Two-day Prototyping Exercise

Overview

Thanks for doing this prototype. I hope you have fun and feel free to ask questions to help you through this work!

This is related to a recent real project we did and prototyped so I hope the explanations are not too confusing. We'd like feedback on this document as well.

The purpose of this is to see what you can deliver yourself with no constraints. You can use any tools and tech you like. You can use cloud services or even go completely serverless if you wanted.

The important concept of a prototype is to build something really fast in order to learn more about the problem, the edge cases, and validate the requirements with the customer. Sometimes what people thought was a requirement ends up not really being one, so we use this approach to reduce waste from wrong or missing requirements.

For this purpose, we don't care about the polish in the interface or code quality, we care about actual working functionality.

The tools you choose should be ones that allow you to deliver the most functionality in the least amount of time. They don't have to scale, the plan is to refactor them out or throw them away once the learning is gathered from the prototype. To put this another way, if a tech or tool can support 1 user at 1 time with < 100 pieces of data, it's good enough.

You will need to host 2 apps and provide us 2 URLs for testing. We will test the app together with you and discuss what you've done at the end.

The Product

You decided to automate your home with smart speakers, lights and all of your screens. You've got many brands of devices that each use different APIs to talk and so you want to build an admin app to configure your devices in your home and a mobile web app to actually do the controlling.

The details on how the admin configures the controls is further in this document. The mobile web app is really simple and just has a list of devices, and for each one, a way to see the current controls states and the ability to modify their values.

The mobile app is fully configurable from the admin - the list of devices and even the controls displayed for each device is dynamic. Changes to controls in the mobile app should be visible in the admin as well.

Designs for the mobile web app are presented first, then an explanation below:

You'll want to build a web app that you can browse on a mobile device. It doesn't need to exactly match the designs, but if you can have a mobile feel that is great.

Features & Requirements

As described above, users of this app can register their devices in the system, and for each one they should be able to specify their name (e.g.: “living room Sonos”), an IP address and their type.

The reason we want devices to have a type, is so we know which controls to display in the mobile web app for each of them. For example, the type “Sonos Audio” may have a volume slider, a power button and a songs playlist.

These device types aren’t just for knowing which controls to use though, as they will also determine the HTTP API needed to control the device as well. So, for example, a “Sonos Audio” device may have the same ui controls as a “Spotify Audio” device, but they use different APIs.

Controls

We want this prototype to support these basic controls:

- **Slider** (shown in the screenshot above, used to pick one value from 0 to 100)
- **Select** (not shown in the screenshot above, used to pick an item from a list of items)
- **Buttøn** (not shown in the screenshot above, used for triggering a device command)

The controls that each device type should support needs to be configurable in the admin, choosing one or more controls from a list of the available ones for each device type.

For each control that we assign for a specific device type, users should also be able to add a custom name for each control.

For example, when configuring a “Sonos Audio” device type, you decide you want a slider control to represent the volume, so you pick a slider and name it “volume”. Then you add a “select” control to represent a playlist of songs, and name it “playlist”, and lastly you add another “select” control and this time name it “EQ”, so you can pick from a fixed list of available equalizer presets.

When adding a “select” control for a device type, besides picking the name for it, you’ll also need to configure which elements that particular list has. Each element only consists of a String name. For example, an “EQ” “select” control could have the following items within it: “rock”, “pop”, “jazz”, “balanced”.

Initial Configuration

Here is an initial configuration, although this should be editable in the admin interface:

Device types:

- Samsung Audio
    - Power (button)
    - Volume (slider)
    - Playlist (select) (should have 2 song names or more to pick from)
- Sony Audio
    - Power (button)
    - Volume (slider)
    - Playlist (select) (should have 2 song names or more to pick from)
- Apple TV
    - Power (button)
    - Brightness (slider)
    - Volume (slider)
- Citrus Lights
    - On/Off (button)

Registered devices in the system:

- “Bedroom Apple TV” (Apple TV)
- “Livingroom Player” (Samsung Audio)
- “Livingroom Lights” (Citrus Lights)

You will need to figure out where to store the values of each control so that when a specific device’s control is modified, that value is stored and reflected in the admin panel.

Admin web app

The admin app is where you configure controls, devices & device types. You can use any technology that you want. We didn’t provide wireframes for this.

When you demo the prototype, the sample configuration from above should be present (with any reasonable modifications) and should be editable. We will go in and tweak things and make sure the mobile web app UI updates accordingly.

You are the only user and don’t need to worry about authentication.

The admin needs to be able to view, add, edit and delete the following items:

- Controls
- Device Types
- Devices

Use Cases for Testing

1) Change the volume in the mobile web app:
    a) Change the volume or brightness of the Apple TV device.
    b) Refresh the admin app and see the volume & brightness for that device updated.
2) In the admin panel, change a control from a slider to a button, refresh the mobile app and see the changed control.
3) In admin panel, add a control to a device type:
    a) In mobile app refresh and see the additional control on devices of that device type
    b) Change the value of the control and see it stored somewhere in the admin panel.
4) In the admin panel, add a new device type, and then add a new device of this type.
    a) In the mobile web app, see the new device, see that its controls match those in new type.
    b) Make changes to controls in new device, see them in the admin panel.

Sending Commands

You still aren't going to actually send HTTP commands for all these devices, but you can nonetheless prototype this part as well, for one of them.

You should only tackle this part after all other functionality is complete. If you don't have enough time to implement this part, simply document what changes you would make to your code if you did have enough time.

When configuring a device type, besides picking which controls it needs, you should also associate each control with a specific HTTP call. For the purpose of this prototype, this HTTP call should simply be an HTTP verb + URL (e.g.: `PUT foo.com/volume`). No need to think about auth, headers or anything else.

For the "Citrus Lights" device described in the "Initial Configuration" section, we've created a very simple API that you can use. This is just a mock API with only 1 endpoint, used for turning the lights on and off. This endpoint is the following:

POST http://automation-prototype.herokuapp.com/citrus-light/power (Returns a 202 if successful)

When adding the “Citrus Light” device type, you should be able to specify that this type uses as its only control an “On/Off” button, and that when that button is activated, the endpoint above is called.

Beware though, the Citrus-Light API is unfortunately not well documented, and a priori you don’t know how well it works or if there are any problems with it, so a bit of testing may be needed. If you find any problems with this API endpoint, try to work around them! :-)

For all other devices in the system, you can leave the HTTP call action blank. We will only test the Citrus-Light device, checking with the access logs of the API to monitor the requests coming in.