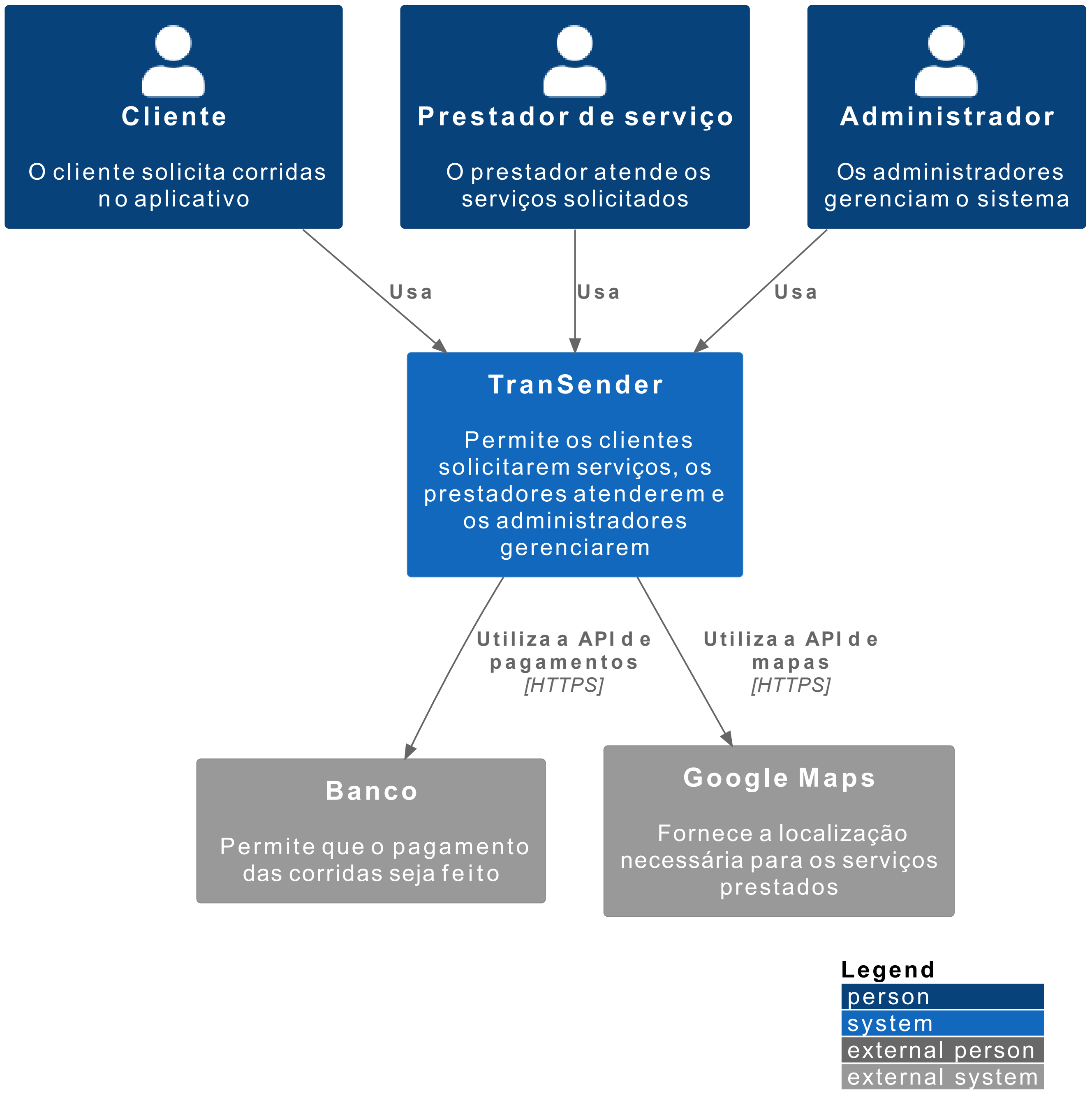
TranSender



Cliente
O cliente solicita corridas no aplicativo
Prestador de serviço
O prestador atende os serviços solicitados
Administrador
Os administradores gerenciam o sistema
Usa
Usa
Usa
TranSender
Permite os clientes solicitarem serviços, os prestadores atenderem e os administradores gerenciarem
Utiliza a API de pagamentos [HTTPS]
Utiliza a API de mapas [HTTPS]
Banco
Permite que o pagamento das corridas seja feito
Google Maps
Fornece a localização necessária para os serviços prestados
Legend
person
system
external person
external system

Level 1: System Context diagram

A System Context diagram is a good starting point for diagramming and documenting a software system, allowing you to step back and see the big picture. Draw a diagram showing your system as a box in the centre, surrounded by its users and the other systems that it interacts with.

Detail isn't important here as this is your zoomed out view showing a big picture of the system landscape. The focus should be on people (actors, roles, personas, etc) and software systems rather than technologies, protocols and other low-level details. It's the sort of diagram that you could show to non-technical people.

Scope: A single software system.

Primary elements: The software system in scope. Supporting elements: People (e.g. users, actors, roles, or personas) and software systems (external dependencies) that are directly connected to the software system in scope. Typically these other software systems sit outside the scope or boundary of your own software system, and you don't have responsibility or ownership of them.

Intended audience: Everybody, both technical and non-technical people, inside and outside of the software development team.

TranSender

Introduction

This project was created using c4builder

Take a look at

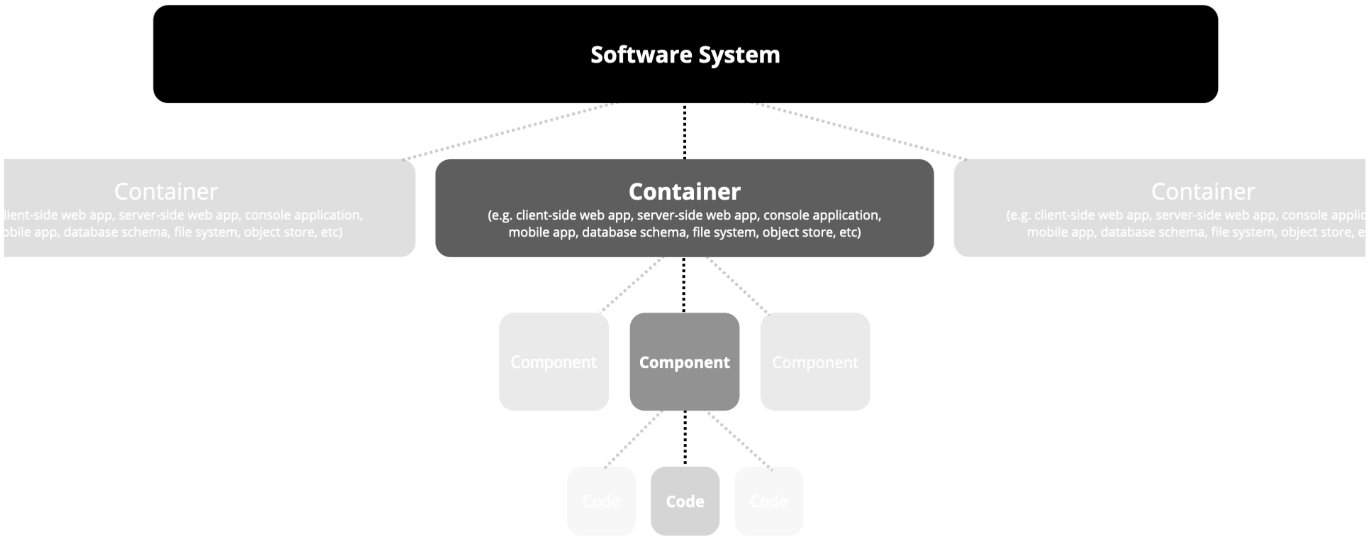
- PlantUml creates diagrams from plain text.
- Markdown creates rich text documents from plant text.
- C4Model the idea behind maps of your code
- C4-PlantUML C4 syntax support for generating plantuml diagrams
- vscode-plantuml plugin for visual studio code to view diagrams at design time

Open the terminal and run the following commands to start compiling the documentation

```
npm i -g c4builder
c4builder
```

> Note on using local images inside markdown files
>
> Images should be placed next to the markdown file using them.
>
> All of them will be copied over to the `docs` folder either in `/` (in the case of a single MD/PDF file) or following the same folder structure as in `src`, so make sure they have unique names.

Abstractions used


Software System
Container
(e.g. client-side web app, server-side web app, console application, mobile app, database schema, file system, object store, etc)
Component
Code

A **software system** is made up of one or more **containers** (applications and data stores), each of which contains one or more **components**, which in turn are implemented by one or more **code** elements (classes, interfaces, objects, functions, etc).

Person

However you think about your users (as actors, roles, personas, etc), people are the various human users of your software system.

Software System

A software system is the highest level of abstraction and describes something that delivers value to its users, whether they are human or not. This includes the software system you are modelling, and the other software systems upon which your software system depends (or vice versa).

Container

A container represents something that hosts code or data. A container is something that needs to be running in order for the overall software system to work. In real terms, a container is something like:

- Server-side web application: A Java EE web application running on Apache Tomcat, an ASP.NET MVC application running on Microsoft IIS, a Ruby on Rails application running on WEBrick, a Node.js application, etc.
- Client-side web application: A JavaScript application running in a web browser using Angular, Backbone.JS, jQuery, etc).
- Client-side desktop application: A Windows desktop application written using WPF, an OS X desktop application written using Objective-C, a cross-platform desktop application written using JavaFX, etc.
- Mobile app: An Apple iOS app, an Android app, a Microsoft Windows Phone app, etc.
- Server-side console application: A standalone (e.g. "public static void main")
- etc

Component

Component The word "component" is a hugely overloaded term in the software development industry, but in this context a component is simply a grouping of related functionality encapsulated behind a well-defined interface. If you're using a language like Java or C#, the simplest way to think of a component is that it's a collection of implementation classes behind an interface. Aspects such as how those components are packaged (e.g. one component vs many components per JAR file, DLL, shared library, etc) is a separate and orthogonal concern.

An important point to note here is that all components inside a container typically execute in the same process space.