# EDP Challenge Project 1

In this lab, we're going to exercise the HTML, CSS, JavaScript, and Git knowledge you've gained. We're going to build a front-end user interface (UI) that interacts with a Star Wars API (SWAPI). We'll do this in pairs in breakout rooms.

# The Resulting Site

The result will be a web site that begins with a list of Star Wars characters.

A screenshot of a computer

Description automatically generated

Clicking on a name will take you to the details page for that character. Each character has a home world and a list of films they've appeared in.

A screenshot of a computer

Description automatically generated

You'll be able to click on film and navigate to the details page for that film. The film will have a list of characters and a list of planets. These also are clickable so the user can get details for them.

A screenshot of a computer

Description automatically generated

As you can see, we're allowing the user to navigate between people, planets, and films. You and your partner are going to create this scheme. Partner 1 will create the films page. Partner 2 will create the planets page. We've provided the characters page (index.html) as a starter example. See **guided\_project\_1\_starter.zip** in Basecamp the starter example.

# Using GitHub to collaborate with your team members

Your team will use one GitHub repository for collaboration. Partner 1 will set up a **new** GitHub repository using the same GitHub account that was used in the DevOps Foundations course. In GitHub, **invite** the other team member as a **collaborator** so that they can push.

A screenshot of a computer

Description automatically generated

# Getting the starter

Partner 1 should get the starter files, **guided\_project\_1\_starter.zip,** from Basecamp. They should then commit the starter files to their local repository and push to the GitHub repository. Partner 2 can then pull those into their local repository.

Both partners can now run the project by opening a terminal window in the folder where you have all the HTML, CSS, JS files and running:

$ npx browser-sync start --server --files "\*.html,\*.js,\*.css"

You'll know it's working when you can browse to <http://localhost:3000> and see the list of people.

# Writing the new pages

Partner 1 will create the films page:

* Create a new development branch in git. Do all your development in this branch. Feel free to push your changes to the upstream repo at any time through this process.
* Create films.html and films.js.
* Read the film ID from the querystring (ie. get the 10 from "http://localhost:3000/films?id=10"). You can do that by using URLSearchParams like this: const sp = new URLSearchParams(window.location.search); and then const id = sp.get('id'); YYY
* Using the fetch API, make a GET request from:  
  ` https://swapi2.azurewebsites.net/api/films/${id}`
* There will be exactly one film in the response. Read it into a JavaScript object called film.
* Make another GET request from ` https://swapi2.azurewebsites.net/api/films/${id}/characters` to get the characters in that film.
* Make one last GET request from ` https://swapi2.azurewebsites.net/api/films/${id}/planets` to get the planets for that film.
* Display whatever details you like on the page. At minimum, you must display the characters and the planets. The character/planet name must be a clickable hyperlink that navigates the user to the details page for that entity.

While Partner 1 is writing the films page, Partner 2 will do the exact same thing but with planets. Planets must link to all films and all people.

# Merging the branches

Once each partner's page is working properly, they should merge their changes to the main branch and push to the upstream repo. We are setting you up for conflicts. This is your opportunity to get experience in merging cleanly. Feel free to do this along with your partner so you can talk through the conflicts and how to merge them.

When you have a synchronized codebase in all three places (two local and one remote repository) and they're running as required, you are finished.

# Bonus work

If you finish early, do these things for extra credit.

Make the pages look good with CSS. Try to write one CSS file for the entire site, not one for each HTML page. Feel free to change the layout.

Here's an advanced assignment: Store a few arrays in local storage; planets, films, and people. Each one should simply store the entity's ID and name. Populate the arrays as you make your GET requests. That way, when you're creating links, if you already have the entity name in the array, you won't have to fetch it again. You're just saving data locally, so you don't have to go the API server every time.