# A template application of the UD-Viz package

This repository holds a template (demonstration) application of the [UD-Viz](https://github.com/VCityTeam/UD-Viz) JavaScript package. The goal of this template application is to

* illustrate the main features of [UD-Viz](https://github.com/VCityTeam/UD-Viz),
* provide code that demonstrates how such features can be configured/extended/embeded and eventually combined/integrated within a full autonomous application,
* illustrate the JavaScript ecosystem required for building and running it,
* be used as a template for creating/declining your own application.

Because this template application is fully functional maybe the simplest way to understand what it does is to build it and run it.

## Install npm

For the npm installation refer [here](https://github.com/VCityTeam/UD-SV/blob/master/Tools/ToolNpm.md)

Required npm version: UD-Viz-Template has been reported to work with npm versions npm 6.X.

Reminder: npm install -g npm@6.14.15 enables to [switch npm version](https://github.com/VCityTeam/UD-SV/blob/master/Tools/ToolNpm.md" \l "task-switch-npm-version)

## Installing and running the template application

The template application can be locally (on your desktop) started in the following way

npm install  
npm run debug # integrates building

and then use your favorite (web) browser to open http://localhost:8000/.

Note that technically the npm run debug command will use the [webpack-dev-server npm package](https://github.com/webpack/webpack-dev-server) that - runs node application that in turn launched a vanilla http sever in local (on your desktop) - launches a watcher (surveying changes in sources) - in case of change that repacks an updated bundle - that triggers a client (hot) reload

## Technical notes concerning the template application

Some modules used by the DemoFull require some server-side components to be installed on some server (possibly your desktop). For example \* the 3D objects (buildings) are (by default) serverd by a LIRIS server and thus require no specific configuratione there is nothing more to do \* handling of documents will require you to [install the API\_enhanced\_city](https://github.com/VCityTeam/UD-Serv/blob/master/API_Enhanced_City/INSTALL.md). \* you can also modify the [application configuration file](../../UD-Viz-Template/assets/config/config.json)

## Making your own UD-Viz based application

The present UD-Viz-Template repository holds all the required elements constituting an independent JavaScript application (using the UD-Viz package among others) as well as the technical means to build and run (and debug) that application. In order to realize your own UD-Viz based application it thus suffice to duplicate this repository and start adjusting, modifying, extending and deriving the code of your duplicate.

First create a new repository, e.g. https://github.com/exampleuser/MyApp.git (the git repository does not need to be hosted at github) to host your new application.

Then [replicate this git repository](https://docs.github.com/en/repositories/creating-and-managing-repositories/duplicating-a-repository) which can be done with e.g. the following commands :

# Create a scratch directory  
mkdir foo; cd foo   
  
# Make a bare clone of this repository  
git clone --bare https://github.com/VCityTeam/UD-Viz-Template.git  
  
# Mirror-push to the new repository  
cd UD-Viz-Template.git  
git push --mirror https://github.com/exampleuser/MyApp.git  
  
# Remove the temporary scratch directory  
cd ../..  
rm -rf foo   
  
# Cleanly clone your new repository  
git clone https://github.com/exampleuser/MyApp.git

You can then proceed with using your MyApp with exactly the same instructions as for this UD-Viz-Template application that is \* [npm install](https://github.com/VCityTeam/UD-Viz-demo" \l "installing-the-demo-applications) (install the dependencies) \* [npm run debug](https://github.com/VCityTeam/UD-Viz-demo/blob/master/README.md" \l "installing-demofull) (building and running the application) \* optionnaly you can lint your code with [eslint](https://eslint.org/) by running the npm run eslint command. This new repository now holds a buildable (npm install) and runnable (npm run debug) application (just follow the Readme.md as you did for UD-Viz-Template), that you can start adapting to suit your needs.

The main entry point in order to customization your new MyApp application is the [src/bootstrap.js file](https://github.com/VCityTeam/UD-Viz-Template/blob/master/src/bootstrap.js) that is centered on [UD-Viz’s Template.Allwidgets class](https://github.com/VCityTeam/UD-Viz/blob/master/src/Templates/AllWidget/AllWidget.js).

Then you can also adapt the [assets/config/config.json](../../../../../../assets/config/config.json) configuration file that defines e.g. \* links to the used assets for the icons, logos of your application, \* the extents i.e. the geographical portion of the territory that will be displayed, \* some default data streams used e.g. - the background\_image\_layer that define the terrain (through a [WMS (Web Mapping Service)](https://www.lib.ncsu.edu/gis/ogcwms) stream), - some 3d buildings (based on [3DTiles](https://github.com/CesiumGS/3d-tiles)) refer e.g. to the 3DTilesLayer entry, - the default camera position within the scene, - …

FIXME for all the bottom of this page

Ensuite trois cas d’utilisation:

1. on veut créer une demo a partir de brique existante mais on veut configuré lesquelles, dans ce cas on peut utilisé un template et lui filé la config adapté. comme pour le AllWidget template avec une config qui va décrire quel widget est utilisé. EBO: on va conserver la config ?
2. on veut créer une demo mais le code n’existe pas, dans ce cas la meilleure méthode est rajouter son code dans ud-viz et de dev avec les deux repo side by side. on peut aussi rajouter son code dans son projet et se demerder avec l’api proposé par les template pour y incorporer son code. typiquement on pourrait dev son propre widget (grace a du code + bas niveau si necessaire, dans ce cas widget) de ud-viz et ensuite l’ajouter via l’api de allwidget template.
3. le besoin n’est pas couvert par un template existant. pareil meilleure méthode créer le template dans ud-viz a partir de code + bas niveau de ud-viz (game, widget, views) et de dev side by side sinon dans son projet créer les classes manquantes a partir du code ud-viz plus bas (toujours widget, game, view)

quand je dis meilleure méthode c’est mieux car le projet ud-viz beneficie directement de features réutilisable par les autres dev, et ca évite une étape d’intégration si jamais on désirait l’intégrer plus tard.

### When working with a docker container: the [diff](https://en.wikipedia.org/wiki/Diff) alternative strategy

If you demo is defined within a [docker container](https://en.wikipedia.org/wiki/Docker_(software)) then an alternative strategy (to the complete replication of the DemoFull directory) consists in (within your Dockerfile) - cloning the UD-Viz-demo repository, - placing yourself (with [WORKDIR](https://docs.docker.com/engine/reference/builder/" \l "workdir)) inside the DemoFull directory, - overwriting the DemoFull code with your partial customizations (e.g. just overwriting BaseDemo.js and the config.json files).

A example of this docker container based strategy can be found in the [DatAgora\_PartDieu](https://github.com/VCityTeam/UD-Reproducibility/blob/master/Demos/DatAgora_PartDieu/) demo as illustrated by the [Dockerfile](https://github.com/VCityTeam/UD-Reproducibility/blob/master/Demos/DatAgora_PartDieu/ud-viz-context/Dockerfile" \l "L28) commands.