Requirements Analysis

Abstract

The application that we want to build is basically a repository of web services. It provides the option of automatically registering web services, given it's source code repository. This mechanism is based on the idea of web registry (see diagram 1).

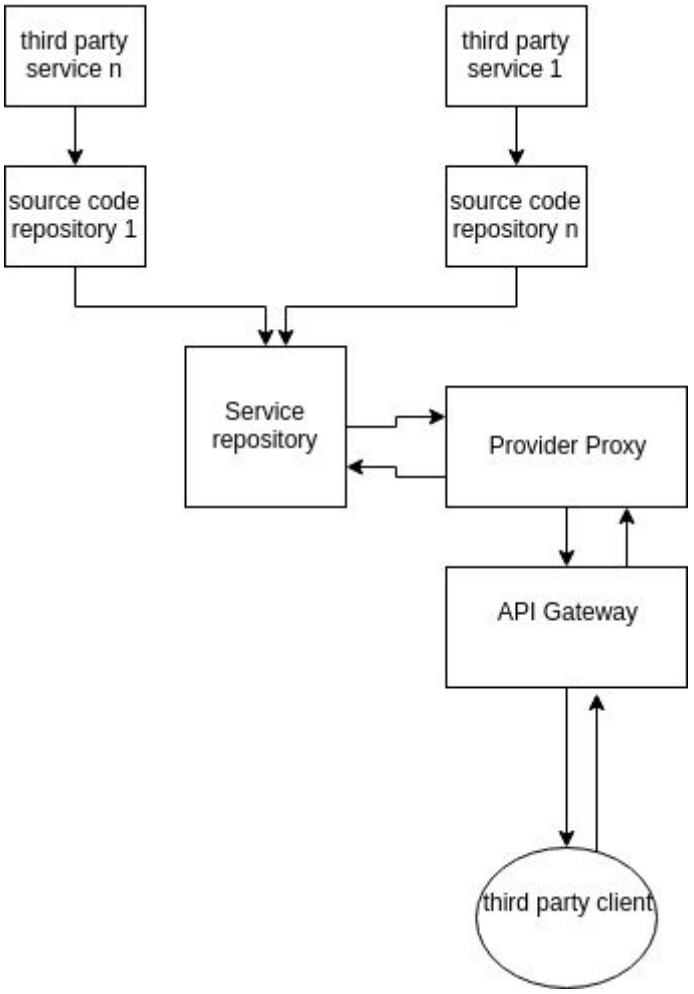

Diagram 1.

On top of this abstract layer we want to add a real world use. We will build a platform that will provide through the means of a web interface different information about the Yassi city. Basically we will build a series of web services for the repository described above and a web interface that will act as client for this collection of services. All these components together will act as the app <<SmartCity>> (See diagram 2).

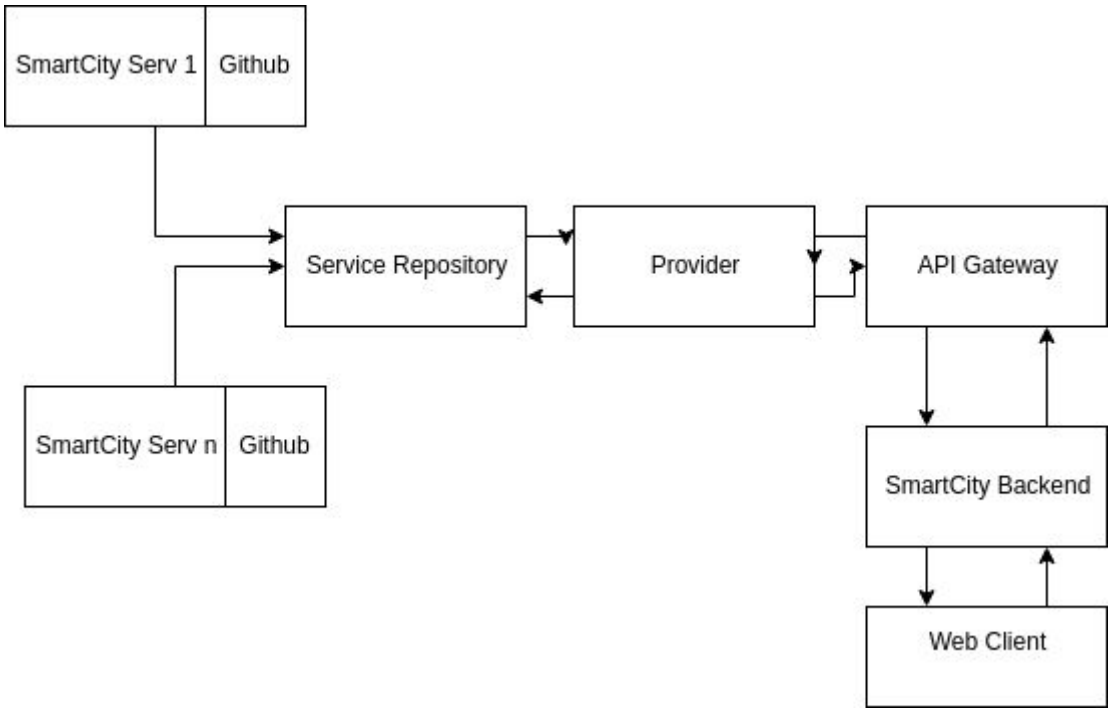

Diagram 2.

Main components

From a high level perspective. The main software components and its critical sub-components are:

1. *Service repository*
    a. *source repository scrappers*
        i. *adapters for both github and bitbucket*
    b. *adapter/runners/environment providers*
        i. *environment adapters -- software sub-components that prepare a programmatic virtual environment in which each service will run*
        ii. *service runners -- software sub-components with the purpose of running each service in the repository*
        iii. *output providers -- collects the output generated by each service, this sub-component connects to the environment in which the service is running and extracts the output from the environment to an in-memory location present in the repository*
2. *Third party services*
    a. *service registry file (this has to be present on each service repository, based on the information in this file the core service repository will know how to build an environment and run each third party service)*

3. *Provider proxy -- this component queries an adapter over each service in repository and prepares the output for the Gateway. Also acts as a load balancer*
    a. *output collectors -- collects the output generated by each service*
    b. *output adapters -- adapts the output produces by each service (as described in the registry file) to a generic, standardize format specific to the project*
4. *API Gateway -- this component exposes a series of generic both RESTful SOAP APIs over the services collected in the repository*
    a. *REST endpoints*
    b. *SOAP endpoints*
5. *Web client -- basically, a bunch of JavaScript put together in an nice-looking web interface and API clients*

Technology stack

This project has a broad range of programming languages and tools that need to be put together for the project to work properly, since the project itself is heavy back-end based. Grouped by each component these are the software tools that we intend to use:

- Java (Spring, Guava, Apache Cassandra), Python, Docker, Bash for the service repository
- HTML, JavaScript, React, CSS -- for the web client

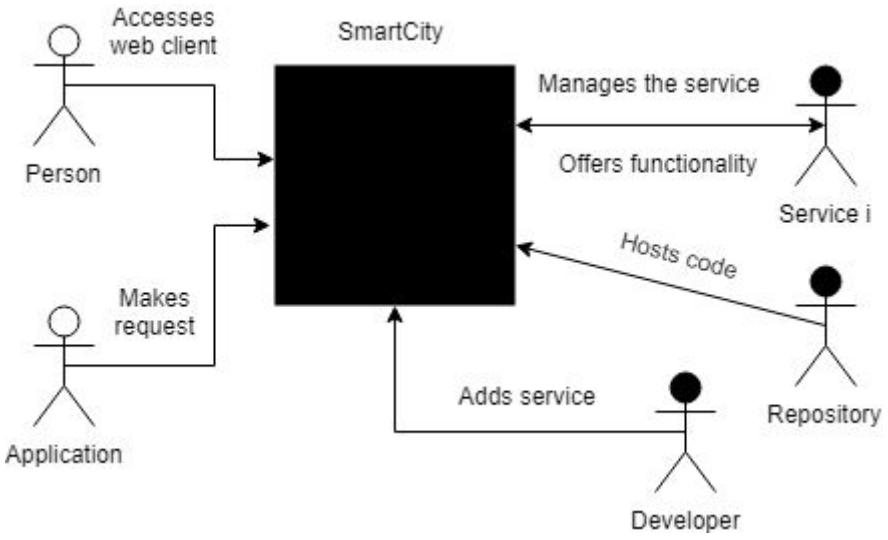

Diagram 3.

Actors

The primary actors are the users of the platform, which are either:

1. A person -- *through the use of the front-end web client (Main component 5.)*
2. or an application -- *through the REST / SOAP API that is exposed and acts as a facade for the collection of services.*

While the secondary actors are those that help the ends meet. The main entities in this category are :

1. Third party services -- *because the purpose of this platform is to pave the way from the user to the service.*
2. Developer -- *which is needed in order to add a new service*
3. Repository service -- *source repository will be hosted outside of our system, this implies a new actor is involved*