система для проведения онлайн-олимпиад и конкурсов по программированию, которая позволяет автоматически проверять решения задач, отправленные участниками.
    4. **API (Application Programming Interface)** - Интерфейс программирования приложений, набор определений и протоколов для разработки и интеграции программного обеспечения. API позволяет вашему приложению взаимодействовать с внешними сервисами или компонентами без необходимости знать детали их реализации.
    5. **CSV (Comma-Separated Values)** - Формат файла, используемый для хранения табличных данных. Каждая строка файла является одной записью данных, а столбцы разделены запятыми или другими разделителями.
    6. **Docker** - Платформа для разработки, доставки и запуска приложений в контейнерах, что обеспечивает их легкое и надежное развертывание.
    7. **Docker Compose** - Инструмент для определения и запуска многоконтейнерных Docker приложений, позволяет с помощью YAML-файла конфигурации настроить службы вашего приложения.
    8. **Node.js** - Среда выполнения для JavaScript на стороне сервера, которая позволяет разработчикам создавать масштабируемые веб-приложения, используя JavaScript для всей стековой технологии.
    9. **NPM (Node Package Manager)** - Менеджер пакетов для языка программирования JavaScript, позволяет устанавливать, обновлять и управлять зависимостями в проектах Node.js.
    10. **Rate Limiting** - Механизм, который ограничивает количество запросов, которые могут быть отправлены к API за определенный период времени, для предотвращения перегрузки сервера.
    11. **Middleware** - Промежуточное программное обеспечение, которое обрабатывает запросы перед их доставкой до конечного обработчика на сервере. В контексте веб-разработки middleware часто используется для выполнения аутентификации, логирования, обработки ошибок и других функций.
    12. **Маршрутизация (Routing)** - Процесс определения того, как приложение отвечает на клиентский запрос к определенному конечному точкам URI (Uniform Resource Identifier), которые могут включать в себя HTTP методы, такие как GET, POST и т.д.

# ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Лист регистрации изменений | | | | | | | | | |
| Номера листов (страниц) | | | | | Всего листов (страниц в докум.) | № документа | Входящий № сопроводительного докум. и дата | Подп. | Дата |
| Изм. | Измененных | Замененных | Новых | Аннулированных |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |