Приложение А. Техническое задание

### 1 Введение

Серверное программное обеспечение необходимо для выгрузки, сборки и запуска консольных приложений удалённой среде.

### 2 Основания для разработки

Задание на выпускную квалификационную работу было выдано научным руководителем: необходимо разработать серверное программное обеспечение, которое позволяет загружать проекты на сервер, производить сборку и запуск.

### 3 Назначение программы

Данное программное обеспечение предназначено для использования в учебной сфере, облегчая процесс проверки программ студентов.

### 4 Требование к программе

### 4.1 Требования к функциональным характеристикам

### 4.1.1 Требования к составу выполняемых функций

Программа должна обеспечивать функции:

* + Загрузка проектов в формате zip, загрузка конфигураций проектов, загрузка спецификации сборки, загрузка файлов в формате txt для тестирования загруженных проектов
  + хранение данных должно производиться в базе данных на сервере
  + просмотр и редактирование данных, загруженных в систему
  + возможность обновления и удавления
  + сборка загруженного проекта
  + запуск загруженного проекта
  + интерактивное взаимодействие - ввод данных в запущенный проект

### 4.1.2 Требование к организации входных данных

Входными данными для программы:

* + архивы в формате zip, содержащие проекты
  + данные формате txt
  + строки в формате json, для обращение к серверу

### 4.1.3 Требование к формированию ответа сервера

Сервер должен формировать ответ на каждый пользовательский запрос указывая статус ответа.

Ответы сервера

* + 200 OK, сервер успешно выполнил операцию
  + 403 Forbidden, у пользователя не достаточно прав для выполнения операции
  + 400 Bad Request, сервер не смог понять запрос

### 4.2 Требования к надёжности

### 4.2.1 Требования к обеспечению надёжного устойчивого функционирования программы

Разрабатываемый программный продукт должен обеспечивать устойчивое функционирование за исключением сбоев из-за неисправности вычислительной системы или системного программного обеспечения.

Устойчивое функционирование программы должно быть обеспечено выполнением совокупности определённых организационно-технических мероприятий:

* + организация постоянного питания технических средств;
  + подключение к сети Internet;
  + использования лицензионного или свободно распространяемого программного обеспечения;
  + обеспечения доступа пользователя с допустимыми характеристиками и навыками.

### 4.2.2 Время восстановления после отказа

Время восстановления после отказа, вызванного сбоем электропитания

технических средств или другими внешними факторами, не должно превышать времени на перезагрузку операционной системы и повторный ззапуск приложения.

Время восстановления после отказа, вызванного неисправностью

технических средств, например критическим сбоем операционной системы, не должно превышать времени, необходимого на устранение неисправностей технических средств и переустановки программных компонентов.

### 4.2.3 Отказы из-за некорректных действий оператора

Отказы программы вследствие некорректных действий пользователя при

взаимодействии с операционной системой должны быть исключены при выполнении условий эксплуатации программных и технических средств и строгому следованию руководству пользователя.

# 4.3 Условия эксплуатации

### 4.3.1 Климатические условия эксплуатации

Климатические условия эксплуатации, при которых должны обеспечиваться заданные характеристики, должны удовлетворять требованиям, предъявляемым к техническим средствам в части условий их эксплуатации.

### 4.3.2 Требование к видам обслуживая

Разрабатываемая программа не требует какого либо обслуживания.

#### **4.4 Требование к составу и параметрам технических средст**в

В состав технических средств должен входить сервер, на котором развёрнуто приложение и персональный компьютер.

Для обеспечения работоспособности требуются следующие технические характеристики:

* + Intel Pentium4 / Athlon 64;
  + оперативная память объёмом не менее 4Гб;
  + жёсткий диск объёмом 30Гб и вышe;
  + процессор с частотой не менее 2.5ГГц, а также с не менее 4-мя логическими ядрами
  + не менее 8Гб оперативной памяти
  + твердотельный накопитель с объёмом не менее 50Гб;
  + поддержка виртуализации

### 4.5 Требование к программным средствам используемым программой

Внутри операционной системы должен быть установлен Docker версии 19.3 и выше, компилятор языка программирования Go версии 18.1 и выше, Node JS версии 18.1 и выше.

### 4.6 Специальные требования

Специальные требования к программе не предъявляются.

### 4.6 Требование к документации программного продукта

Состав программной документации должен включать в себя

* + Техническое задание
  + Руководство программиста
  + Руководство пользователя

ПРИЛОЖЕНИЕ Б. РУКОВОДСТВО ПРОГРАММИСТА

### 1 Назначение программы

Разрабатываемый программный продукт предназначен для решения задачи автоматизированной сборки и выполнении консольных приложений пользователя, написанных на различных языках программирования, с возможностью тестирования проектов на заготовленных данных, или же применяя ручной ввод.

### 2 Структура программного решения

### 2.1 Описание уровней приложения

В связи со спецификой разработки web-приложений программа разделена на несколько уровней, на каждом уровне описаны свои структуры данных.

Язык программирования Go, выбранный для реализации проекта, не является объектно ориентированным языком, вместо объектов язык содержит структуры в которые выделяются функции, используемые в приложении. Структуры имеют схожее назначение, как и классы в объектно-ориентированных языках, поэтому для программы на языке go также возможно нарисовать диаграмму классов. Полная диаграмма классов представлена на рисунке.

Язык Go не имеет классов, в нем отсутствует механизм наследования, однако в языке есть схожий механизм, который называется “type embedding” или вложение, данный механизм имеет схожий функционал для разработчика, что и наследование. В целом не смотря на то, что объектов в языке нет, при разработке на данном языке применимы все паттерны проектирования из ООП и SOLID принципы, поэтому в руководстве пользователя используются классические термины из ООП.

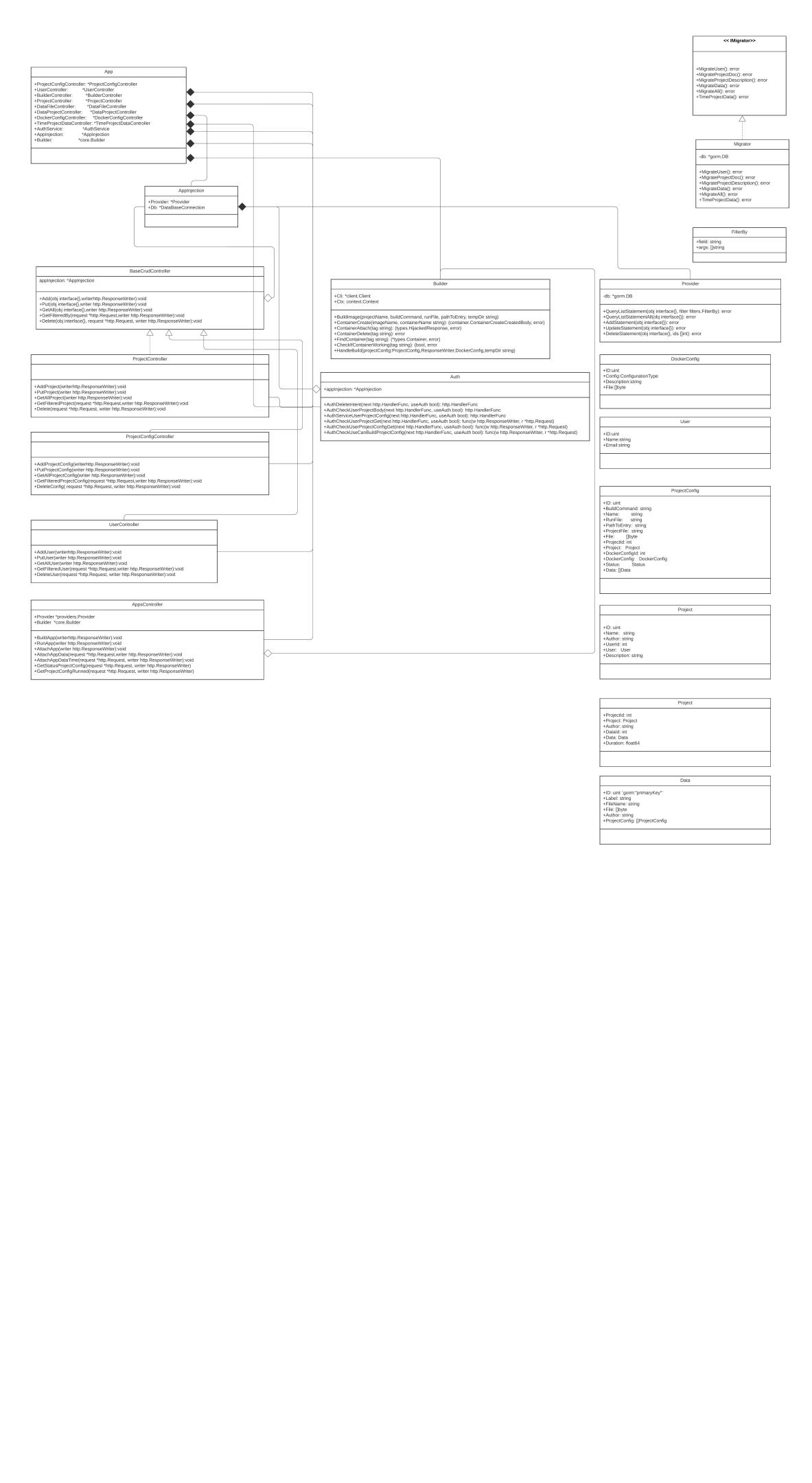


Рисунок 1 - полная диаграмма классов

### 2.2 Уровень представления

Уровень представления отвечает за кодирование декодирование данных, в проектируемом приложении на данном уровне должны обрабатываться пользовательские запросы с помощью специальных классов контроллеров.

В приложении должны быть предусмотрены следующие контроллеры

* + Контроллер сборки
  + Базовый контроллер
  + Контроллер проектов
  + Контроллер конфигураций проектов
  + Контроллер пользователей
  + Контроллер тестовых данных
  + Контроллер метрик
  + Контроллер конфигураций сборки

Диаграмма классов уровня представления:

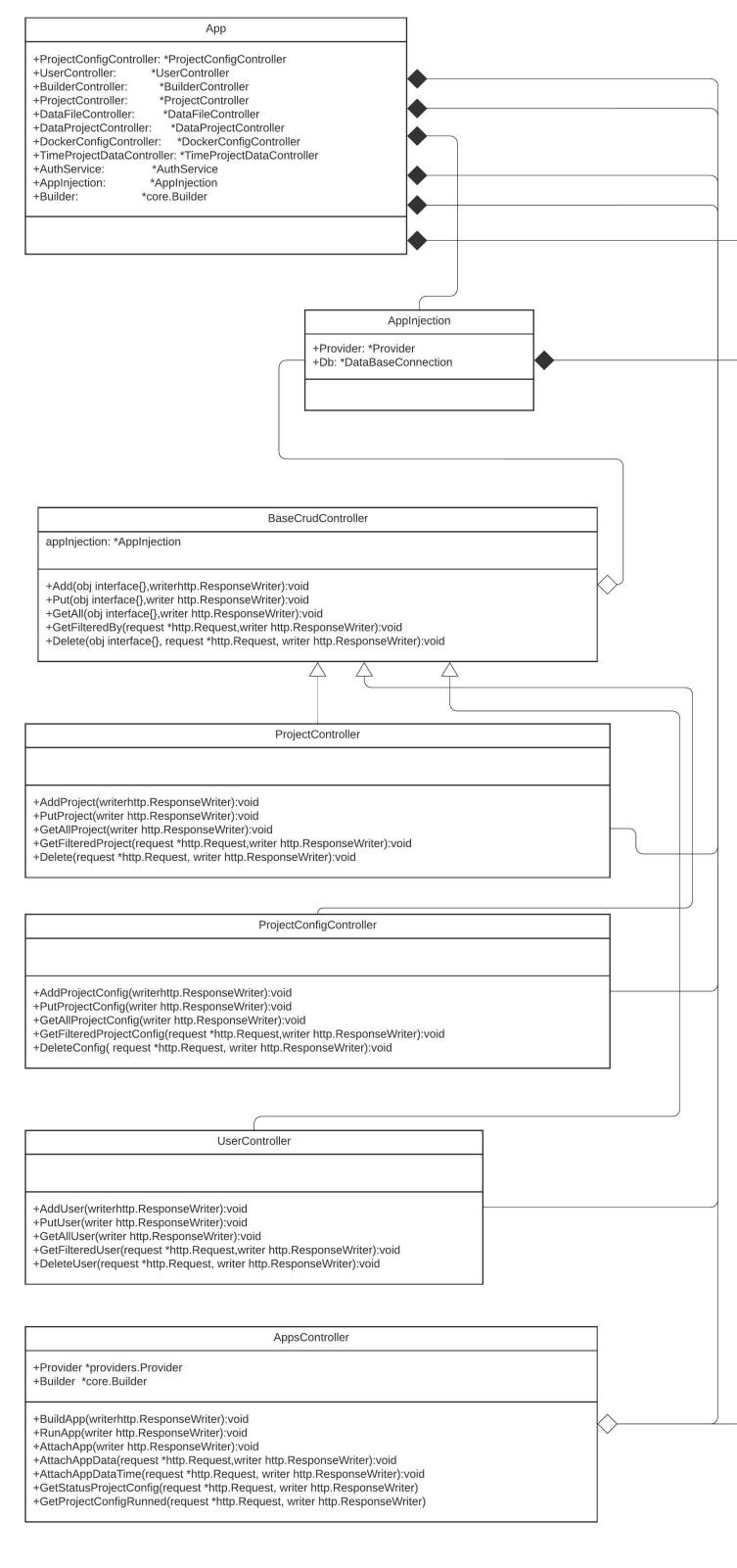


Рисунок 2 - диаграмма классов для уровня представления

### 2.3 Уровень бизнес логики

На данном уровне сосредоточена основная логика приложения, алгоритмы. На данном уровне находится сервис UserAppsService, отвечающий за сборку и запуск пользовательских приложений, а так же сервис Auth, выполняющий проверку авторизации пользователя.

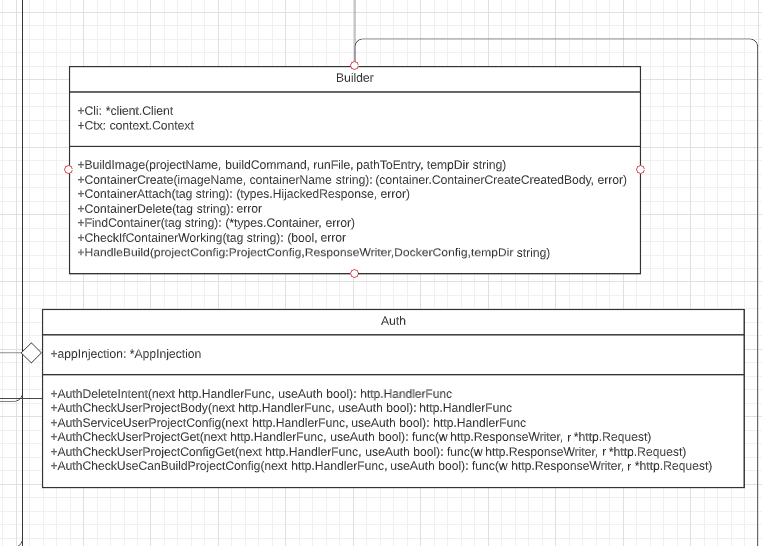


Рисунок 3 - Диаграмма классов уроня бизнес логики

### 2.4 Уровень работы с базой данных

На данном уровне сосредоточена логика работы с базой данных, данный уровень содержит структуру Provider, модели, а так же мигратор, с помощью которого создаются таблицы в базе данных.

### 3 Описание работы процедур

### 3.1 Принцип работы уровня представления

Поскольку программа представляет собой web приложение, введём некоторые понятия. Контроллер - класс или структура, которая имеет методы обрабатывающее пользовательские запросы, данные методы декодируют пользовательские запросы получая тем самым объекты, которые в свою очередь принимают участие в бизнес логике.

В программе используется базовый контроллер BaseCrudController, который в свою очередь содержит указатель на провайдер, используемый приложением. Данный контроллер содержит CRUD методы, использующие провайдер для добавления, удаления, обновления, чтения из базы данных. Другие контроллеры, в которые были “унаследованы” от базового контроллера, вызывают его методы при обработке запросов. Дочерние контроллеры не содержат ссылки на провайдер. Методу дочернего контроллера достаточно вызвать нужный метод базового контроллера. Получается, что метод базового контроллера как бы обернут в метод дочернего, это позволяет сконцентрироваться на дополнительной логике дочернего контроллера, не волнуясь о том как именно базовый контроллер использует провайдер.

Таким образом, все контроллеры “унаследованные” от базового имеют 5 методов, выполняющих следующие операции.

* + Добавление
  + Удаление
  + Обновление
  + Получения всех записей из базы данных
  + Получение записей из базы данных по фильтру

А реализация данных методов сосредоточена в базовом контроллере. Контроллер декодирует тело запроса в экземпляр того типа, с которым он умеет работать для этого вызывается функция Decode внутрь которой передаётся ссылка

Не все контроллеры унаследованы от базового, есть контроллеры которые обрабатывают специфические пользовательские запросы. Таким контроллером является AppsController. AppsController содержит указатель на провайдер и сервис сборки. Данный контроллер обрабатывает пользовательские запросы на сборку и запуск конкретного проекта в системе, кроме того, контроллер обрабатывает команды пользователя на ввод данных в запущенное приложения а так же команду на применение тестового набора данных. Таким образом, методы экземпляра структуры AppsController выполняют следующие операции:

* + Построение проекта
  + Запуск проекта
  + Получение статуса проекта
  + Запрос на ввод данных
  + Запрос на применения заготовленного набора данных

### 3.3 Принцип работы уровня работы с базой данных

Для упрощения работы с базой данных применяется технология, ORM.

ORM - object relation mapping, используя данную технологию таблицы базы данных, взаимосвязи создаются из структур описанных на языке go, другими словами в базе данных создаётся таблица, содержащая те поля, которые содержит структура, а внешние ключи генерируются благодаря типу полей. Если, структура A имеет поле типа B, то при создании сущности в базе данных сущность А будет иметь внешний ключ к таблице B. Индексы и первичные ключи так же описываются при определении полей структуры. Для использования orm, необходима библиотека, в проекте используется пакет gorm. Благодаря данному пакету экземпляр структуры провайдер позволяет легко без написания полного SQL запроса выполнять CRUD операции. На данном уровне так же присутствует структура Migrator. Экземпляр данной структуры имеет следующие методы:

* + MigrateUser
  + MigrateProjectDoc
  + MigrateProjectDescription
  + MigrateData
  + MigrateAll
  + TimeProjectData

Данные методы вызываются при запуске проекта в первый раз, если на момент запуска проекта в базе данных не существует таблиц, соответствующих данным структурам, таблицы будут созданы.

Модель базы данных представлена на рисунке

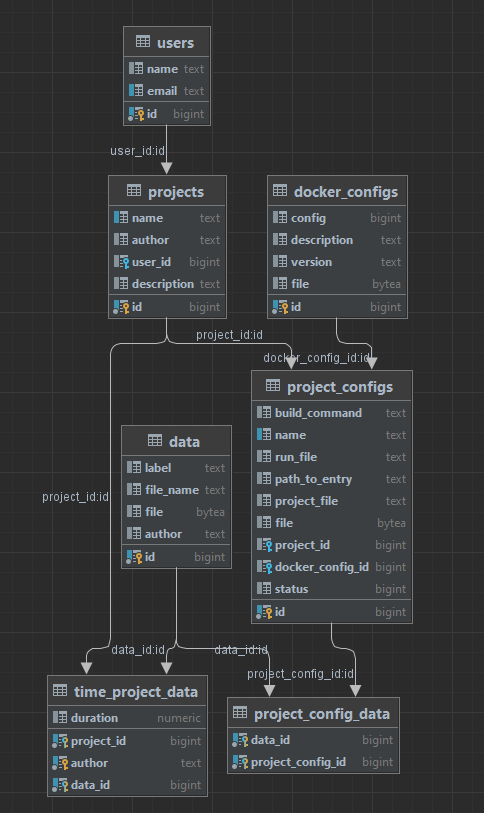


Рисунок - модель базы данных

Основными сущностями, необходимыми для работы с проектами пользователей являются Project и ProjectConfig. Сущность Project содержит название, описание, автора, добавившего проект, а так же ссылку на пользователя-автора, добавившего проект. ProjectConfig - сущность, которая содержит поля:

* Команда сборки
* Имя файла для запуска после сборки
* Путь к корню проекта внутри архива
* Наименование архива
* Содержание архива
* Внешний ключ на проект к которому относится данная конфигурация
* Внешний ключ на docker конфигурацию необходимую для сборки

# 3.3 Принцип работы уровня бизнес логики

На данном уровне находится сервис - Builder, данный сервис содержит методы для “управления” приложениями пользователя.

Главной задачей данного сервиса является сборка проекта. Для выполнения данной операции используется метод HandleBuild. Данный метод получает на вход экземпляры структур ProjectConfig, DockerConfig, ResponseWriter, наименование временно папке на компьютере. Данный метод выполняет следующие действия по цепочке.

Сперва происходит выгрузка архива проекта из базы данных во временную папку на сервере, затем, происходит процесс извлечения папки в архиве во временную папку. Затем к проекту применяется конфигурация docker - в корне проекта создаётся docker-файл. После чего вызывается метод BuildImage, после папка с проектом удаляется из временной папки. После чего архив удаляется.

Метод BuildImage, позволяет собрать проект в качестве входных параметров он принимает название проекта, команду сборки, имя исполняемого файла, путь внутри архива к корню проекта, а так же название локальной папки на севере в которой временно располагается проект.

В сервисе так же существуют метод для запуска.

Метод ContainerRun позволяет создать и запустить собранный проект в системе. Данный метод принимает на вход имя собираемого приложения и имя сборки.

Метод ContainerAttach создаёт соединение с запущенным приложением и возвращает указатель на данное соединение. В дальнейшем, используя данное соединение система имеет возможность записывать входные данные в пользовательское приложение и считывать результат.

В системе так же предусмотрены метод удаления запущенного приложения и метод проверки существования запущенного пользовательского приложения по имени. Другими словами данный сервис выполняет CRUD операции с приложениями пользователей, а так же предоставляет возможность получения соединения с конкретным приложением для ввода и вывода данных.

В приложении предусмотрена возможность использовать заранее заготовленные данные, для этого на уровне работы с базой данных реализованы структуры Data и TimeProjectData

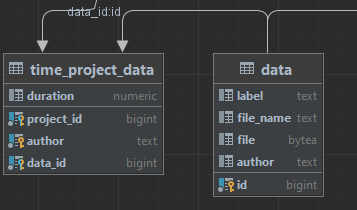


Рисунок - модели Data и TimeProjectData

Сущность TimeProjectData используется для сохранения времени работы приложения пользователя для проведения анализа.

### 3.4 Маршрутизация

Для добавления маршрутизации в реализуемом приложении присутствует функции AddRoutes, данный метод сопоставляет методы контроллеров и url.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Путь | Тело запроса | Http запрос | Результат | Возможные ошибки |
| /user | Структура User в формате json | POST | Структура -User в формате json | 403 Forbidden  400 BadRequest |
| /user | - | GET | Структура - массив User в формате json | 403 Forbidden  400 BadRequest |
| /user | Структура User в формате json | DELETE | - | 403 Forbidden  400 BadRequest |
| /user | Структура User в формате json | PUT | Структура User в формате json | 403 Forbidden  400 BadRequest |
| /user/filter | - | GET | Структура - массив User в формате json | 403 Forbidden  400 BadRequest |
| /user/info | - | GET | Структура User в формате json | 403 Forbidden  400 BadRequest |
| /project\_config | Структура ProjectConfig в формате json | POST | Структура -ProjectConfig в формате json | 403 Forbidden  400 BadRequest |
| /project\_config | - | GET | Структура - массив ProjectConfig в формате json | 400 BadRequest |
| /project\_config | Структура ProjectConfig в формате json | DELETE | - | 403 Forbidden  400 BadRequest |
| /project\_config | Структура ProjectConfig в формате json | PUT | Структура ProjectConfig в формате json | 403 Forbidden  400 BadRequest |
| /project\_config/filter | - | GET | Структура - массив ProjectConfig в формате json | 400 BadRequest |
| /project\_config/file | Экземпляр структуры FormData содержащий поле Fail и id конфигурации | GET | Структура - массив Project в формате json | 400 BadRequest |
| /project | Структура Project в формате json | POST | Структура -Project в формате json | 403 Forbidden  400 BadRequest |
| /project | - | GET | Структура - массив Project в формате json | 400 BadRequest |
| /project | Структура Project в формате json | DELETE | - | 403 Forbidden  400 BadRequest |
| /project | Структура Project в формате json | PUT | Структура Project в формате json | 403 Forbidden  400 BadRequest |
| /project/filter | - | GET | Структура - массив Project в формате json | 400 BadRequest |
| /docker\_config/all | - | GET | Структура - массив ProjectConfig в формате json | 400 BadRequest |
| /docker\_config | Экземпляр структуры FormData, содержащий все поля структуры DockerConfig | POST | Структура - ProjectConfig в формате json | 400 BadRequest |
| /data | - | GET | Структура - массив Data в формате json | 400 BadRequest |
| /data | Экземпляр структуры FormData, содержащий все поля структуры DockerConfig | POST | Структура - массив Data в формате json | 400 BadRequest |
| /data | Экземпляр структуры FormData, содержащий все поля структуры DockerConfig | DELETE | Структура - массив Data в формате json | 400 BadRequest |
| /data/filter | - | GET | Структура - массив Data в формате json | 400 BadRequest |
| /data/filter/project\_config | - | GET | Структура - массив Data в формате json | 400 BadRequest |
| /data/content | - | GET | Text | 400 BadRequest |
| /builder/build | Поле id в заголовке | POST | - | 403 Forbidden  400 BadRequest |
| /builder/run | Экземпляр структуры RunIntent в формате json | POST | - | 400 BadRequest |
| /builder/attach | Экземпляр структуры AttachIntent в формате json | POST | - | 400 BadRequest |
| /builder/attach/data | Экземпляр структуры AttachIntentData в формате json | POST | - | 400 BadRequest |
| /builder/status | - | GET | text | 400 BadRequest |
| /builder/is\_rinning | - | GET | text | 400 BadRequest |
| /builder/attach/data/time | Экземпляр структуры AttachIntentDataTime в формате json | POST | - | 403 Forbidden  400 BadRequest |
| /project\_data | Экземпляр структуры ProjectData в формате json | POST | Структура DataProject в формате json | 400 BadRequest |
| /project\_data | Экземпляр структуры ProjectData в формате json | DELETE | - | 400 BadRequest |
| /project\_data | - | GET | Структура DataProject в формате json | 400 BadRequest |

### 3.5 Внедрение зависимостей

Для внедрения зависимостей в приложении присутствует структура AppInjection. Все контроллеры содержат указатель на экземпляр данной структуры.

В данной структуре имеет указатель на провайдер, и указатель на структуру-подключение к базе данных. Данные указатели необходимы для работы с базой данных. Другими словами структура appInjection служит хранилищем зависимостей для дальнейшего внедрения их в другие части приложения.

### 3.6 Авторизация и Middleware

Для проведения авторизации в приложении используется jwt токен, чтобы получить токен клиент обращается сперва в сервис авторизации, затем при наличии пользователя в системе сервис авторизации вернёт токен, который содержит имя пользователя, а так же срок действия токена.

Для проверки доступа пользователя к тем или иным ресурсам в приложении предусмотрен middleware. Middleware - это прослойка между вызовом метода контроллера и рутингом, это некоторая функция которая сперва обрабатывает запрос, проверяет доступен ли пользователю тот или иной ресурс и предоставляет доступ, передавая управление, запрашиваемому методу или же прерывает выполнение запроса. Для реализации подобного middleware используется сервис проверки доступа, в модели приложения он находится на уровне бизнес логики. Структура сервиса авторизации представлен на рисунке:

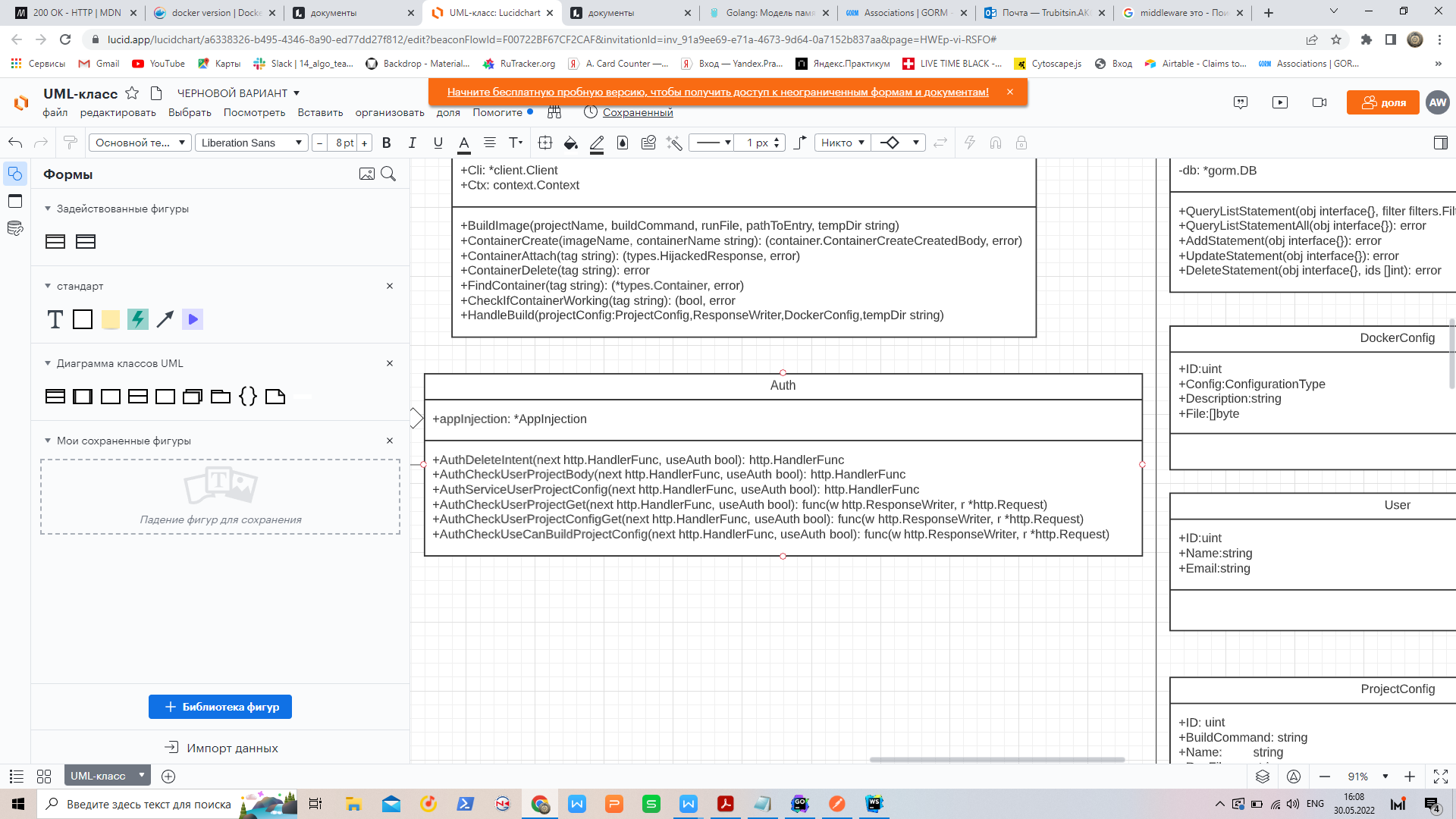


Рисунок - сервис проверки доступа

Данная структура содержит методы для проверки доступа к конкретным конечным точкам, поэтому для каждой “защищённой” контрольной точки реализуется свой метод, некоторые методы, например AuthCheckUserProjectBody используются для “защиты” нескольких конечных точек.

#### 3.7 Описание сообщений, выводимых программисту

В ходе выполнения программы предусмотрен стандартный механизм логирования ошибок

|  |  |
| --- | --- |
| Событие | Сообщение |
| Возникновение необработанной ошибки | Глобальная ошибка: (тест ошибки) |

Приложение В. РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

# 1 Функциональное назначение программы

# 1.1 Условия использования программы

Программа предназначена для сборки запуска и выполнения консольных приложений пользователя в удалённой среде на сервере.

### 1.2 Эксплуатационное назначение программы

Ограничений на эксплуатацию программы не вводится.

### 1.3 Состав функций

Программа обеспечивает возможность выполнения перечисленных функций:

* выгрузка проекта пользователя в систему
* сборка приложения пользователя внутри системы
* запуск приложения пользователя внутри системы
* ввод данных в запущенное приложение пользователя
* загрузка тестовых наборов данных
* ввод тестовых наборов в запущенное приложение пользователя
* выгрузка проектов пользователей
* просмотр и выгрузка данных
* сохранение времени работы приложения в системе

### 2 Условия выполнения программы

### 2.1 Минимальный состав аппаратных средств

* + Intel Pentium 4 / Athlon 64 или более поздней версии с поддержкой SSE2;
  + оперативную память объемом не менее 2Гб;
  + жесткий диск объемом 20 Гб, и выше;
  + наличие устройств ввода (клавиатура, мышь/тачпад);
  + операционная система Windows 7+ / Mac OS X 10.6+ / Ubuntu 10.04+ / Debian 6+ и подобные;
  + стабильное интернет-соединение.

### 2.2 Минимальный состав программных средств

* + операционная система Windows 7+ / Mac OS X 10.6+ / Ubuntu 10.04+ / Debian 6+ и подобные;
  + современный веб-браузер: Google Chrome 49 и выше, Mozilla Firefox 49 и выше, Opera 40 и выше, Apple Safari 9 и выше, Яндекс.Браузер 16.9 и выше, Microsoft Edge Browser 14.14 и выше.

### 2.3 Требования к пользователю

Программа рассчитана на пользователя, имеющего базовые знания о работе за персональным компьютером, а также о решаемой программой задаче.

#### 3 Выполнение программы

Для выполнения всех описанных процедур не предусмотрен интерфейс пользователя, так как данная часть проекта реализуется в другом дипломе. В данном разделе рассмотрим те функции, которые вызывает пользователь на сервере.

### 3.1 Регистрация и авторизация

Для выгрузки проекта в систему пользователь сперва зарегистрироваться и войти в систему.

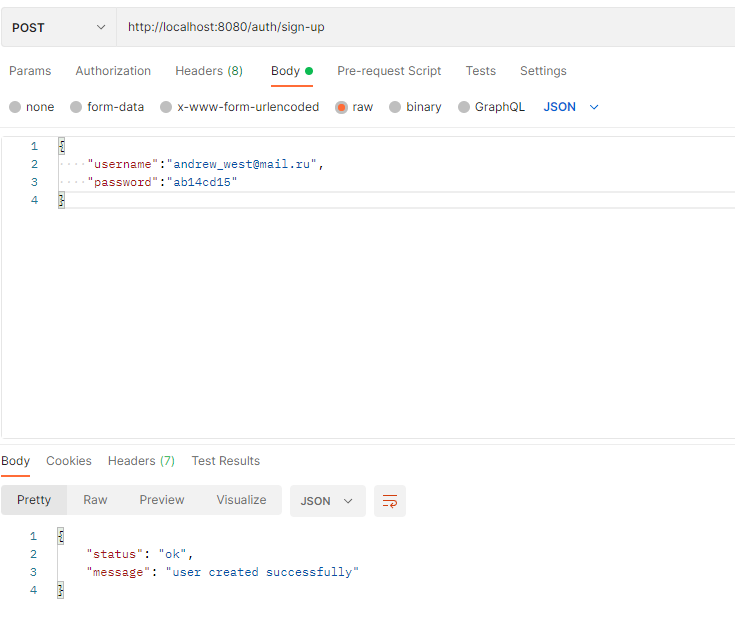


Рисунок - регистрация пользователя

После регистрации пользователь имеет возможность войти в систему.

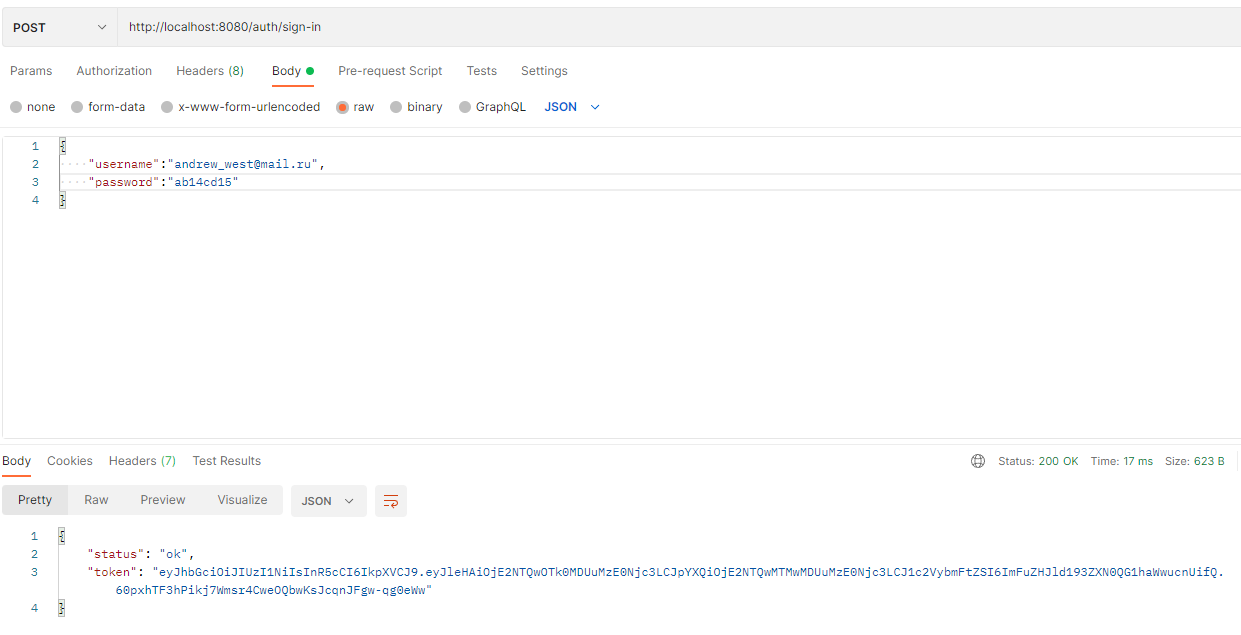


Рисунок - вход в систему

После входы пользователя в систему, пользователю возвращается токен, в дальнейшем токен используется в заголовках всех запросов к защищённым ресурсам, токен применяется автоматически.

После авторизации создаётся экземпляр структуры пользователя в системе, для этого смежная система использует конечная точка

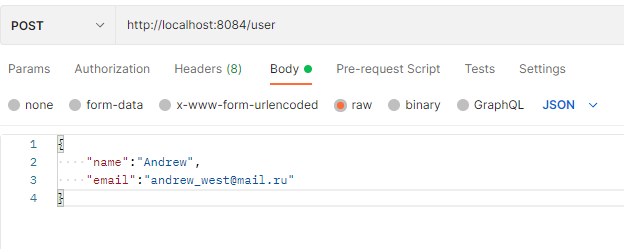


Рисунок - создание пользователя в системе и тело запроса

#### 3.2 Загрузка приложения в систему

Пользователь загружает приложение в систему, для этого необходимо указать название, описание. Id текущего пользователя после авторизации будет указано автоматически.

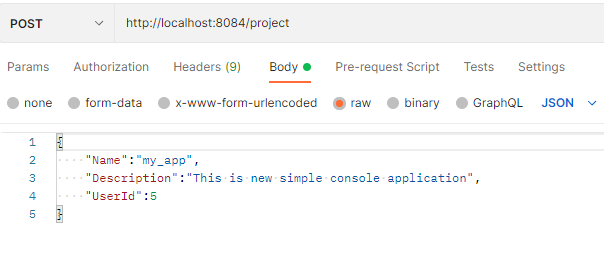


Рисунок - создание проекта в системе и тело запроса

### 3.3 Добавление конфигурации в систему

После создания проекта в системе, необходимо загрузить его конфигурацию. Конфигурация проекта содержит:

* + команду сборки
  + имя выполняемого файла после сборки
  + путь к корню проекта пользователя
  + id используемой docker конфигурации
  + id проекта, к которому прикрепляется конфигурация
  + архив, содержащий проект

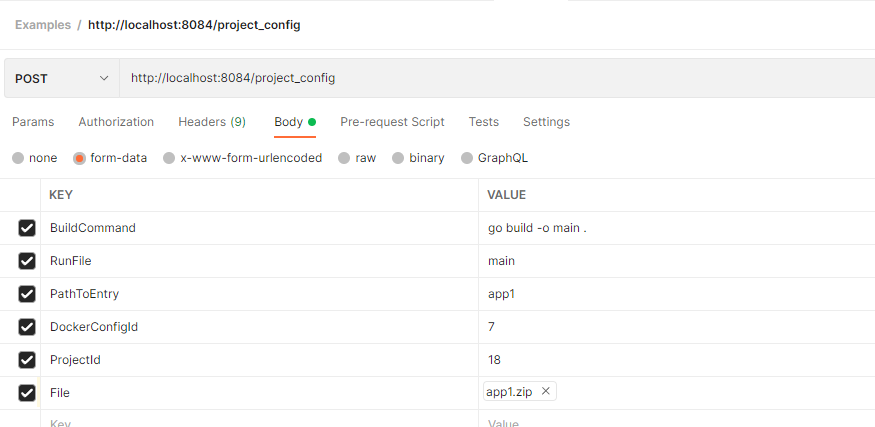


Рисунок - добавление конфигурации проекта и тело запроса

В данном руководстве в качестве проекта в систему загружается простое интуитивно понятное приложение, написанное на языке go.

Загружаемое приложение выполняет несколько операций.

Сначала использует стороннюю библиотеку и генерирует уникальный идентификатор с помощью сторонней библиотеки, затем принимает на вход два числа и выводит их сумму. После чего печатает текст из файла, который находится внутри проекта. Данный набор операций был выбран для демонстрации способности системы обеспечивать ввод данных в приложение пользователя, вывод данных, возможности использовать внутренние файлы, как источник данных, а так же для демонстрации способности системы поддерживать пользовательские приложения, имеющие зависимости от сторонних библиотек. Подробнее результат выполнения рассмотрен в разделе 3.6

### 3.4 Сборка проекта в системе

После конфигурации проекта, пользователь может собрать проект.

В системе предусмотрено ограничение прав на данную операцию, так только автор проекта может его собрать, после чего проект станет доступен всем остальным для запуска.

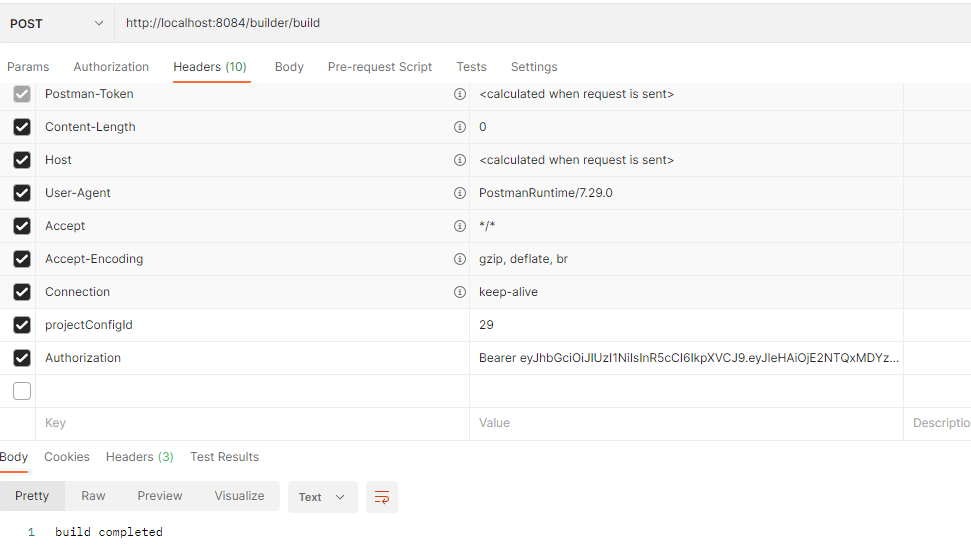


Рисунок - сборка проекта

### 3.5 Запуск проекта в системе

На данном этапе с перспективы пользователя система создаёт некое рабочее пространство для работы конкретного пользователя с приложением. В системе предусмотрена работа нескольких пользователей с ‘репликой’ одного приложения. В таком случае для каждого пользователя создаётся рабочее пространство. Для чтобы такое рабочее пространство создать необходимо сгенерировать имя рабочего пространства. Имя рабочего пространства складывается из названия проекта и уникального идентификатора пользователя.

Таким образом запрос к системе имеет следующий вид:

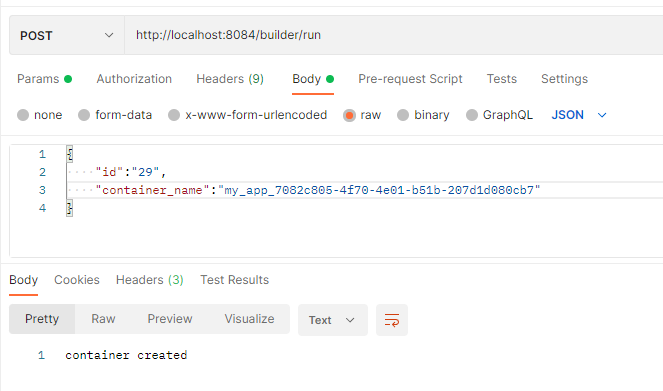


Рисунок - запуск проекта в рабочем пространстве

После выполнения запроса пользователь получает сообщение об успешном запуске.

### 3.6 Ручной ввод данных в приложение пользователя

В приложении предусмотрена возможность ручного ввода данных в запущенное приложение пользователя. Данную операцию выполняет следующий запрос.

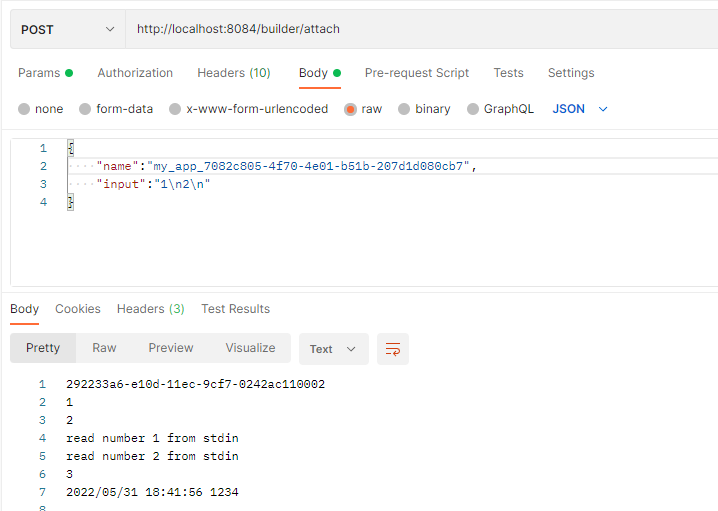


Рисунок - запрос на ручной ввод и тело запроса

Тело запроса содержит поле input, в котором находится текст, вводимый пользователем, а так же поле name - имя рабочего пространства.

В случае успешного выполнения запроса, система вернёт результат работы приложения пользователя.

Как видно на рисунке приложение пользователя, находясь в изолированной системе правильно сгенерировало уникальный идентификатор, используя внешнюю библиотеку, правильно считало входные параметры, применило математическую операцию и вывело данные из файла внутри проекта пользователя.

### 3.7 Выгрузка тестовых данных в систему

Источником данных для приложения пользователя может служить заранее заготовленный набор данных.

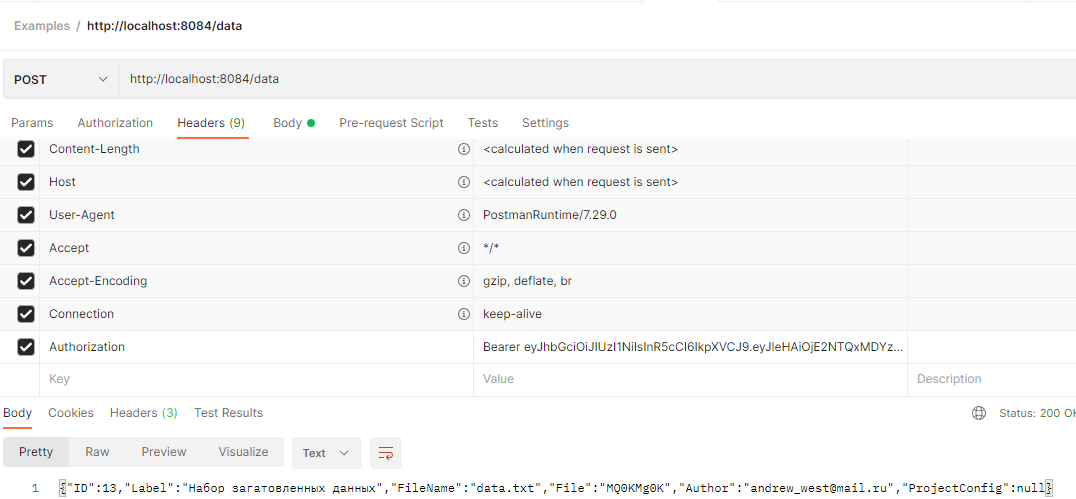


Рисунок - добавление заготовленных данных

### 3.8 Ввод тестовых данных в приложение пользователя

При наличии в системе тестовых данных, пользователю предоставляется возможность их применить в качестве альтернативы ручному вводу.

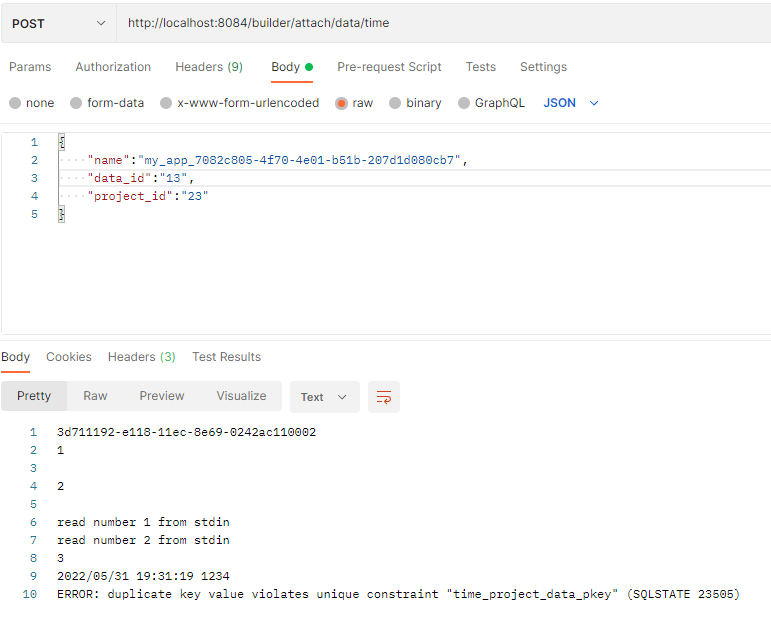


Рисунок - использование тестовых данных

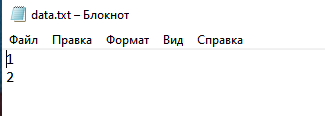


Рисунок - текст файла

В системе присутствует два вида запросов с применением заготовленных данных, которые доступны пользователю, отличие заключается в сохранении метрик - времени выполнения программы. Другими словами, если пользователь хочет сохранить в системе время работы на тестовых данных, он выполняет запрос по пути /builder/attach/data/time, в противном случае /builder/attach/data.

### 3.9 Получение времени выполнения пользовательского приложения

Система предоставляет пользователю возможность получать метрики - время выполнения программы на заготовленных данных, для этого необходимо выполнить запрос, показанный на рисунке.

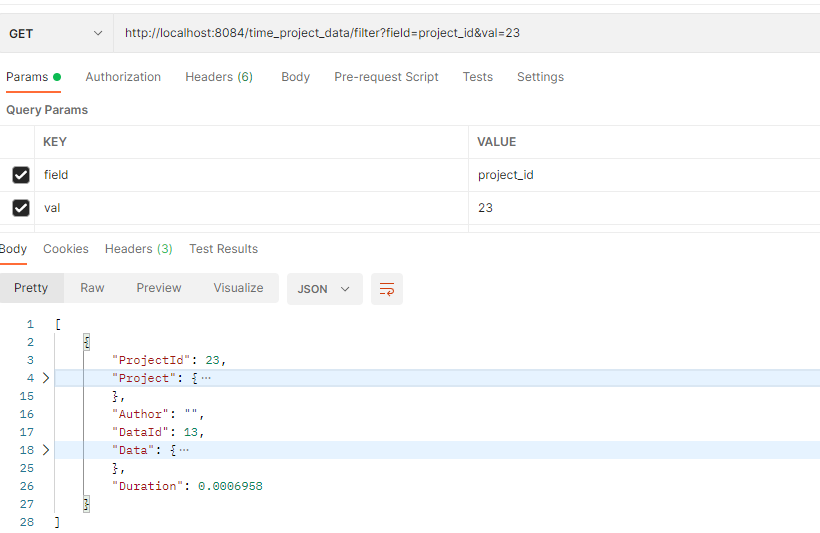


Рисунок - получение временных метрик

### 3.10 Особенности использования системы

### 3.10.1 Дополнительные функции, доступные пользователю

В разделах 3.1 - 3.9 руководства пользователя описан базовый сценарий применения системы. Помимо в системе присутствуют запросы на добавление, обновление, удаление, чтение всех элементов системы, с которыми взаимодействует пользователь. Для выполнения данных операций используются POST PUT DELETE GET запросы соответственно, кроме того в системе предусмотрены GET запросы для получения данных по фильтру для этого необходимо использовать GET запрос с добавление суффикса /filter?filed=#1&val=#2, где #1 и #2 название фильтруемого поля и его значение соответственно, таким образом реализован запрос на получения времени выполнения проекта на тестовых наборах данных на рисунке “получение временных метрик”.

### 3.10.2 Особенности работы с запущенным приложением

Поскольку для каждого пользователя выделяется отдельное рабочее пространство для работы с приложением, данном пространство никак не мешает другим пользователем и располагается в системе до тех пор, пока программа пользователя не завершит работы. После завершения работы пространство автоматически уничтожается. Для определения состояния рабочего пространства, в системе предусмотрены специальный метод, доступный пользователю. Данный метод вызывается с помощью запроса на рисунке.

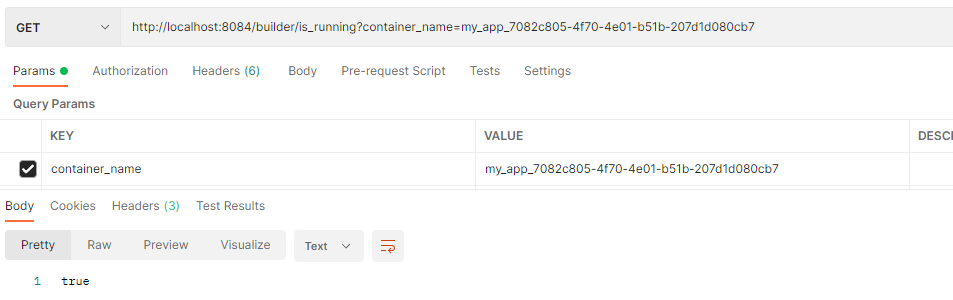


Рисунок - получение статуса рабочего пространства

### 4 Сообщения пользователю системы

Каждый запрос пользователя содержит ответ либо в виде структуры json, либо человеко-читаемый текст. Таким образом на каждый GET POST PUT запрос в случае успешного выполнения система выведет json структуру, содержащую запрашиваемый, добавляемый, обновляемый объект. При успешном удалении система не возвращает сообщений. При неудачном удалении система так же вернёт сообщение об ошибки.

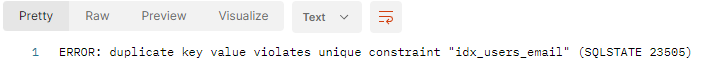


Рисунок - ошибка при некорректном добавлении пользователя

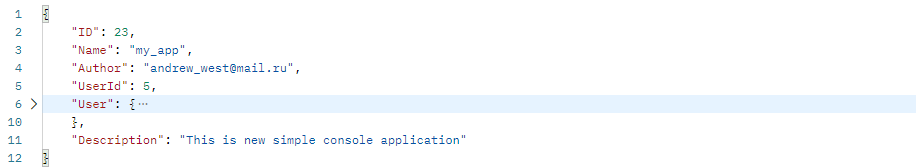


Рисунок - сообщение об успешном добавлении проекта

Сообщения после запросов на сборку, запуск, и ввод параметров отличаются.

В случае успешного выполнения сборки пользователю вернётся ответ “build completed”, в случае ошибки, вернётся текст ошибки.

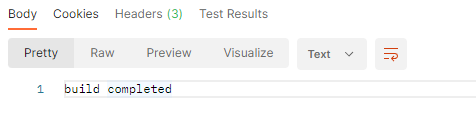


Рисунок - успешное выполнение сборки проекта

В случае успешного выполнения запуска, система вернёт сообщение “container created”, что будет свидетельствовать об успешном создании рабочего пространства пользователя.

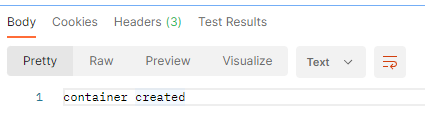


Рисунок - успешное создание контейнера

При использовании ввода параметров или заготовленного набора параметров в случае успешного выполнения пользователю вернётся ответ, содержащий результат вывода из приложения пользователя в противном случе сообщение об ошибке.

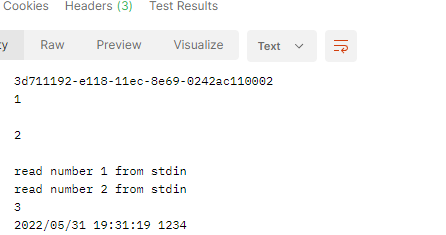


Рисунок - результат вывода приложения пользователя

# Оценка качества решения

### Тестирование ПО

Тестирование - это процесс запуска и выполнения программы для выявления неправильного функционирования программы или программных модулей.

Тестирование программного обеспечения производится в трёх условиях:

* + нормальных. В этом случае тестируется программа с входными данными из допустимого диапазона значения;
  + экстремальных. В этом случае тестирование производится на границах диапазона допустимых значений;
  + исключительных. Этот вид тестирования предполагает передаче программе заведомо не корректных данных, которые могут потенциально привести к её поломке.

Ниже в таблице приведено краткое описание того, что является для той или иной операции, допустимым (нормальным) или экстремальным значением, а также описаны исключительные ситуации. Рассмотрены основные операции в системы.

|  |  |  |  |
| --- | --- | --- | --- |
| Операция | Нормальное значение | Экстремальное значение | Исключительные ситуации |
| Создание пользователя | Длина строк в имени не превышает 255 символов | Длина строки в поле 0 или более 255 символов | - |
| Создание проекта | Длина строк в названии и описании не превышает 255 символов | Длина строки в поле 0 или более 255 символов | - |
| Удаление проекта | Запись с идентификатором , существует в системе | - | Запись по идентификатору не существует. |
| Обновление проекта | Длина строк в названии и описании не превышает 255 символов | Длина строки в поле 0 или более 255 символов |  |
| Чтение проекта по несуществующему идентификатору | Запись с идентификатором существует | - | Запись по идентификатору не существует |
| Создание конфигурации проекта | Длина строк во всех полях не превышает 255 символов | Длина строки в поле 0 или более 255 символов | - |
| Удаление конфигурации проекта | Запись с идентификатором , существует в системе | - | Запись по идентификатору не существует. |
| Обновление конфигурации проекта | Длина строк во всех полях не превышает 255 символов | Длина строки в поле 0 или более 255 символов |  |
| Создание docker конфигурации | Поле fail содержит файл | - | Поле fail не содержит файл |
| Удаление docker конфигурации | Запись с идентификатором , существует в системе | - | Запись по идентификатору не существует. |
| Обновление docker конфигурации | Поле fail содержит файл | - | Поле fail не содержит файл |
| Чтение docker конфигурации по id | Запись с идентификатором , существует в системе | - | Запись по идентификатору не существует. |
| Добавление тестовых данных | Поле fail содержит файл | - | Поле fail не содержит файл |
| Удаление тестовых данных | Запись с идентификатором , существует в системе | - | Запись по идентификатору не существует. |
| Сборка приложения | Конфигурация с идентификатором , существует в системе | - | Запись по идентификатору не существует. |
| Запуск приложения | Конфигурация с идентификатором , существует в системе | - | Конфигурация с идентификатором не существует в системе |
| Ручной ввод в приложение | Имя рабочего пространства существует в системе | - | Имя рабочего пространства не существует в системе |
| Ввод заготовленных данных в приложение | Идентификатор данных существует в приложении | - | Идентификатор данных не существует в приложении |

### 4.5 Тестирование в нормальных условиях

В данном разделе приводится тестирование в нормальных условиях - на данных из допустимого диапазона. На рисунках ниже приведены основные действия пользователя в системе.

При успешном добавлении проекта, программа вернёт json, содержащий структуру проекта

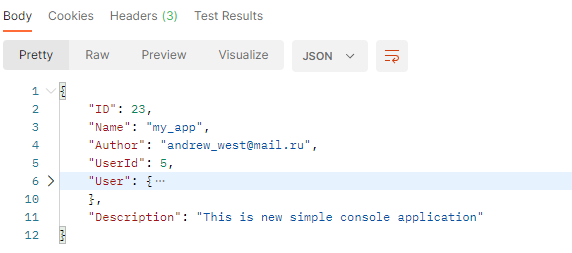


Рисунок - успешное добавление проекта

При успешном добавлении конфигурации проекта, программа вернёт json, содержащий структуру проекта

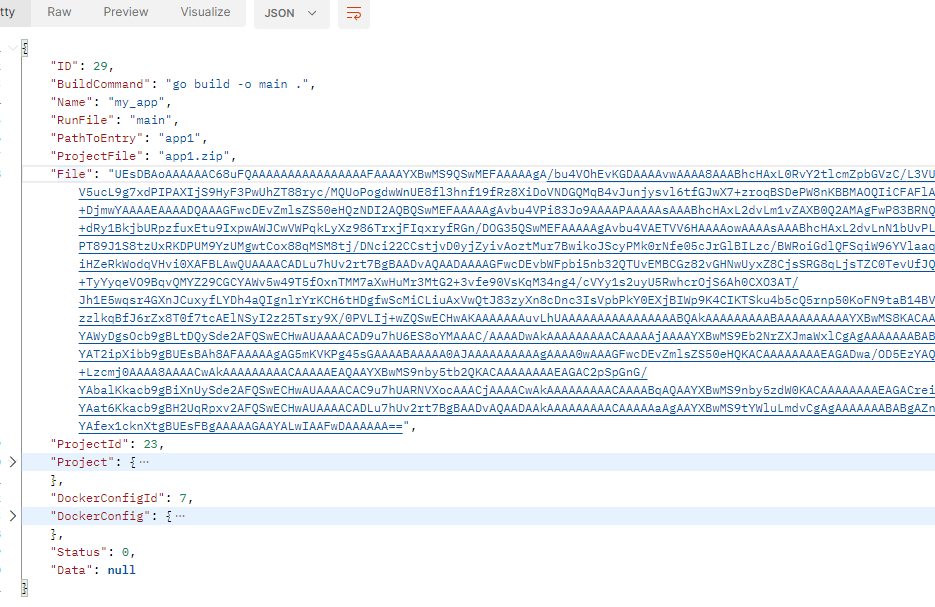


Рисунок - успешное добавление конфигурации

При выполнении сборки проекта в нормальных условиях программа возвращает ответ об успешном выполнении сборки.

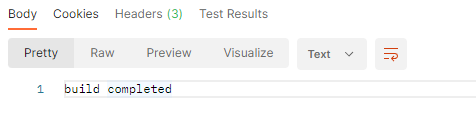


Рисунок - сообщение об успешном выполнении сборки

При успешном запуске проекта в нормальных условиях программа возвращает сообщение о создании рабочего пространства.

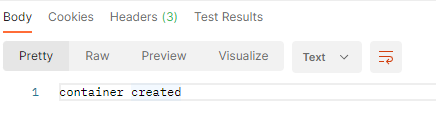


Рисунок - сообщение об успешном создании рабочего пространства

При использовании заранее заготовленного набора входных параметров или при ручном вводе программа вернёт из потока вывода результат работы приложения пользователя.

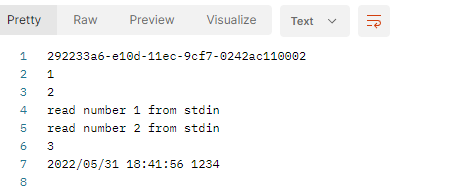


Рисунок - результат ввода параметров в нормальных условиях

### 4.6 Тестирование в экстремальных условиях

При тестировании в экстремальных условиях используются значениях на границах диапазона. В системе нет функций параметров которые соответствуют определённому диапазону, за исключением текстовых полей сущностей так.

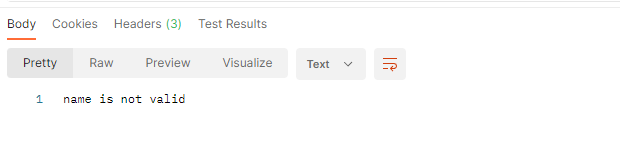


Рисунок - ответ системы при создании сущности с именем длиной 0 символов

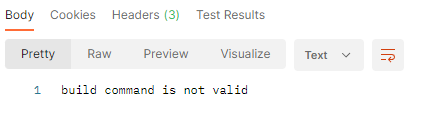


Рисунок - ответ системы при создании конфигурации с командой сборки длиной 0 символов

### 4.7 Тестирование в исключительных условиях

Тестирование в исключительных условиях проводится на заведомо некорректных данных. На рисунках ниже рассматриваются основные действия пользователя в системе. При вводе заведомо ошибочных значений система будет возвращать соответствующее сообщение об ошибки, однако продолжит свою работу.

При создании конфигурации проекта предполагается что пользователь может намеренно не прикрепить файл. В таком случае система вернёт следующий ответ.

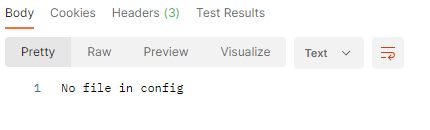


Рисунок - сообщение системы при создании конфигурации без файла

При неправильном заполнении поля projecConfiId(ссылка на проект) клиентским приложением во время использовании запроса на сборку, система вернёт следующий ответ.

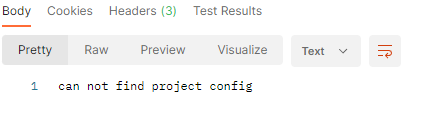


Рисунок - сообщение системы при неправильном заполнении поля projctConfig

При указании несуществующего идентификатора клиентским приложением во времени запуска, система вернёт следующий ответ.

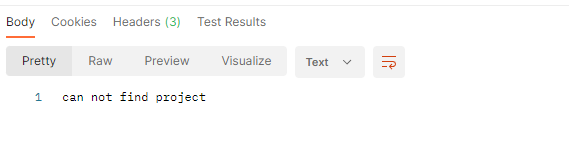


Рисунок - сообщение системы при использовании несуществующего в системе идентификатора проекта

В случае если клиентское приложение не верно указало идентификатор используемого файла, то система вернет следующее сообщение

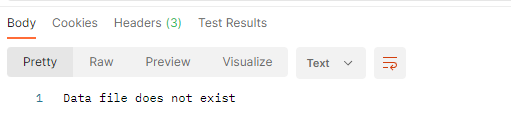


Рисунок - сообщение системы при указании неправильного идентификатора данных

### 4.8 Оценка полноты программы

Все основные методы уровней бизнес логики и работы с базой данных покрыты unit-тестами. Таким образом все методы чтения, записи, удаления и обновления из базы данных покрыты тестами, как и все методы, позволяющие собирать и запускать приложения.

Кроме того приложение содержит несколько вариантов проектов пользователей на разных языках программирования C# и Go, для тестирования сборки, запуска и ввода аргументов. Для проведения тестирования в проект так же были добавлен экземпляр приложения пользователя, содержащий внешние зависимости. Кроме того, в проекте присутствует тестовое приложение, с задержкой выполнения, данный проект необходим, для того чтобы протестировать функционал по подсчёту времени выполнения на тестовых данных.

Из выше описанного следует, что приложение в достаточной степени покрыто тестами. Показатели надёжности приведены в приложении Г.

### 4.9 Оценка качества программного обеспечения

Разработанный программный продукт относится к подклассу ПС 509 (прочие программные средства). Согласно ГОСТ 28195-89 «Оценка качества программных средств. Общие положения» к данному классу программных средств применяются все показатели качества из номенклатуры, приведенные в таблице 2. Результаты анализа и значения показателей содержатся в приложении Г.

Таблица 2 – Номенклатура показателей качества ПС

|  |  |  |
| --- | --- | --- |
| Наименование групп и комплексных показателей качества | Обозначение показателя | Характеризуемое свойство |
| 1. Показатели надежности ПС |  | Характеризуют способность ПС в конкретных областях применения выполнять заданные функции в соответствии с программными документами в условиях возникновения отклонений в среде функционирования, вызванных сбоями технических средств, ошибками во входных данных, ошибками обслуживания и другими дестабилизирующими воздействиями |
| 1.1. Устойчивость функционирования | H1 | Способность обеспечивать продолжение работы программы после возникновения отклонений, вызванных сбоями технических средств, ошибками во входных данных и ошибками обслуживания |
| 1.2. Работоспособность | Н2 | Способность программы функционировать в заданных режимах и объемах обрабатываемой информации в соответствии с программными документами при отсутствии сбоев технических средств |
| 2. Показатели сопровождения |  | Характеризуют технологические аспекты, обеспечивающие простоту устранения ошибок в программе и программных документах и поддержания ПС в актуальном состоянии |
| 2.1. Структурность | С1 | Организация всех взаимосвязанных частей программы в единое целое с использованием логических структур «последовательность», «выбор», «повторение» |
| 2.2. Простота конструкции | С2 | Построение модульной структуры программы наиболее рациональным с точки зрения восприятия и понимания образом |
| 2.3. Наглядность | С3 | Наличие и представление в наиболее легко воспринимаемом виде исходных модулей ПС, полное их описание в соответствующих программных документах |
| 2.4. Повторяемость | С4 | Степень использования типовых проектных решений или компонентов, входящих в ПС |
| 3. Показатели удобства применения |  | Характеризуют свойства ПС, способствующие быстрому освоению, применению и эксплуатации ПС с минимальными трудозатратами с учетом характера решаемых задач и требований к квалификации обслуживающего персонала |
| 3.1. Легкость освоения | У1 | Представление программных документов и программ в виде, способствующем пониманию логики функционирования программы в целом и ее частей |
| 3.2. Доступность эксплуатационных программных документов | У2 | Понятность, наглядность и полнота описания взаимодействия пользователя с программой в эксплуатационных программных документах |

Продолжение таблицы 2

|  |  |  |
| --- | --- | --- |
| 3.3. Удобство эксплуатации и обслуживания | У3 | Соответствие процесса обработки данных и форм представления результатов характеру решаемых задач |
| 4. Показатели эффективности |  | Характеризуют степень удовлетворения потребности пользователя в обработке данных с учетом экономических, вычислительных и людских ресурсов |
| 4.1. Уровень автоматизации | Э1 | Уровень автоматизации функций процесса обработки данных с учетом рациональности функциональной структуры программы с точки зрения взаимодействия с ней пользователя и использования вычислительных ресурсов |
| 4.2. Временная эффективность | Э2 | Способность программы выполнять заданные действия в интервал времени, отвечающий заданным требованиям |
| 4.3. Ресурсоемкость | Э3 | Минимально необходимые вычислительные ресурсы и число обслуживающего персонала для эксплуатации ПС |
| 5. Показатели универсальности |  | Характеризуют адаптируемость ПС к новым функциональным требованиям, возникающим вследствие изменения области применения или других условий функционирования |
| 5.1. Гибкость | Г1 | Возможность использования ПС в различных областях применения |
| 5.2. Мобильность | Г2 | Возможность применения ПС без существенных дополнительных трудозатрат на ЭВМ аналогичного класса |
| 5.3. Модифицируемость | Г3 | Обеспечение простоты внесения необходимых изменений и доработок в программу в процессе эксплуатации |
| 6. Показатели корректности |  | Характеризуют степень соответствия ПС требованиям, установленным в ТЗ, требованиям к обработке данных и общесистемным требованиям |
| 6.1. Полнота реализации | К1 | Полнота реализации заданных функций ПС и достаточность их описания в программной документации |
| 6.2. Согласованность | К2 | Однозначное, непротиворечивое описание и использование тождественных объектов, функций, терминов, определений, идентификаторов и т.д. в различных частях программных документов и текста программы |
| 6.3. Логическая корректность | К3 | Функциональное и программное соответствие процесса обработки данных при выполнении задания общесистемным требованиям |

После анализа показателей и факторов качества, рассчитанным в приложении Г, был сделан вывод, что

* + Наиболее низким показателем является показатель надёжности введу отсутствия средств по восстановления при сбоях оборудования, такого как процессор и программных компонентов, таких как операционная система.
  + Наиболее высоким показателем является показатель корректности, в веду полноты программной документации, её непротиворечивости. Все программные модули реализованы и подробно описаны.

Приложение Г

* 1. **Показатели надёжности**

Таблица Г.2 – Оценочные элементы фактора «Надежность ПС»

|  |  |  |  |
| --- | --- | --- | --- |
| Код элемента | Наименование | Метод оценки | Оценка |
| Средства восстановления при ошибках на входе | | | |
| Н0101 | Наличие требований к программе по устойчивости функционирования при наличии ошибок во входных данных | Экспертный | 1 |
| Н0102 | Возможность обработки ошибочных ситуаций | То же | 1 |
| Н0103 | Полнота обработки ошибочных ситуаций | » | 1 |
| Н0104 | Наличие тестов для проверки допустимых значений входных данных | » | 1 |
| Н0105 | Наличие системы контроля полноты входных данных | » | 1 |
| Н0106 | Наличие средств контроля корректности входных данных | » | 1 |
| Н0107 | Наличие средств контроля непротиворечивости входных данных | » | 0 |
| Н0108 | Наличие проверки параметров и адресов по диапазону их значений | » | 1 |
| Н0109 | Наличие обработки граничных результатов | » | 1 |
| Н0110 | Наличие обработки неопределенностей (деление на 0, квадратный корень из отрицательного числа и т.д.) | » | 1 |
|  |  | Итого: | 0,9 |
| Средства восстановления при сбоях оборудования | | | |
| Н0201 | Наличие требований к программе по восстановлению процесса выполнения в случае сбоя операционной системы, процессора, внешних устройств | » | 0 |

Таблица Г.2 – Оценочные элементы фактора «Надежность ПС»

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Н0202 | | Наличие требований к программе по восстановлению результатов при отказах процессора, ОС | | » | | 0 | |
| Н0203 | | Наличие средств восстановления процесса в случае сбоев оборудования | | » | | 0 | |
| Н0204 | | Наличие возможности разделения по времени выполнения отдельных функций программ | | » | | 0 | |
| Н0205 | | Наличие возможности повторного старта с точки останова | | » | | 0 | |
|  | |  | | Итого: | | 0 | |
| Реализация управления средствами восстановления | | | | | | | |
|  | |
| Н0301 | | Наличие централизованного управления процессами, конкурирующими из-за ресурсов | | » | | 0 | |
| Н0302 | | Наличие возможности автоматически обходить ошибочные ситуации в процессе вычисления | | » | | 1 | |
| Н0303 | | Наличие средств, обеспечивающих завершение процесса решения в случае помех | | » | | 0 | |
| Н0304 | | Наличие средств, обеспечивающих выполнение программы в сокращенном объеме в случае ошибок или помех | | » | | 0 | |
| Н0305 | | Показатель устойчивости к искажающим воздействиям | | Расчетный | | 1 | |
|  | |  | | Итого: | | 0,4 | |

|  |  |  |  |
| --- | --- | --- | --- |
| Функционирование в заданных режимах | | | |
| Н0401 | Вероятность безотказной работы | Расчетный | 0,9 |
|  |  | Итого: | 0,9 |
| Обеспечение обработки заданного объема информации | | | |
| Н0501 | Оценка по среднему времени восстановления | Расчетный | 1 |
| Н0502 | Оценка по продолжительности преобразования входного набора данных в выходной | Расчетный | 1 |
|  |  | Итого: | 1 |

* 1. **Показатели сопровождения**

Таблица Г.3 – Оценочные элементы фактора «Надежность ПС»

|  |  |  |  |
| --- | --- | --- | --- |
| Код элемента | Наименование | **Метод оценки** | **Оценка** |
| Простота архитектуры проекта | | | |
| С0101 | Наличие модульной схе­мы программы | Экспертный | 1 |
| С0102 | Оценка программы по числу уникальных модулей | » | 1 |
|  | | Итого: | 1 |
| Сложность архитектуры проекта | | | |
| С0201 | Наличие ограничений на размеры модуля | » | 0 |
|  |  | Итого: | 0 |
| Межмодульные связи | | | |
| С030 | Наличие требований к не­зависимости модулей про­граммы от типов и форма­тов выходных данных | » | 1 |
| С0301 | Наличие проверки кор­ректности передаваемых данных | » | 1 |
| С0302 | Оценка простоты программы по числу точек входа и выхода | Расчетный | 0,5 |
| С0303 | Осуществляется ли пере­дача результатов работы модуля через вызывающий его модуль | Экспертный | 0 |

|  |  |  |  |
| --- | --- | --- | --- |
| С0304 | Осуществляется ли конт­роль за правильностью дан­ных, поступающих в вызы­вающий модуль от вызыва­емого | » | 1 |
|  |  | Итого: | 0,7 |
| Экспертиза принятой системы идентификации | | | |
| С0601 | Использование при пост­роении программ метода структурного программиро­вания | » | 1 |
| С0602 | Соблюдение принципа разработки программы сверху вниз | » | 1 |
| С0603 | Оценка программы по числу циклов с одним вхо­дом и одним выходом | » | 1 |
| С0604 | Оценка программы по числу циклов | » | 1 |
|  |  | Итого: | 1 |
| Комментарии логики программ проекта | | | |
| С0801 | Наличие комментариев ко всем машинозависимым частям программы | » | 0 |
| С0802 | Наличие комментариев к машинозависимым операторам программы | » | 0 |
| С0803 | Наличие комментариев в точках входа и выхода программы | » | 0 |
|  |  | Итого: | 0 |

Продолжение таблицы Г.3

|  |  |  |  |
| --- | --- | --- | --- |
| Оформление текста программ | | | |
| С0901 | Соответствие комментари­ев принятым соглашениям | » | 0,5 |
| С0902 | Наличие комментариев-за­головков программы с ука­занием ее структурных и функциональных характе­ристик | » | 0,5 |
| С0903 | Оценка ясности и точнос­ти описания последователь­ности функционирования всех элементов программы | » | 0,5 |
|  |  | **Итого:** | **0,5** |
| **Простота кодирования** | | | |
| С1001 | Используется ли язык высокого уровня | Экспертный | 0.5 |
| С1002 | Оценка простоты прог­раммы по числу переходов по условию | Расчетный | 0,8 |
|  |  | **Итого:** | **0,7** |

**1.4 Показатели удобства применения**

Таблица Г.4 – Оценочные элементы фактора "Удобство применения ПС"

|  |  |  |  |
| --- | --- | --- | --- |
| Код элемента | Наименование | **Метод оценки** | **Оценка** |
| Освоение работы ПС | | | |
| У0101 | Возможность освоения программных средств по документации | Экспертный | 1 |
| У0102 | Возможность освоения ПС на контрольном примере при помощи ЭВМ | » | 1 |
| У0103 | Возможность поэтапного освоения ПС | » | 1 |
|  |  | Итого: | 1 |
| Документация для освоения | | | |
| У0201 | Полнота и понятность документации для освоения | » | 1 |
| У0202 | Точность документации для освоения | » | 1 |
| У0203 | Техническое исполнение документации | » | 1 |
|  |  | Итого: | 1 |
| Полнота пользовательской документации | | | |
| У0301 | Наличие краткой аннотации | » | 0 |
| У0302 | Наличие описания решаемых задач | » | 1 |
| У0303 | Наличие описания структуры функции ПС | » | 1 |
| У0304 | Наличие описания основных функций ПС | » | 1 |
|  |  | Итого: | 0,8 |

Продолжение таблицы Г.4.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| У0306 | Наличие описания частных функций | | » | 0 |
| У0307 | Наличие описания алгоритмов | | » | 0,7 |
| У0308 | Наличие описания межмодульных интерфейсов | | » | 0 |
| У0309 | Наличие описания пользовательских интерфейсов | | » | 1 |
| У0310 | Наличие описания входных и выходных данных | | » | 1 |
| У0311 | Наличие описания диагностических сообщений | | » | 1 |
| У0312 | Наличие описания основных характеристик ПС | | » | 1 |
| У0314 | Наличие описания программной среды функционирования ПС | | » | 1 |
| У0315 | Достаточность документации для ввода ПС в эксплуатацию | | » | 1 |
| У0316 | Наличие информации технологии переноса для мобильных программ | | » | 0 |
|  | | | Итого: | 0,7 |
| Точность пользовательской документации | | | | |
| У0401 | | Соответствие оглавления содержанию документации | » | 1 |
| У0402 | | Оценка оформления документации | » | 1 |
| У0403 | | Грамматическая правильность из­ложения документации | » | 1 |
| У0404 | | Отсутствие противоречий | » | 1 |
| У0405 | | Отсутствие неправильных ссылок | » | 1 |
| У0406 | | Ясность формулировок и описаний | » | 1 |
| У0407 | Отсутствие неоднозначных формулировок и описаний | | » | 1 |
| У0408 | Правильность использования тер­минов | | » | 1 |
| У0409 | Краткость, отсутствие лишней де­тализации | | » | 1 |
| У0410 | Единство формулировок | | » | 1 |
| У0411 | Единство обозначений | | » | 1 |
| У0412 | Отсутствие ненужных повторений | | » | 1 |
| У0413 | Наличие нужных объяснений | | » | 1 |
|  |  | | Итого: | 1 |

Продолжение таблицы Г.4.

|  |  |  |  |
| --- | --- | --- | --- |
| Понятность пользовательской документации | | | |
| У0501 | Оценка стиля изложения | » | 1 |
| У0502 | Дидактическая разделенность | » | 1 |
| У0503 | Формальная разделенность | » | 1 |
| У0504 | Ясность логической структуры | » | 1 |
| У0505 | Соблюдение стандартов и правил изложения в документации | » | 0,9 |
| У0506 | Оценка по числу ссылок вперед в тексте документов | » | 0 |
|  |  | Итого: | 0,8 |
| Техническое исполнение пользовательской документации | | | |
| У0601 | Наличие оглавления | » | 1 |
| У0602 | Наличие предметного указателя | » | 0 |
| У0603 | Наличие перекрестных ссылок | » | 0 |
| У0604 | Наличие всех требуемых разделов | » | 1 |
| У0605 | Соблюдение непрерывности нумерации страниц документов | » | 1 |
| У0606 | Отсутствие незаконченных разделов абзацев, предложений | » | 1 |
| У0607 | Наличие всех рисунков, чертежей, формул, таблиц | » | 1 |
| У0608 | Наличие всех строк и примечаний | » | 1 |
| У0609 | Логический порядок частей внутри главы | » | 1 |
|  |  | Итого: | 0,8 |
| Прослеживание вариантов пользовательской документации | | | |
| У0701 | Наличие полного перечня документации | » | 1 |
|  |  | Итого: | 1 |
| Эксплуатация | | | |
| У0801 | Уровень языка общения пользователя с программой | » | 1 |
| У0802 | Легкость и быстрота загрузки и запуска программы | » | 1 |
| У0803 | Легкость и быстрота завершения работы программы | » | 1 |
| У0804 | Возможность распечатки содержимого программы | » | 0 |
| У0805 | Возможность приостановки и повторного запуска работы без потерь информации | » | 0 |
|  |  | Итого: | 0,6 |

Продолжение таблицы Г.4.

|  |  |  |  |
| --- | --- | --- | --- |
|  | | Итого: | 0,6 |
| Управление меню | | | |
| У0901 | Соответствие меню требованиям пользователя | » | 1 |
| У0902 | Возможность прямого перехода вверх и вниз по многоуровневому меню (пропуск уровней) | » | 1 |
|  | | Итого: | 1 |
| Функция Help | | | |
| У1001 | Возможность управления подробностью получаемых выходных данных | » | 0 |
| У1002 | Достаточность полученной информации для продолжения работы | » | 1 |
|  | | Итого: | 0,5 |
| Управление данными | | | |
| У1101 | Обеспечение удобства ввода данных | » | 1 |
| У1102 | Легкость восприятия | » | 1 |
|  | | Итого: | 1 |
| Рабочие процедуры | | | |
| У1201 | Обеспечение программой выполнения предусмотренных рабочих процедур | » | 1 |
| У1202 | Достаточность информации, выдаваемой программой для составления доп. процедур | » | 1 |
|  | | Итого: | 1 |

**1.5 Показатели эффективности**

Таблица Г.5 – Оценочные элементы фактора "Эффективность ПС"

|  |  |  |  |
| --- | --- | --- | --- |
| Код элемента | Наименование | Метод опенки | Оценка |
| Уровень автоматизации | | | |
| Э0101 | Проблемно-ориентированные функции | » | 1 |
| Э0102 | Машинно-ориентированные функции | » | 0 |
| Э0103 | Функции ведения и управления | » | 1 |
|  |  | Итого: | 0,7 |

Продолжение таблица Г.5

|  |  |  |  |
| --- | --- | --- | --- |
| Э0104 | Функции ввода/вывода | » | 1 |
| Э0105 | Функции защиты и проверки данных | » | 1 |
| Э0106 | Функции защиты от несанкционированного доступа | » | 1 |
| Э0107 | Функции контроля доступа | » | 1 |
| Э0108 | Функции защиты от внесения изменений | » | 1 |
| Э0109 | Наличие соответствующих границ функциональных областей | » | 1 |
| Э0110 | Число знаков после запятой в результатах вычислений | » | 0 |
|  | | Итого: | 0,8 |
| Временная эффективность | | | |
| Э0201 | Время выполнения программ | » | 1 |
| Э0202 | Время реакции и ответов | » | 1 |
| Э0203 | Время подготовки | » | 0,5 |
| Э0205 | Затраты времени на защиту дан­ных | » | 1 |
| Э0206 | Время компиляции | » | 1 |
|  | | Итого: | 0,9 |
| Ресурсоемкость | | | |
| Э0301 | Требуемый объем внутренней па­мяти | » | 1 |
| Э0302 | Требуемый объем внешней памяти | » | 1 |
| Э0303 | Требуемые периферийные устрой­ства | » | 1 |
| Э0304 | Требуемое базовое программное обеспечение | » | 1 |
|  | | Итого: | 1 |

**1.6 Показатели универсальности**

Таблица Г.6 – Оценочные элементы фактора "Универсально ПС"

|  |  |  |  |
| --- | --- | --- | --- |
| Код элемента | Наименование | Метод оценки | Оценка |
| Широта охвата функций | | | |
| Г0101 | Оценка числа потенциальных пользователей | Экспертный | 1 |
| Г0102 | Оценка числа функций ПС | То же | 1 |
| Г0103 | Насколько набор функций удовлетворяет требованиям пользователя | » | 1 |

Продолжение таблицы Г.6.

|  |  |  |  |
| --- | --- | --- | --- |
| Г0104 | Насколько возможности программ охватывают область решаемых пользователем задач | » | 1 |
| Г0105 | Возможность настройки формата выходных данных для конкретных пользователей | » | 0 |
|  |  | Итого: | 0,8 |
| Простота архитектуры проекта | | | |
| Г0201 | Наличие схемы иерархии модулей программы | » | 1 |
| Г0202 | Оценка независимости модулей | » | 1 |
| Г0203 | Оценка числа уникальных элементов/реквизитов | » | 1 |
| Г0204 | Используется ли в текущем вызове модуля информация, полученная в предыдущем вызове | » | 0 |
| Г0205 | Оценка организации точек входа и выхода модуля | » | 1 |
| Г0206 | Наличие описания атрибутов модуля | » | 1 |
|  |  | Итого: | 0,8 |
| Сложность архитектуры проекта | | | |
| Г0301 | Оценка программ по числу переходов и точек ветвления | » | 1 |
|  |  | Итого: | 1 |
| Сложность структуры кода программы | | | |
| Г0401 | Использование метода пошагового уточнения | » | 0 |
| Г0402 | Наличие описания структуры программ | » | 1 |
| Г0403 | Наличие описания связей между элементами структуры программы | » | 1 |
|  | | Итого: | 0,7 |
| Качество комментариев | | | |
| Г1201 | Наличие заголовка в программе | » | 1 |
| Г1202 | Комментарии к точкам ветвлений | » | 0 |
| Г1205 | Комментарии к операторам объяв­ления переменных | » | 0 |
| Г1206 | Оценка семантики операторов | » | 1 |
| Г1207 | Наличие соглашений по форме представления комментариев | » | 0 |
| Г1208 | Наличие общих комментариев к программам | » | 0 |
|  | | Итого: | 0,4 |
| Использование описательных средств языка | | | |
| Г1301 | Использование языков высокого уровня | » | 0 |
| Г1302 | Семантика имен используемых пе­ременных | » | 1 |
| Г1303 | Использование отступов, сдвигов и пропусков при формировании текс­та | » | 1 |
| Г1304 | Размещение операторов по стро­кам | » | 1 |
|  | | Итого: | 0,8 |
| Независимость модулей | | | |
| Г1401 | Передача информации для управ­ления по параметрам | » | 0 |
| Г1402 | Наличие передачи результатов ра­боты между модулями | » | 0 |
| Г1403 | Наличие проверки правильности данных, получаемых модулями от вызываемого модуля | » | 1 |
| Г1404 | Наличие проверки правильности данных, получаемых модулями от вызываемого модуля | » | 0 |
| Г1405 | Параметрическая передача вход­ных данных | » | 1 |
|  | | **Итого:** | **0,4** |

**1.7 Показатели корректности**

Таблица Г.7 – Оценочные элементы фактора "Корректность ПС"

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Код элемента | | Наименование | Метод оценки | Оценка |
| Требования, предъявляемые к полноте документации разработчика | | | | |
| К0101 | | Наличие всех необходимых документов для понимания и использования ПС | Экспертный | 1 |
| К0102 | | Наличие описания и схемы иерар­хии модулей программы | » | 0 |
| К0103 | | Наличие описания основных функций | » | 1 |
| К0104 | | Наличие описания частных функций | » | 1 |
| К0105 | | Наличие описания данных | » | 0,5 |
| К0106 | | Наличие описания алгоритмов | » | 0,5 |
| К0107 | | Наличие описания интерфейсов между модулями | » | 0 |
| К0108 | | Наличие описания интерфейсов с пользователями | » | 1 |
| К0109 | | Наличие описания используемых числовых методов | » | 1 |
| К0110 | | Указаны ли все численные методы | » | 1 |
| К0111 | | Наличие описания всех параметров | » | 0,5 |
| К0112 | | Наличие описания методов наст­ройки системы | » | 1 |
| К0113 | | Наличие описания всех диагностических сообщений | » | 0 |
| К0114 | | Наличие описания способов проверки работоспособности программы | » | 0 |
|  | | | Итого: | 0,6 |
| Полнота программной документации | | | | |
| К0201 | | Реализация всех исходных модулей | » | 1 |
| К0202 | | Реализация всех основных функций | » | 1 |
| К0203 | | Реализация всех частных функций | » | 0 |
| К0204 | | Реализация всех алгоритмов | » | 0 |
| К0205 | | Реализация всех взаимосвязей в системе | » | 1 |
| К0206 | | Реализация всех интерфейсов меж­ду модулями | » | 0 |
| К0207 | | Реализация возможности настрой­ки системы | » | 1 |
| К0208 | | Реализация диагностики всех гра­ничных и аварийных ситуаций | » | 1 |
| К0209 | | Наличие определения всех данных (переменные, индексы, массивы и проч.) | » | 1 |
| К0210 | | Наличие интерфейсов с пользо­вателем | » | 1 |
|  | | | Итого: | 0,7 |
| Непротиворечивость документации разработчика | | | | |
| К0301 | Отсутствие противоречий в описа­нии частных функций | | » | 1 |
| К0302 | Отсутствие противоречий в описа­нии основных функций в разных до­кументах | | » | 1 |
| К0303 | Отсутствие противоречий в описа­нии алгоритмов | | » | 1 |
| К0304 | Отсутствие противоречий в описа­нии взаимосвязей в системе | | » | 1 |
| К0305 | Отсутствие противоречий в описании интерфейсов между модулями | | » | 1 |
| К0306 | Отсутствие противоречий в описании интерфейсов с пользователем | | » | 1 |
| К0307 | Отсутствие противоречий в описании настройки системы | | » | 1 |
| К0309 | Отсутствие противоречий в описании иерархической структуры сообщений | | » | 1 |

Продолжение таблицы Г.7

|  |  |  |  |
| --- | --- | --- | --- |
| К0310 | Отсутствие противоречий в описании диагностических сообщений | » | 1 |
| К0311 | Отсутствие противоречий в описа­нии данных | » | 1 |
|  | | Итого: | 1 |
| Непротиворечивость программы | | | |
| К0401 | Отсутствие противоречий в выполнении основных функций | » | 1 |
| К0402 | Отсутствие противоречий в выпол­нении частных функций | » | 1 |
| К0403 | Отсутствие противоречий в выпол­нении алгоритмов | » | 1 |
| К0404 | Правильность взаимосвязей | » | 1 |
| К0406 | Правильность реализации интер­фейса с пользователем | » | 1 |
| К0407 | Отсутствие противоречий в настройке системы | » | 1 |
| К0408 | Отсутствие противоречий в диаг­ностике системы | » | 1 |
| К0409 | Отсутствие противоречий в общих переменных | » | 1 |
|  |  | Итого: | 1 |
| Единообразие интерфейсов между модулями и пользователями | | | |
| К0501 | Единообразие способов вызова мо­дулей | » | 1 |
| К0502 | Единообразие процедур возврата управления из модулей | » | 1 |
| К0503 | Единообразие способов сохранения информации для возврата | » | 1 |
| К0504 | Единообразие способов восстанов­ления информации для возврата | » | 1 |
| К0505 | Единообразие организации списков передаваемых параметров | » | 1 |
|  | | Итого: | 1 |
| Единообразие кодирования и определения переменных | | | |
| К0601 | Единообразие наименования каж­дой переменной и константы | » | 1 |
| К0602 | Все ли одинаковые константы встречаются во всех программах под одинаковыми именами | » | 1 |
| К0603 | Единообразие определения внеш­них данных во всех программах | » | 1 |

Продолжение таблицы Г.7

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| К0604 | Используются ли разные идентифи­каторы для разных переменных | | » | 1 |
| К0605 | Все ли общие переменные объяв­лены как общие переменные | | » | 1 |
| К0606 | Наличие определений одинаковых атрибутов | | » | 1 |
|  | | | Итого: | 1 |
| Соответствие документации стандартам | | | | |
| К0701 | | Комплектность документации в со­ответствии со стандартами | » | 1 |
| К0702 | | Правильное оформление частей документов | » | 1 |
| К0703 | | Правильное оформление титульных и заглавных листов документов | » | 1 |
| К0704 | | Наличие в документах всех разде­лов в соответствии со стандартами | » | 1 |
| К0705 | | Полнота содержания разделов в соответствии со стандартами | » | 1 |
| К0706 | | Деление документов на структур­ные элементы: разделы, подразделы, пункты, подпункты | » | 1 |
|  | | | Итого: | 1 |
| Соответствие ПС стандартам программирования | | | | |
| К0801 | | Соответствие организации и вы­числительного процесса эксплуата­ционной документации | » | 1 |
| К0802 | | Правильность заданий на выпол­нение программы, правильность на­писания управляющих и операторов (отсутствие ошибок) | » | 1 |
| К0803 | | Отсутствие ошибок в описании дей­ствий пользователя | » | 1 |
| К0804 | | Отсутствие ошибок в описании за­пуска | » | 1 |
| К0805 | | Отсутствие ошибок в описании ге­нерации | » | 1 |
| К0806 | | Отсутствие ошибок в описании на­стройки | » | 1 |
|  | | | Итого: | 1 |

Продолжение таблицы Г.7

|  |  |  |  |
| --- | --- | --- | --- |
| Полнота тестирования проекта | | | |
| К1001 | Наличие требований к тестированию программ | » | 1 |
| К1002 | Достаточность требований к тестирова­нию программ | » | 1 |
| К1003 | Отношение числа модулей, отработавших в процессе тестирования и отладки к общему числу модулей | Расчетный | 0,8 |
| К1004 | Отношение числа логических блоков, отработавших в процессе тестирования и отладки (Qтб), к общему числу логических блоков в программе | То же | 1 |
|  | | Итого: | 0,95 |

**1.8 Результаты оценки качества программного продукта**

**1.8.1 Итоговые оценки метрик**

Итоговая оценка k-ой метрики j-го критерия ведется по формуле (Г.1).

(Г.1)



где Q – число оценочных элементов в k-ой метрике;

m – оценочный элемент k-ой метрики.

Таблица Г. 8 – Метрики

|  |  |  |
| --- | --- | --- |
| Код | Метрика | Оценка |
| Н01 | Средства восстановления при ошибках на входе | 0,9 |
| Н02 | Средства восстановления при сбоях оборудования | 0 |
| Н03 | Реализация управления средствами восстановления | 0,4 |
| Н04 | Функционирование в заданных режимах | 0,9 |
| Н05 | Обеспечение обработки заданного объема информации | 1 |
| С01 | Простота архитектуры проекта | 1 |
| С02 | Сложность архитектуры проекта | 0 |
| С03 | Межмодульные связи | 0,7 |
| С06 | Экспертиза принятой системы идентификации | 1 |
| С08 | Комментарии логики программ проекта | 0 |
| С09 | Оформление текста программ | 0,5 |
| С10 | Простота кодирования | 0,7 |
| У01 | Освоение работы ПС | 1 |
| У02 | Документация для освоения | 1 |
| У03 | Полнота пользовательской документации | 0,8 |
| У04 | Точность пользовательской документации | 1 |

Продолжение таблицы Г. 8

|  |  |  |
| --- | --- | --- |
| У05 | Понятность пользовательской документации | 0,8 |
| У06 | Техническое исполнение пользовательской документации | 0,8 |
| У07 | Прослеживание вариантов пользовательской документации | 0,6 |
| У08 | Эксплуатация | 0,6 |
| У09 | Управление меню | 1 |
| У10 | Функция Help | 0,5 |
| У11 | Управление данными | 1 |
| У12 | Рабочие процедуры | 1 |
| Э01 | Уровень автоматизации | 0,7 |
| Э02 | Временная эффективность | 0,9 |
| Э03 | Ресурсоемкость | 1 |
| Г01 | Широта охвата функций | 1 |
| Г02 | Простота архитектуры проекта | 1 |
| Г03 | Сложность архитектуры проекта | 0 |
| Г04 | Сложность структуры кода программы | 0,7 |
| Г05 | Применение стандартных протоколов связи | 1 |
| Г06 | Применение стандартных интерфейсных программ | 0 |
| Г07 | Зависимость от используемого комплекса технических средств | 0 |
| Г08 | Зависимость от базового программного обеспечения | 1 |
| Г09 | Изоляция немобильности | 1 |
| Г10 | Простота кодирования | 0,7 |
| Г12 | Качество комментариев | 0,4 |
| Г13 | Использование описательных средств языка | 0,8 |
| Г14 | Независимость модулей | 0,4 |
| К01 | Требования, предъявляемые к полноте документации разработчика | 0,6 |
| К02 | Полнота программной документации | 0,7 |
| К03 | Непротиворечивость документации разработчика | 1 |
| К04 | Непротиворечивость программы | 1 |
| К05 | Единообразие интерфейсов между модулями и пользователями | 1 |
| К06 | Единообразие кодирования и определения переменных | 1 |
| К07 | Соответствие документации стандартам | 1 |
| К08 | Соответствие ПС стандартам программирования | 1 |
| К10 | Полнота тестирования проекта | 0,95 |

**1.8.2 Абсолютные показатели критериев**

Абсолютные показатели критериев i-го фактора качества определяется по формуле (Г.2).

, (Г.2)



где n – количество метрик, относящихся к j-ому критерию;

– весовой коэффициент k-ой метрики j-го критерия.



Таблица Г. 9 – Показатели критерия

|  |  |  |
| --- | --- | --- |
| Код | Абсолютный показатель критерия | Оценка |
| Н | Устойчивость функционирования | P11 = (0.9 + 0 + 0.4) / 3 = 0.43 |
| Работоспособность | Р12= (1 + 0.9) / 2 = 0.95 |
| C | Простота конструкции | Р22= (1 + 0 + 0.7) / 3 = 0.56 |
| Наглядность | Р23= (1 + 0 + 0.5) / 3 = 0.5 |
| Структурность | Р24= 0.7 |
| У | Легкость освоения | Р31= (1 + 1) / 2 = 1 |
| Доступность эксплуатационных программных документов | Р32= (0.7 + 1 + 0.8 + 0.7 + 0.7) / 5 = 0.8 |
| Удобство эксплуатации и обслуживания | Р33= (0.6 + 1 + 0,5 + 1 + 1 + 0.8) / 5 = 0.98 |
| Э | Уровень автоматизации | P41=0.7 |
| Временная эффективность | P42=0.9 |
| Ресурсоемкость | P43=1 |
| Гибкость | Р51= (1 +1 + 0 + 0.7 + 1 + 0) / 6 = 0.61 |
| Мобильность | Р52= (0 + 1 + 1) / 3 = 0.67 |
| Модифицируемость | Р53= (0.7 + 0.7 + 0.4 + 0.8 + 1) / 5 = 0.72 |
| Полнота реализации | Р61= (0,7 + 0,6) / 2 = 0,65 |
| Согласованность | Р62= (1 + 1 + 1 + 1 +1) / 5 = 1 |
| Проверенность | Р64= (1 + 0,95) / 2 = 0,98 |

**1.8.3 Относительные показатели критериев**

Относительные показатели критериев качества равны абсолютным показателям, т.к. в качестве базовых значений мы принимаем единицу.

**1.8.4 Факторы качества**

Фактор качества вычисляется по формуле (Г.3).

(Г.3)



Таблица Г. 10 – Факторы качества

|  |  |
| --- | --- |
| Показатель | Значение |
| Показатель надежности ПС | = (0.43 + 0.95) / 2 = 0,69 |
| Показатель сопровождения | = (0.56 + 0.5 + 0.7) / 3 = 0.59 |
| Показатель удобства применения | = (1 + 0.98 + 0.8) / 3 = 0.93 |
| Показатель эффективности | = (0.7 + 0.9 + 1) / 3 = 0.87 |
| Показатель универсальности | = (0.72 + 0.61 + 0.67) / 3 = 0.67 |
| Показатель корректности | = (0.65 + 0.98 + 1) / 3 = 0.88 |