Project of Software Development

Lab 2 - 24Feb
Programming Exercise: OutSify

Lab topic

OutSystems Dynamic pages:

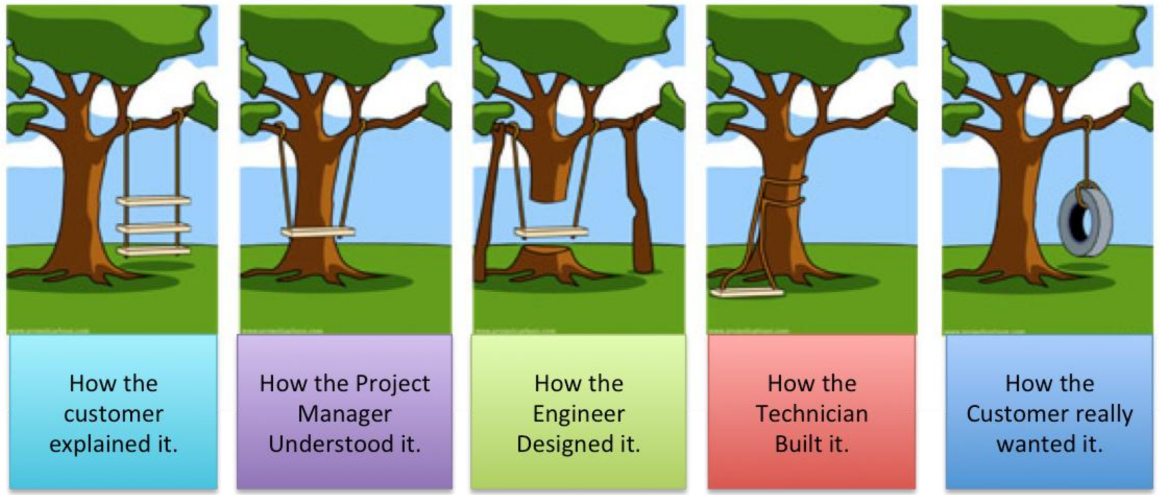
- Aggregates 101
- Building Screens with Data
- Modeling Data Relationships
- Data Model Integrity

Exercise:

- OutSify:
  - Finish the static version
  - Dynamic version

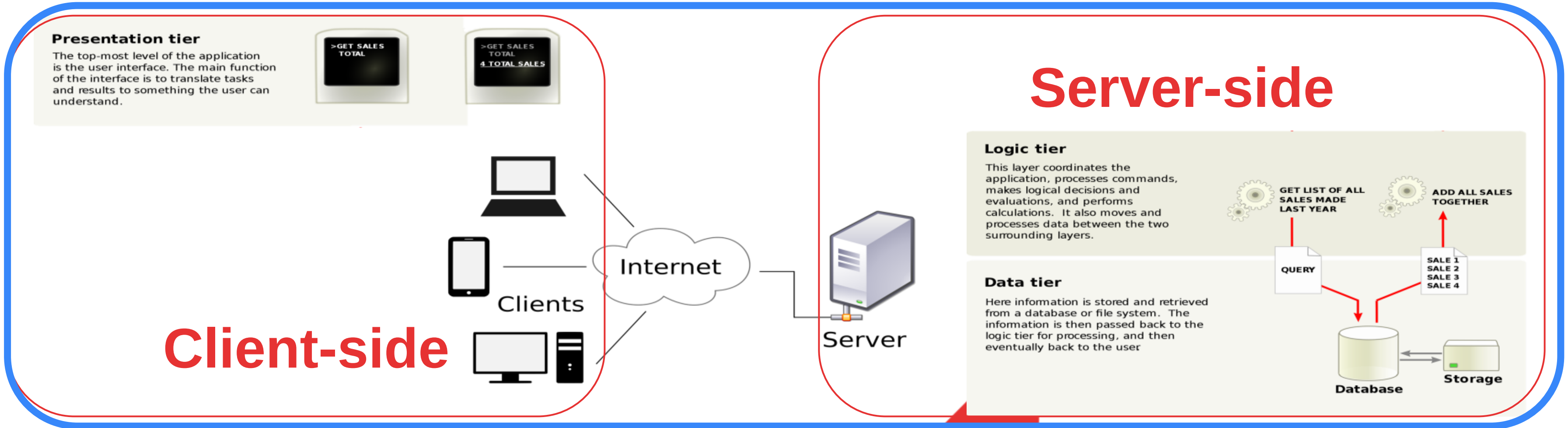
Bibliography:

- OutSystems online training:
  https://www.outsystems.com/training/courses/125/logic/?LearningPathId=18


How the customer explained it.
How the Project Manager Understood it.
How the Engineer Designed it.
How the Technician Built it.
How the Customer really wanted it.

OutSystems Online Training: Becoming a Reactive Web Developer

Short review and Q&A: Dynamic pages & exercise solution


Presentation tier
The top-most level of the application is the user interface. The main function of the interface is to translate tasks and results to something the user can understand.
>GET SALES TOTAL
>GET SALES TOTAL
4 TOTAL SALES
Clients
Internet
Client-side
Server
Server-side
Logic tier
This layer coordinates the application, processes commands, makes logical decisions and evaluations, and performs calculations. It also moves and processes data between the two surrounding layers.
GET LIST OF ALL SALES MADE LAST YEAR
ADD ALL SALES TOGETHER
QUERY
SALE 1
SALE 2
SALE 3
SALE 4
Data tier
Here information is stored and retrieved from a database or file system. The information is then passed back to the logic tier for processing, and then eventually back to the user
Database
Storage

OutSify - OutSystems Spotify (Dynamic version)

Website Architecture

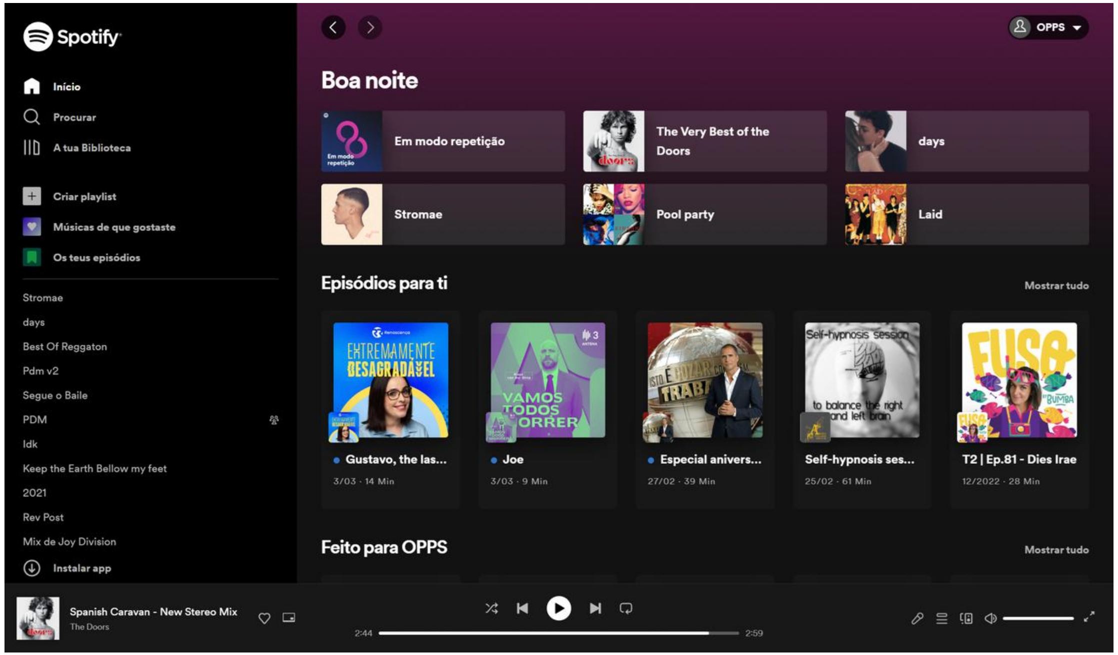
Functional aspects: Create a Web Media Player which imitates Spotify:

- **Dynamic** Homepage with playlists **from database**
- **Dynamic** Detail screen to show and play the playlist content, including some stats, **from database**

Visual appearance: use the Design Guidelines defined by Spotify

Technical constraints: 1 screen for the Homepage and 1 screen for all details page

Dependencies: all lectures content and all homeworks up to and including Lesson 7



ISEG Lisbon School of Economics & Management Universidade de Lisboa

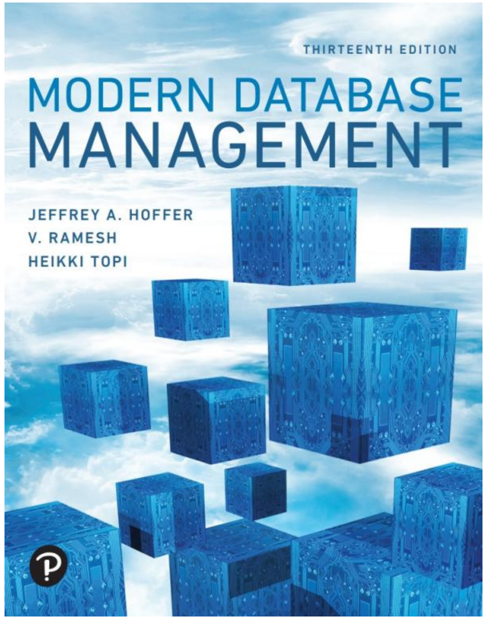
U LISBOA | UNIVERSIDADE DE LISBOA

Step 1 - Create the entities

- You will need to store the playlists and the songs of each playlist in the database
- All information (name, author, album, link to Spotify or youtube, etc.) must be in the database
- All entities must be in the 3rd normal form (Hoffer, chapter 4, "Introduction to normalization, pages 176-185)
- All entities must have the keys defined (primary and foreign)

Minimum requirements:

- Insert at least 7 playlists records and at least 2 records of songs per playlist using Service Studio's "Data editor"



Step 2 - Create a new dynamic homepage

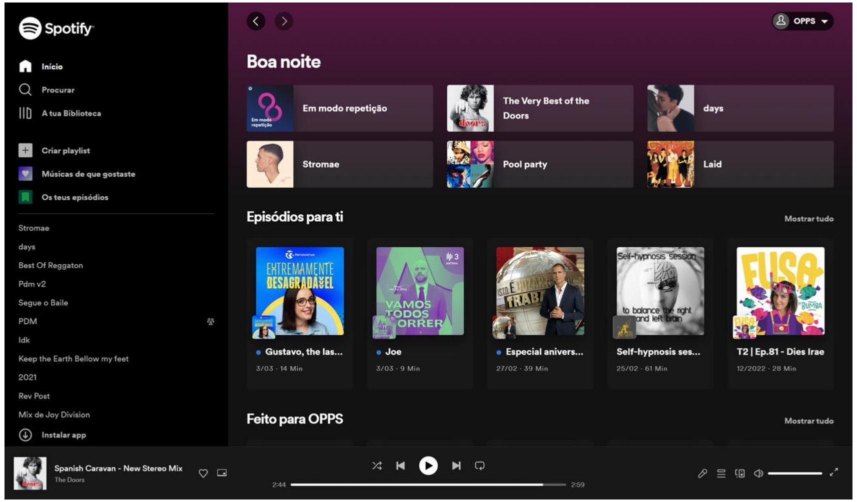
Display a dynamic list of playlists instead of the hard-coded interface. No playlist information (name, etc.) can be hard-coded

Use an icon for the playlist image.

Minimum requirements:

- Top-left logo
- 5 playlists from the database
- Page title
- After implementing the “playlist page”:
  - Highlight the last played playlist in the “Top 6 playlists”
  - Menu option to reset the count of plays

Tip: In the gallery, insert a list



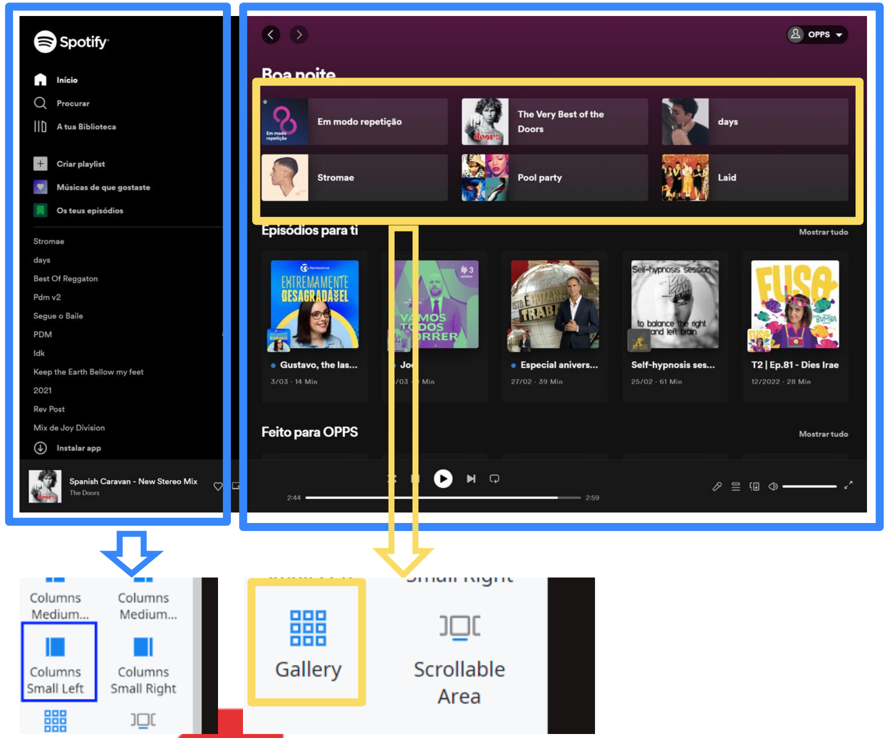
Step 2 - Create a new dynamic homepage (solution)

Think about the visual structure (organization) of your page

The menu is on the left side, the content is on the right side (blue)

The playlists are displayed in 3 areas equally sized (yellow)

- Use the Gallery widget to show a dynamic list of items and allow users to sequentially browse content



Step 2 - Create a new dynamic homepage (solution)

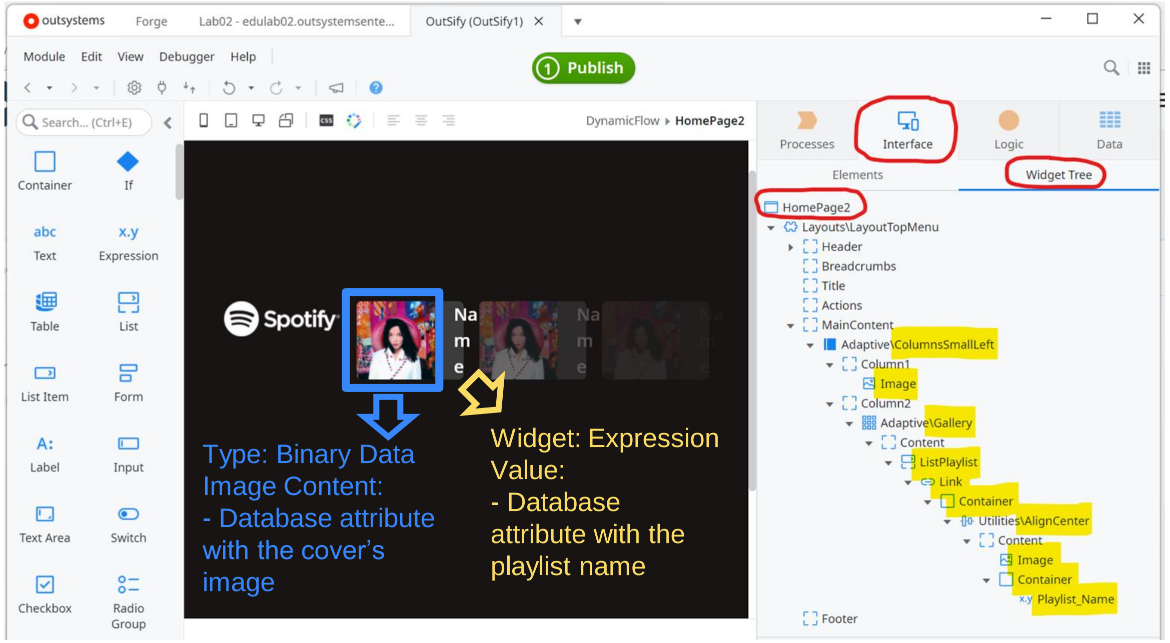
In the gallery, insert a list

Fetch all the playlists (don't forget to sort them) from the database using an aggregate

In the list:

- Set source = the aggregate
- Recreate the style of the static playlist items but with the **dynamic content** from the aggregate (instructions here on the right)

Test it on your smartphone too!



Type: Binary Data
Image Content:
- Database attribute with the cover's image

Widget: Expression
Value:
- Database attribute with the playlist name

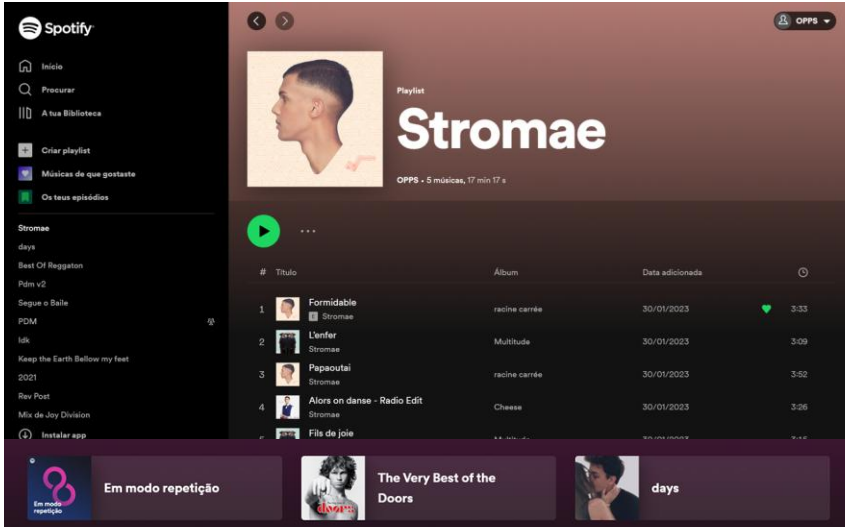
```
HomePage2
  Layouts\LayoutTopMenu
    Header
    Breadcrumbs
    Title
    Actions
    MainContent
      Adaptive\ColumnsSmallLeft
        Column1
          Image
        Column2
          Adaptive\Gallery
            Content
              ListPlaylist
                Link
                  Container
                    Utilities\AlignCenter
                      Content
                        Image
                        Container
                          Playlist_Name
    Footer
```

Step 3 - Create a new dynamic playlist page

Display a dynamic list of the playlist’s songs from database instead of the hard-coded interface. No song information (name, author, album, link to Spotify or youtube, etc.) can be hard-coded

Minimum requirements:

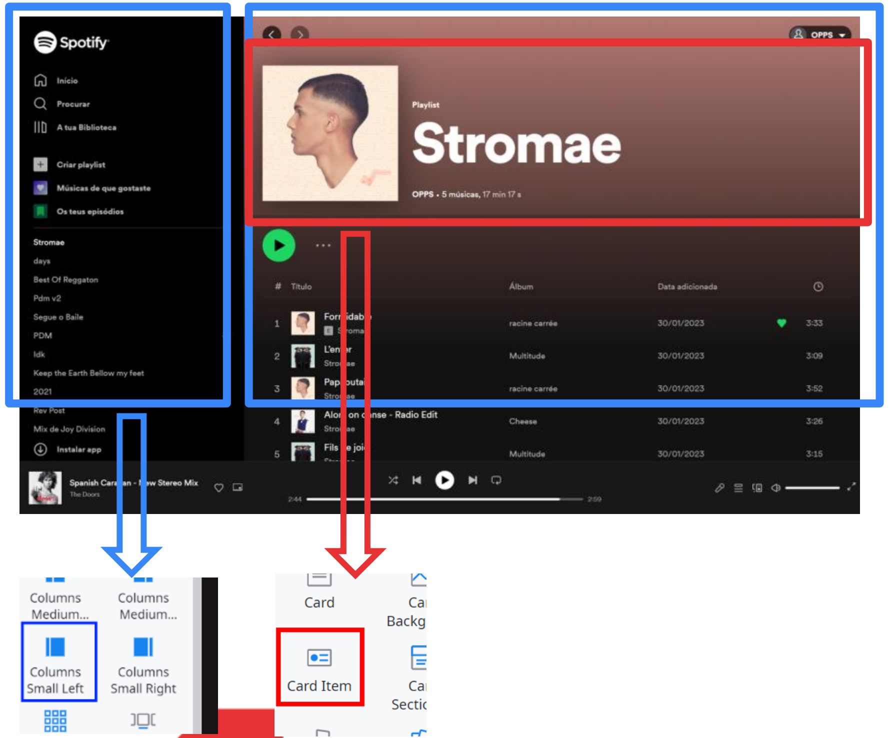
- Top-left logo
- Link to homepage on the menu
- Link to each song in the playlist (2 songs):
    - To Spotify, using the Spotify URI from Step 1
    - To Youtube, opening a new page to play the videos
    - Page title including the playlist name
    - Same “Top 5 playlists” from Homepage



#	Título	Álbum	Data adicionada	🕒
1	Formidable Stromae	racine carrée	30/01/2023	3:33
2	L’enfer Stromae	Multitude	30/01/2023	3:09
3	Papaoutai Stromae	racine carrée	30/01/2023	3:52
4	Alors on danse - Radio Edit Stromae	Cheese	30/01/2023	3:26
	Fils de joie			

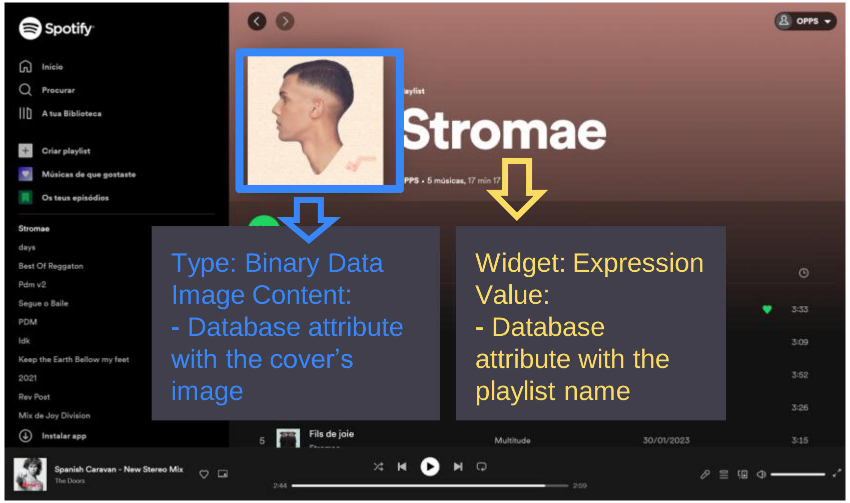
Step 3 - Create a new dynamic playlist page (solution)

1. Use the same structural elements as in the static version: ColumnsSmallLeft (blue) and CardItem (red).
2. For the CardItem content, set the images and the playlist name as dynamic, using the attributes of a new aggregate (same as we did for the playlist element in the Homepage screen).
3. This aggregate must fetch the playlist details of the playlist defined by the input variable.
4. Set the Title property of the Playlist Screen as dynamic: "OutSify - featuring the playlist " + *The Name of the Playlist in the database*



Step 3 - Create a new dynamic playlist page (solution)

1. Use the same structural elements as in the static version: ColumnsSmallLeft (blue) and CardItem (red).
2. For the CardItem content, set the images and the playlist name as dynamic, using the attributes of a new aggregate (same as we did for the playlist element in the Homepage screen). This aggregate must fetch the playlist details of the playlist defined by the input variable.
3. Set the Title property of the Playlist Screen as dynamic: "OutSify - featuring the playlist " + *The Name of the Playlist in the database*



Type: Binary Data
Image Content:
- Database attribute with the cover's image

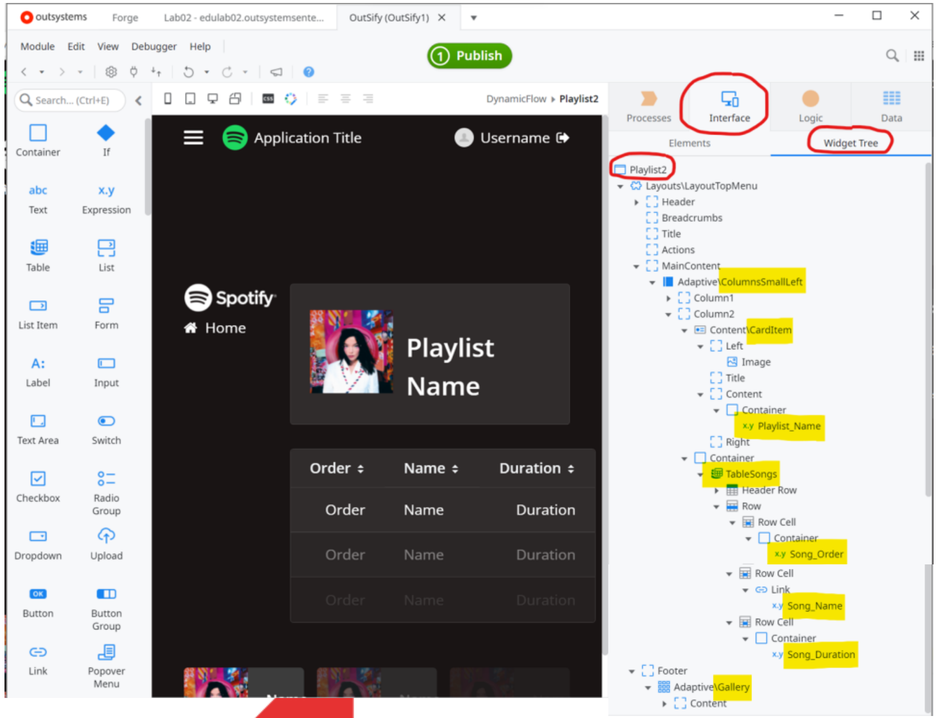
Widget: Expression
Value:
- Database attribute with the playlist name

Step 3 - Create a new dynamic playlist page (solution)

Fetch the songs in the playlist (don’t forget to filter and sort the playlist) from the database using an aggregate

In the “Table”:

- Set source = the aggregate
- Add 3 columns:
    - Order with width=1 col
    - Song name with width=8 col
    - Duration with width=3 col



Step 4 - Playlist counter and other cool features

Playlists gallery:

- With a count of how many times each playlist has been played
- Excluding the playlist being played

Tip:

1. Use the widget “Numbers\Badge” to display the count. The default background of this widget is white and the font colour of your App is white, so the number will not be visible. Therefore, you need to change the background using the widget’s attribute “Color”
2. Create a “PlayCounter” attribute in the Playlist entity to store the number of times each playlist has been played. Where should it be incremented?

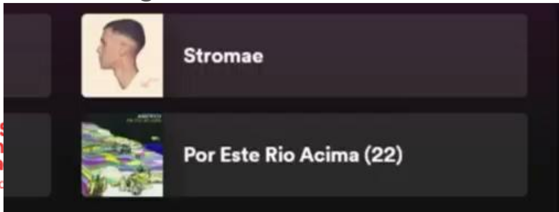
Other optional cool features

Set the WebApp icon:

- Change the icon in Data/Resources/favicon.png (right-button and “Change Resource”)
- Replace all occurrences of “favicon.png” with “/<WebAppName>/favicon.png”

Imitate Spotify’s interface: show a play button when the mouse is over the playlist (link hover) [Requires knowledge of CSS]:

- Search for the Spotify Play Button (transparent PNG, 50px) and add this image to your module
- Google for “css show image on hover” and StackOverFlow



Learning goals

To understand how to create Dynamic Web pages in OutSystems

Creating a Web Application in OutSystems: Entities, fetching data, lists and galleries

How to develop a Reactive WebApp with content that adapts to screen size

Homework: exercise

- Finish your OutSify (Dynamic) exercise:
  - Don't forget to test on both your laptop and smartphone. The expected result shall be similar to https://edulab02.outsystemsenterprise.com/OutSify/Homepage2
  - Add the "anonymous" role to all your Web Screens:
    - Click on the Web Screen (widget tree on the right side)
    - The attributes area will open. Click on "Anonymous"
    - Repeat (a) and (b) for each Web Screen you created
    - Publish and test
  - Submit the address of your Web Application by following up on the email

Expected total effort: 30 to 120 minutes