**Week 1**

Diagram

Description automatically generated

ES5 configuration file – 3 settings described:

WebIDEEnabled=true => makes our destination visible to SAP Web IDE

WebIDESystem=ES5 => system ID the service runs on

WebIDEUsage => contains usages and you can enter multiple usages for a destination, separated by commas without spaces

But… the config settings above are not that useful, are they? Since “Portal” cannot be enabled.

Moreover, this IDE is not used anymore, or at least not like in the video anymore, so we can just skip.

Not skip, but just watch the videos until you see some ui5 content, and focus there.

Seems like you can run an application on the “cloud” anyway, from the SAP Web IDE:

Graphical user interface, text, application, email

Description automatically generated

Click the Green PLAY round button positioned in the top bar.

Quiz related. *(got 4.5/6 points on this)*

What do you need destinations for?

* To connect to cloud services or on premise systems in a secure and standardized way
* For accessing remote OData services, for example on the SAP Gateway demo system ES5

Unit 3 – Local setup

Setting up the development environment:

npm install -g @ui5/cli (to check if you have it, just use ui5 –help)

git clone <https://github.com/SAP/openui5-basic-template-app.git>

npm install

npm run will tell you the commands you can use

* start = starts a server
* test = executes the test suite which you can use for your own CI. (gr8 to keep quality high)
* lint = executes a static code check
* karma = Is actually more awfseq
  + (if you don’t believe me, check for yourself on unit3 video, at min 04:38 :D)
* watch = task supporting your test-driven development
* build = to deploy the application in the end, somewhere

**Related to the code editor:** Visual Studio Code

**Recommends the plugin:** UI5 Snippets & Extensions, UI5 Explorer, ESLint

**There is one obstacle you need to overcome for your local development:**

Browsers normally follow the same origin policy and blocks the so called CORS requests by default.

**CORS = cross-origin resource sharing**

* enables the flexibility to fetch data from other hosts than the actual web server the app is running on.

**Sooo, we gonna use a proxy:**

In **package.json**:

scripts ->

"proxy": "node proxy.js"

devDependencies ->

"cors-anywhere": "^0.4.1"

Then, we must prefix our service application/URL, with the actual proxy URL, to be able to reroute the CORS via that proxy.

http://localhost:8081/

Got error when trying to run the server: errno: -4094,