**Rust multi module microservices Part 5 — Common**

[[](https://medium.com/@omprakashsridharan?source=post_page-----5eb74ea0e6ee--------------------------------)](https://medium.com/@omprakashsridharan?source=post_page-----5eb74ea0e6ee--------------------------------)

[Omprakash Sridharan](https://medium.com/@omprakashsridharan?source=post_page-----5eb74ea0e6ee--------------------------------)

·

Follow

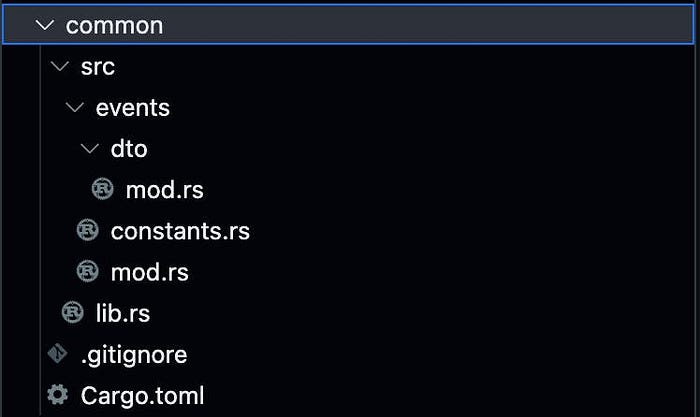
2 min read

·

Jun 27

1

Every workspace can have multiple application crates that can have some common utility functions, constants, models/structs for HTTP Request/Response, Kafka Messages, and much more. For this purpose, we are having the common crate to be used for holding common business domain contexts.



Go ahead and create the above files/folders and we will update the contents as shown below.

**Cargo.toml**

[package]  
name = "common"  
version = "0.0.0"  
edition = "2021"  
  
# See more keys and their definitions at https://doc.rust-lang.org/cargo/reference/manifest.html  
  
[dependencies]  
serde = {workspace = true}  
derive\_builder = { workspace = true}  
strum = {workspace = true}  
apache-avro = {workspace = true}

We add the [derive\_builder](https://docs.rs/derive_builder/latest/derive_builder/) and [strum](https://docs.rs/strum/latest/strum/) crates for utilities. You can read more about them to check out their capabilities.

**src/events/dto/**[**mod.rs**](http://mod.rs/)

use apache\_avro::AvroSchema;  
use derive\_builder::Builder;  
use serde::{Deserialize, Serialize};  
  
#[derive(Serialize, Deserialize, Builder, Clone, Debug, AvroSchema)]  
pub struct CreatedBook {  
 id: i32,  
 title: String,  
 isbn: String,  
}

The CreatedBook struct serves as the structure for the Kafka message that we will produce in books\_api and consume in books\_analytics. Although we use Rust for both producing and consuming, the need for a schema registry to validate the schema might not be as crucial. However, in real-world scenarios, it is highly likely that other microservices will be written in various language-framework combinations, which justifies the need for a schema registry. This registry can prevent messages with non-conforming schemas from being produced or consumed.

**src/events/**[**constants.rs**](http://constants.rs/)

use strum::Display;  
  
#[derive(Display)]  
pub enum Topics {  
 BookCreated,  
}

We have created an enumeration of Topics, but feel free to change it to a String or any other data structure that better suits your needs.

**src/events/**[**mod.rs**](http://mod.rs/)

pub mod constants;  
pub mod dto;

We register the modules of the events and expose them publically.

**src/**[**lib.rs**](http://lib.rs/)

pub mod events;

At this stage, we have completed the implementation of the common modules and are ready to create the two microservices, books\_api and books\_analytics, in the upcoming articles.