# **Документация к проекту "Топливозаправщики"**

## **Введение**

Проект "Топливозаправщики" представляет собой распределенную систему, предназначенную для управления процессом заправки самолетов на аэродроме. Основная цель проекта — автоматизация и оптимизация работы топливозаправщиков, которые перемещаются между объектами аэродрома (гараж, заправка, самолет) для выполнения заказов на заправку. Каждый участник команды отвечает за разработку своего модуля, который взаимодействует с другими компонентами системы. В данной документации будет подробно описан модуль топливозаправщиков, его функциональность, взаимодействие с другими модулями, а также мой вклад в разработку.

## **Мой вклад в проект**

Моя задача заключалась в разработке модуля топливозаправщиков, который отвечает за управление их созданием, движением и выполнением заказов на заправку. В рамках этой задачи я реализовал следующие функции:

1. **Логика работы топливозаправщиков**:  
   Топливозаправщики перемещаются между тремя основными объектами: гаражом, заправкой и самолетом. Каждый заправщик имеет свои параметры, такие как текущее местоположение, статус (занят/свободен), объем топлива в баке и целевой объект, к которому он движется. Логика работы заправщиков реализована в классе FuelTruck, где каждый экземпляр класса представляет собой отдельный топливозаправщик.
2. **Взаимодействие с другими модулями**:  
   Модуль топливозаправщиков взаимодействует с несколькими другими компонентами системы:
   * **Диспетчер руления (DR)**: Отвечает за управление движением заправщиков по аэродрому. Модуль запрашивает у DR маршруты для перемещения между объектами и получает разрешение на переход между контрольными точками.
   * **Управление наземными операциями (УНО)**: После завершения заправки модуль отправляет уведомление в УНО о выполнении заказа.
   * **Сервер самолета**: После завершения заправки модуль отправляет уведомление самолету о том, что заправка успешно завершена.
3. **Пользовательский интерфейс**:  
   Для удобства управления системой был разработан веб-интерфейс, который позволяет настраивать параметры работы топливозаправщиков, такие как максимальное количество заправщиков, частота их перемещения и объем бака. Интерфейс также отображает текущее состояние системы, включая количество созданных заправщиков и выполненных заказов. Лог-панель в интерфейсе позволяет отслеживать действия системы в реальном времени.
4. **Обработка заказов**:  
   Модуль принимает заказы на заправку, которые включают в себя идентификатор самолета, объем топлива и номер заказа. В зависимости от текущей загруженности системы, модуль либо создает новый топливозаправщик, либо назначает задачу свободному заправщику.
5. **Автоматическое обновление интерфейса**:  
   Веб-интерфейс автоматически обновляется каждые 5 секунд, что позволяет пользователю видеть актуальное состояние системы без необходимости обновления страницы.

## **Описание модулей системы**

### **Управление наземными операциями (УНО)**

Управление наземными операциями (УНО) — это модуль, который координирует работу всех наземных объектов на аэродроме, включая топливозаправщики. Основная задача УНО — принимать отчеты о выполнении заказов и управлять взаимодействием между различными модулями системы. В контексте модуля топливозаправщиков, УНО получает уведомления о завершении заправки и обновляет статус заказа.

### **Диспетчер руления (DR)**

Диспетчер руления (DR) отвечает за управление движением всех транспортных средств на аэродроме, включая топливозаправщики. Модуль топливозаправщиков взаимодействует с DR для получения маршрутов перемещения между объектами (гараж, заправка, самолет). DR также контролирует доступность точек гаража, где могут создаваться новые топливозаправщики. Если точка гаража занята, модуль топливозаправщиков ожидает ее освобождения перед созданием нового заправщика.

### **Сервер самолета**

Сервер самолета — это модуль, который отвечает за взаимодействие с самолетами на аэродроме. После завершения заправки модуль топливозаправщиков отправляет уведомление серверу самолета о том, что заправка успешно завершена. Это позволяет самолету продолжить подготовку к вылету.

### **Модуль топливозаправщиков**

Модуль топливозаправщиков является центральным компонентом системы. Он отвечает за создание и управление топливозаправщиками, выполнение заказов на заправку и взаимодействие с другими модулями системы. Основные функции модуля включают:

* Создание новых топливозаправщиков при поступлении заказов.
* Управление движением заправщиков между объектами (гараж, заправка, самолет).
* Отправка уведомлений о завершении заправки в УНО и сервер самолета.
* Обработка ошибок и повторные попытки выполнения задач в случае сбоев.

## **Пользовательский интерфейс**

Пользовательский интерфейс модуля топливозаправщиков реализован в виде веб-страницы, которая позволяет управлять параметрами системы и отслеживать ее состояние. Интерфейс разделен на две основные части:

1. **Верхняя часть**:  
   В верхней части интерфейса отображаются параметры системы, такие как максимальное количество топливозаправщиков, частота их перемещения и объем бака. Пользователь может изменять эти параметры через форму и отправлять их на сервер для обновления.
2. **Нижняя часть**:  
   В нижней части интерфейса находится лог-панель, которая отображает информацию о текущих действиях системы. Лог автоматически обновляется каждые, что позволяет пользователю видеть актуальное состояние системы.

## **API модуля топливозаправщиков**

Модуль топливозаправщиков предоставляет несколько API-эндпоинтов для взаимодействия с другими модулями системы:

1. **Эндпоинт для приема заказов**:  
   /fueltruck/order/<int:order\_no>/<int:volume\_plane>/<int:plane\_id>  
   Этот эндпоинт используется для приема заказов на заправку. Он принимает номер заказа, объем топлива и идентификатор самолета. В зависимости от текущей загруженности системы, модуль либо создает новый топливозаправщик, либо назначает задачу свободному заправщику.
2. **Эндпоинт для обновления параметров системы**:  
   /gas  
   Этот эндпоинт используется для обновления параметров системы через POST-запрос. Пользователь может изменять такие параметры, как максимальное количество топливозаправщиков, частота их перемещения и объем бака. Эндпоинт также используется для отображения текущего состояния системы через GET-запрос.

### **Преимущества подхода с использованием URL для взаимодействия между модулями**

Текущий подход к взаимодействию между модулем топливозаправщиков и диспетчером руления, основанный на использовании URL, уже обладает рядом значительных преимуществ. Этот метод прост, эффективен и хорошо подходит для текущих задач системы. Ниже приведены основные причины, почему этот подход можно считать удачным решением.

#### **Простота и минимализм**

Одним из главных преимуществ текущего подхода является его простота. Взаимодействие между модулями осуществляется через стандартные HTTP-запросы, что делает код минималистичным и легко понятным. Например, запрос на разрешение перехода между контрольными точками выглядит как простой URL:

http://<ip\_dr>:<port>/dispatcher/point/<current\_checkpoint>/<next\_checkpoint>

Такой формат не требует сложных структур данных или дополнительных слоев абстракции, что упрощает разработку и поддержку кода.

#### **Использование стандартных HTTP-методов**

Текущий подход использует стандартные HTTP-методы, такие как GET, что делает его предсказуемым и соответствующим общепринятым практикам. Это упрощает интеграцию с другими системами, так как HTTP — это универсальный протокол, поддерживаемый практически всеми языками программирования и платформами. Такой подход знаком большинству разработчиков и не требует дополнительного обучения.

#### **Легкость отладки и тестирования**

Использование URL делает процесс отладки и тестирования максимально простым. Разработчики могут легко проверить работу системы, вставив URL в браузер или используя инструменты вроде Postman. Это позволяет быстро находить и исправлять ошибки, не углубляясь в сложные логические цепочки. Например, если что-то не работает, можно просто открыть URL в браузере:

http://localhost:5555/fueltruck/order/11/20000/1313

и сразу увидеть ответ сервера.

#### **Минимальные накладные расходы**

Текущий подход не требует сложной сериализации данных. Вместо этого используется простой текстовый ответ ("true" или "false"), что минимизирует накладные расходы на передачу данных. Это особенно важно в системах, где важна производительность и скорость обработки запросов. Это минималистично и эффективно, что позволяет системе работать быстро даже при большом количестве запросов.

#### **Гибкость в интеграции**

Текущий подход с использованием URL легко интегрируется с другими системами, даже если они написаны на разных языках программирования. Поскольку HTTP — это стандартный протокол, любой язык или платформа может отправить запрос на указанный URL и получить ответ. Это делает систему гибкой и готовой к интеграции с различными компонентами.

#### **Прозрачность логики взаимодействия**

Текущий подход делает логику взаимодействия между модулями прозрачной и легко понятной. Разработчики могут легко увидеть, какие данные передаются и какие ответы ожидаются. Например, код, обрабатывающий ответ от сервера, выглядит так:

if response.text == 'true':

return True

else:

return False

Этот код очевиден: если ответ "true", переход разрешен, иначе — запрещен. Такая прозрачность упрощает понимание кода для других разработчиков и снижает вероятность ошибок.

#### **Легкость расширения**

Текущий подход с URL легко расширяем. Если в будущем потребуется добавить новые параметры или функции, это можно сделать, просто изменив URL или формат ответа, не нарушая существующую логику. Например, можно добавить параметр priority в запрос:

http://<ip\_dr>:<port>/dispatcher/point/<current\_checkpoint>/<next\_checkpoint>?priority=high

Это позволяет системе адаптироваться к новым требованиям без значительных изменений в коде.

#### **Минимальная зависимость от внешних библиотек**

Текущий подход не требует сложных библиотек для работы с API. Достаточно использовать стандартные средства языка, такие как библиотека requests в Python. Это делает код легковесным и независимым от дополнительных зависимостей. Например, для работы с API достаточно импортировать одну библиотеку:

import requests

## 

## **Работоспособность модуля**

Модуль топливозаправщиков был протестирован на корректность выполнения заказов, взаимодействие с другими модулями и обработку ошибок. В процессе тестирования были реализованы заглушки для имитации работы других модулей, что позволило проверить функциональность модуля в изолированной среде. Все основные сценарии работы системы были успешно протестированы, включая создание новых топливозаправщиков, выполнение заказов на заправку и обработку ошибок при взаимодействии с другими модулями.

## **Заключение**

Модуль топливозаправщиков успешно интегрирован в распределенную систему и выполняет все необходимые функции для управления заправщиками и выполнения заказов на заправку. В процессе разработки были реализованы пользовательский интерфейс, API для взаимодействия с другими модулями и логика работы топливозаправщиков. Документация и API описаны в соответствии с требованиями проекта, что позволяет другим разработчикам легко интегрировать модуль в систему.