**БрГТУ**

**Технический дизайн для приложения “Requests Scheduler”**

Дата: 13ое Августа 2022

Автор: Иван Иваненко

Версия: 0.2

# Контроль Документа

**Последний раз сохранен:** 06/08/2022 18:03

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Changes | Author | Date |
| 0.1 | Создана первая версия документа | Иван Иваненко | Август, 6, 2022 |
| 0.2 | Обновлен дизайн Requests Interactor | Иван Иваненко | Август, 13, 2022 |

Содержание

[Контроль Документа 2](#_Toc111305688)

[Введение 4](#_Toc111305689)

[Дизайн на уровне компонентов 5](#_Toc111305690)

[Механизм генерации заявок 6](#_Toc111305691)

[Структура конфигурации интенсивности заявок 7](#_Toc111305692)

[Дизайн Requests Interactor 9](#_Toc111305693)

# Введение

Requests Scheduler – кроссплатформенное .NET Core приложение, развретываемое в виде Unix daemon на Unix подобных системах, либо в виде Windows Service (.svc) на системах платформы windows. Данный тип сервиса запускается как фоновый процесс на машине-хосте и не требует никакого взаимодействия с пользователем за исключением редактирования файла конфигурации. В таком случае после редактирования файла требуется перезапустить сервис.

# Дизайн на уровне компонентов

Diagram

Description automatically generated

1. Requests Interactor – библиотека классов, содержащая основную логику сервиса. Независимый и устойчивый компонент. Содержит в себе интерфейс для доступа к конфигурации и интерфейс для доступа к оснвоному функционалу модуля, а также классы, реализующие эту логику.
2. Requests Configuration – библиотека классов для чтения конфигурации и предоставляющие данные конфигурации в формате, удобном для Requests Interactor. Реализует интерфейс для доступа к конфигурации, объявленный в Requests Interactor.
3. Requests Service – Модуль для запуска сервиса и реализации механизма внедрения зависмостей. Самый неустойчивый модуль. Обращается к осному интерфейсу Requests Interactor для доступа к функционалу этого модуля. В Requests Service нет никакой логики за исключением реализации Background Service и механизма внедрения зависимостей.

# Механизм генерации заявок

1. Если заявки для текущего временного промежутка сгенерированы, сервис их обрататывает.
2. Иначе:
   1. Requests Service вызывает метод GenerateRequestsForCurrentPeriod интерфейса IRequestsGenerator модуля Requests Interactor.
   2. RequestsGenerator обращается к своему методам своего инетфейса для чтения конфигурации.
   3. Requests Configuration осуществляет логику чтения конфиуграции.
   4. RequestsGenerator генерирует заявки на основе полученной конфигурации.
3. Requests Service получает объекты заявок, сохраняет для текущего промежутка и продолжает обработку.

# Структура конфигурации интенсивности заявок

{

"$schema": "",

"type": "array",

"items": [

{

"type": "object",

"properties": {

"routeNumber": {

"type": "integer"

},

"dailyRequestsCount": {

"type": "integer"

},

"timePeriods": {

"type": "array",

"items": [

{

"type": "object",

"properties": {

"from": {

"type": "string"

},

"to": {

"type": "string"

},

"requestsCountCoefficient": {

"type": "float"

}

},

"required": [

"from",

"to",

"requestsCountCoefficient"

]

}

]

}

},

"required": [

"routeNumber",

"dailyRequestsCount",

"timePeriods"

]

}

]

}

# Дизайн Requests Interactor

**Diagram

Description automatically generated**

Я не буду рассматривать дизайн классов для остальных компонентов.