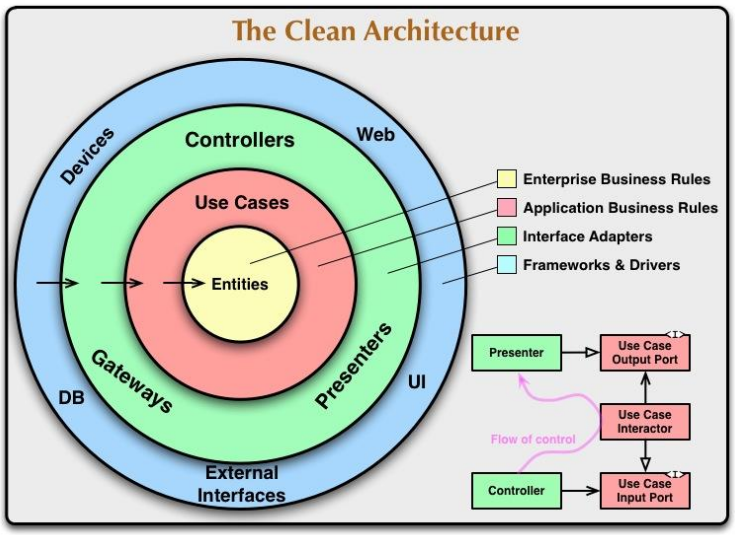
Objective

To make the model developed to be classifying products in categories in real time, it is necessary to create an API to productize this model and make it accessible at any time by sending some data.

Clean Architecture



For developing this software it was decided to use the “Clean Architecture” because this model was created with focus on business rulers and the “Clean Architecture” model this software having the business rulers on the core. This makes this kind of architecture better for models’ deployment. Below are all the responsibilities of every layer.

Entities (Inside Domain)

The core layer, every layer depends on it and holds the most important business rulers

Responsibilities:

- **Product**: guarantee the data reliability on the dataset for the application.

Use Cases (Inside Domain)

This layer holds some business rulers who are less important than the rulers in entities and the use cases of the software that the users are going to need.

Responsibilities:

- **Feature Derivator**: derivate all features who are going to be needed for the model.
- **Rescaler**: rescale all features to the correct scale for the model.

Connectors (Inside Domain)

This layer has the responsibility to connect the business rulers from inner layers with the external factors.

Responsibilities:

- **Data Preparator**: transform all data for the correct format for external factors.
- **Model Runner**: run the model and transform the data for external factors.

External Factors (Outside Domain)

This layer has the responsibility to hold all external factors who are not in the domain and can need to be changed are very flexible to be changed.

Responsibilities:

- **Framework Flask (main.py)**: create an API to expose the application in an endpoint.
- **Tests**: unitary and manual tests for debugging the software.
- **Docker**: containerization of the application to guarantee the accuracy.
- **Model**: contain the model who is running and doing the predictions.
- **Scalers**: contain the scalers for rescaling the data.

Google Cloud

This application was deployed on Google Cloud using two services from it.

Container Registry

This service was selected to keep the Docker Image of this project which is being used to guarantee the accuracy of this application in relation with the performance on localhost.

Cloud Run

This service runs the image from the container registry to a serverless application, creating the endpoint of this application and possibilities for this application to be requested.

CI / CD

All the CI / CD process was done with the github actions, authenticating with a Service Account Key, **creating the docker image who runs all the tests (CI)** and **sending this docker image for the cloud run (CD)**.