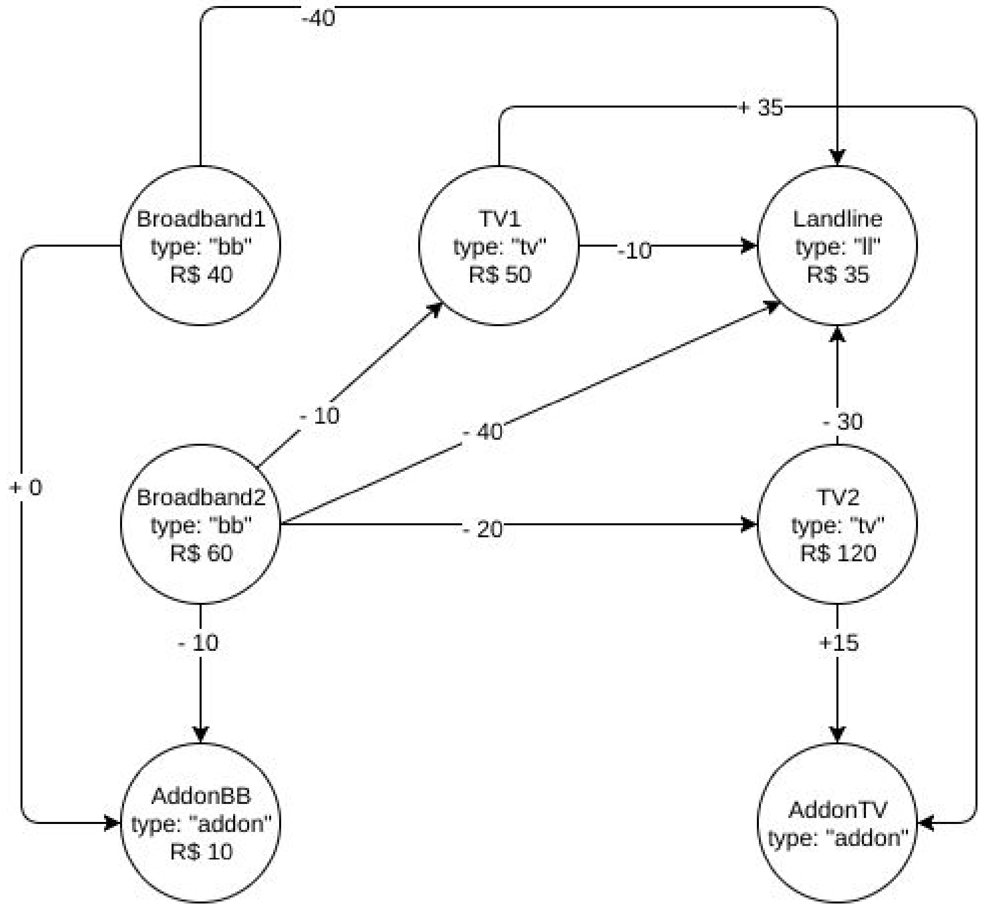
Fullstack Test

Melhor Plano - 2018

Context

One of the issues when selecting a new telecom bundle for your house is the amount of possibilities to combine plans and bundles to get what you need. The diagram below is a simplified version of a bundling scheme that a service provider might offer for Broadband, TV and Landline.



In the diagram, bundles can be mounted starting from any of the nodes and following the possible connections between nodes. The value on the connections represent additional price (positive numbers) or discounts (negative numbers) to get a bundle. Bundles can have several addons, but cannot have more than one broadband (type: "bb"), tv (type: "tv") or landline (type: "ll"). Each node has a "type", "price" and a name. If price is not present, price is equal to zero.

For example:

Starting on node "Broadband2" (costs 60 by itself), we can add the "AddonBB" (cost 10 by itself), but has a discount of -10 if bundled with "Broadband2", also, from "Broadband2" we can add a Landline plan (costs 35 by itself), but has a discount of 40 when bundled with Broadband (this actually happens very often, as a way to incentivise consumers to acquire a landline plan with the broadband plan).

Plan = Broadband2 + AddonBB + Landline
Price = 60 - 10 + 10 -40 +35 = 55

Tasks

1. Create a JSON representation of the diagram below. Make it flexible enough so that the structure can easily represent many "diagrams" in this JSON structure you've chosen. Save it as a file for this exercise, there is no need to use a database to store it.

2. Using Node, create one REST interface "/list-all-broadband" that should mount all possible combinations for a broadband, including standalone plans and bundles. The interface returns a JSON listing all combinations, ordered by price. The algorithm for mounting the boundles should be flexible enough to work with different "diagrams".

Example of combinations: (Broadband1), (Broadband2), (Broadband1 + AddonBB), (Broadband1 + AddonBB + TV1 + Landline)...

3. Using React, create a simple page that consumes the interface you've created on 2, listing each item in a separated card. The card should have the name of the items and the total price. The UI should be understandable, with minimal css, there is no need to put a lot of effort on the aesthetics, the goal here is just to evaluate how you organize the consumption of a REST interface.

Extras:
- Use ES7 or Typescript
- Sass or other css preprocessor
- Try to impress us :)

Tips

Use a react + node starter kit to make it easier to put together a full-stack project. However, make sure you understand the components.

Use clean and short functions.
Avoid abbreviations on variable names, unless it is a counter.
A summary for writing clean code: https://www.youtube.com/watch?v=UjhX2sVf0eg

Delivery

Upload your solution to git and send us the link.