

Enterprise Information Systems Lab

Linked Data Visualization and Exploration

Mentors:

Klaudia Thellmann

Michael Galkin

Group Members:

Alina Arunova (L)

Tatiana Novikova

David Ibhaluobe



+Objectives of LinDaViz project

Goal:

- supporting non-technical users in finding suitable visualizations for a specified subset of the data

Objective:

- development of a largely automatic visualization workflow which guides users step by step through the process of creating visualizations



+ Requirements

Functional:

High priority:

- Display overview ✓
- Filtering ✓
- Searching ✓

Low priority:

- Hiding/Loading
- Sorting
- Export Data

Non-functional:

High priority:

- Data table scalability ✓
- Usability ✓
- Technical System ✓

Documentation

Low priority:

- Backup
- Robustness ✓



+ Architecture

The **LinkDaViz** tool is a JavaScript based web application
It receives as input data in RDF or tabular format.



The frontend module:

- a component for data selection,
- the visualization widgets library,
- a component for configuring a visualisation.



The **backend module** is in charge of computing
visualization recommendations, acts as a data storage for
visualization metadata and saved visualizations.

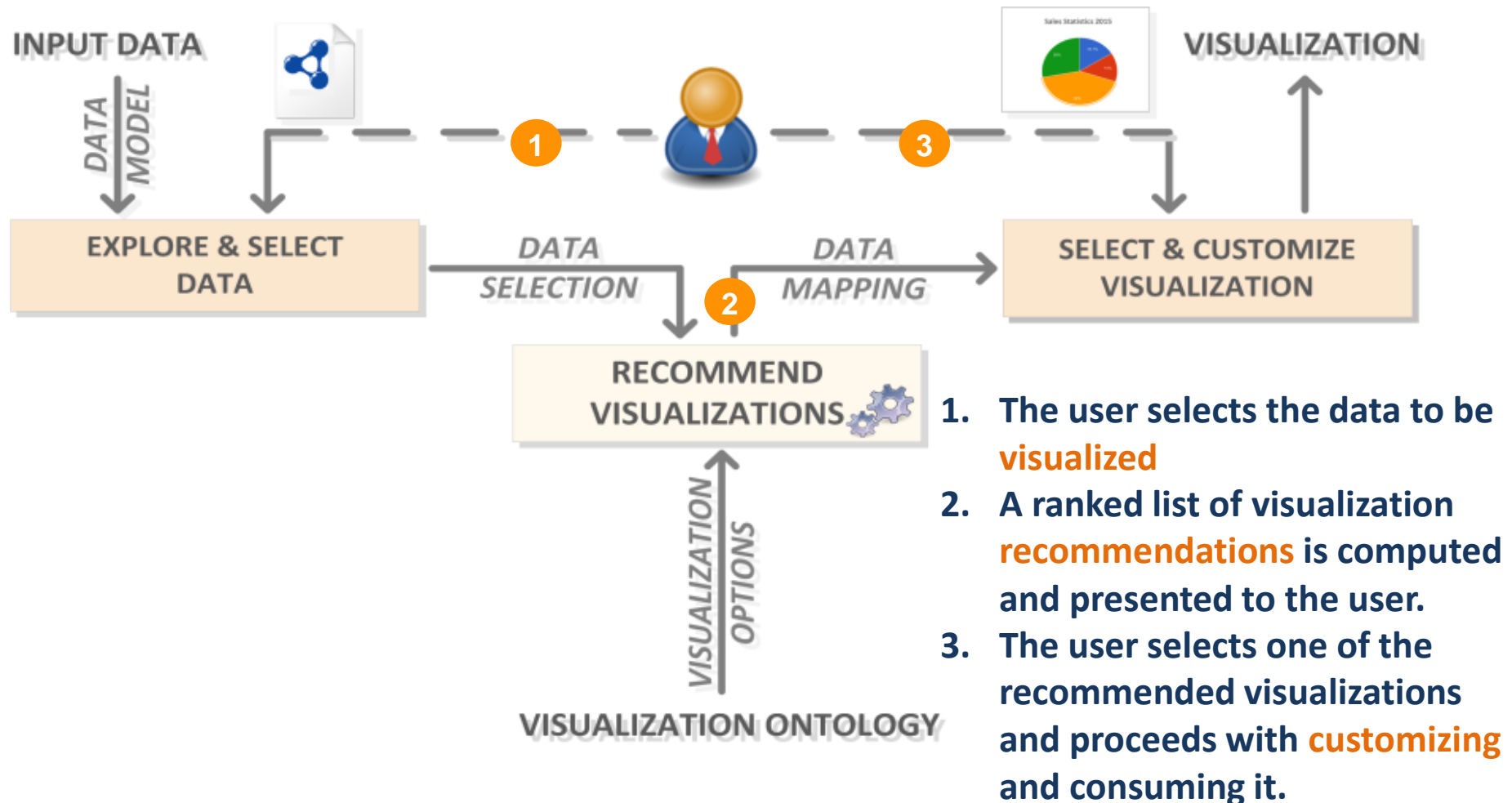


express


The **triplestore** contains the visualization ontology and
the datasets.



+ Visualization Workflow



+ Before implementation:

 LinDaViz

Explore and Select Data

- ☒ Observation class
 - ☒ Precipitation Deviation number
 - type resource
 - ☒ Month number
- ☐ dataSet resource
 - ☒ structure resource
 - type resource
 - ☐ label string
- ☒ Quality Deviation number
- ☐ ComponentSpecification class
 - type resource
- ☒ dimension resource
- ☒ label string
- ☒ measure resource
- ☒ DataStructureDefinition class
- ☒ MeasureProperty class
- ☒ DataSet class
- ☒ DimensionProperty class
- ☒ Ontology class


Preview Data Selection

Show entries

Search:

Precipitation Deviation	Month	Quality Deviation
-1.118	8	-0.06
-0.999	7	-0.028
-0.569	6	0.011
-0.323	4	0.02
-0.195	10	-0.046
-0.092	9	-0.629
-0.085	5	0.029
0.005	12	0.05
0.01	11	0
0.117	1	-0.046

Showing 1 to 10 of 12 entries

 Visualize

Previous 2 Next



Lab Topic

NOW: the tree view for browsing and selecting data does not scale to large datasets.

Problem: If the dataset contains classes with lots of properties you won't be able to get an overview this way.

The lab topic refers to the improvement of data selection and exploration part of the visualization workflow.

Responsibilities:

**Alina
Arunova**

- Keyword-based search in a tree
- Final presentation
- Documentation

**Tatiana
Novikova**

- Implementing filters for numerical values in a tree
- Final presentation
- Documentation
- Poster

**David
Ibhaluobe**

- Horizontal scrolling
- Final presentation
- Documentation

Time plan

Final presentation of the project

Checking with the requirements, final
run-through

- Technical system documentation

Testing, adding features if
necessary



- Hiding/adding

- Export data

Coding on JavaScript

- Filtering, keyword-based search, scrolling

Installing components for LinkDaViz

- Ubuntu OS
- Ember, Nodejs, etc
- LinkDaViz tool
- Requirements

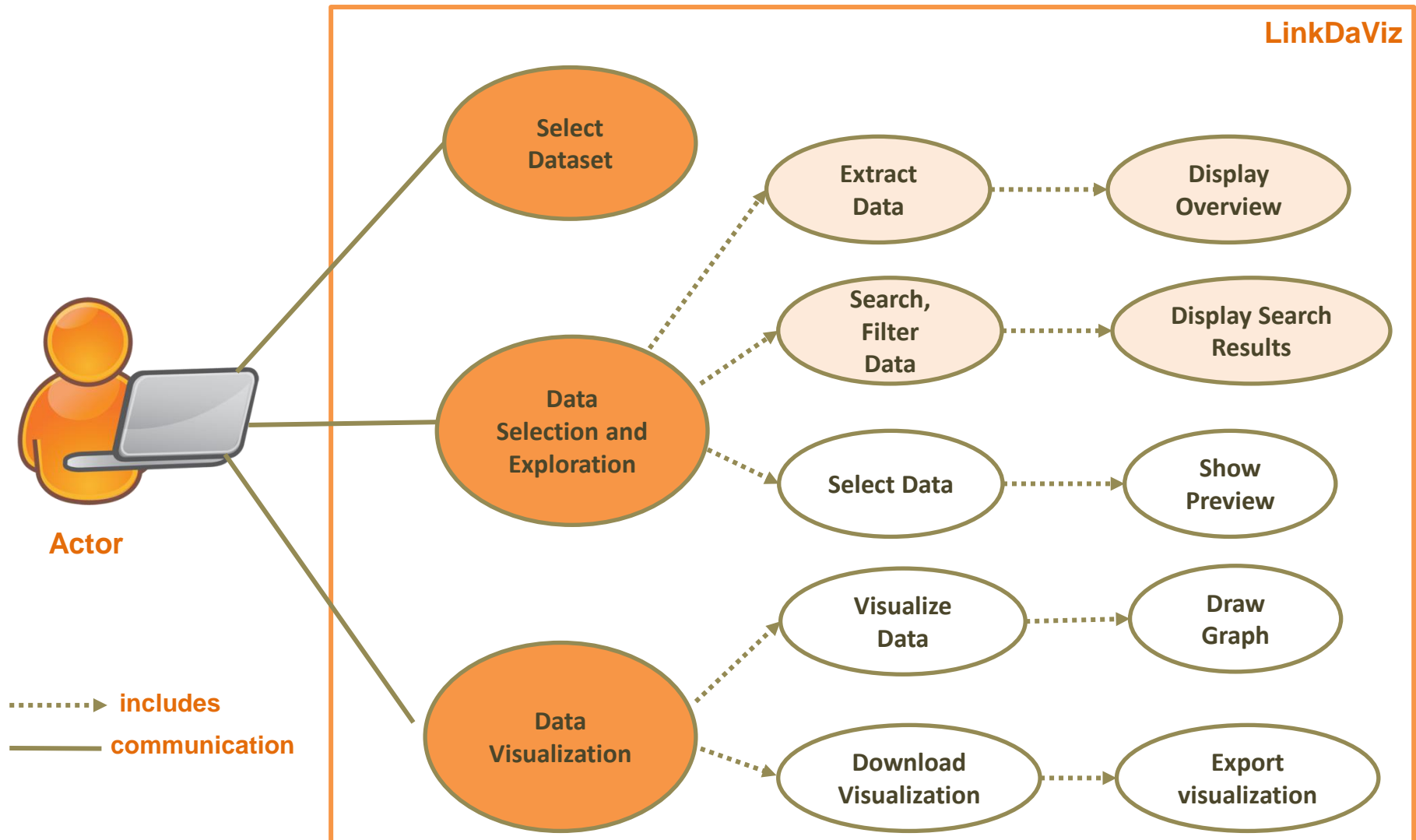
June

August

September

October

+ Use Case Diagram



+

Required tools:



1. **Nodejs** - JavaScript runtime built on Chrome's V8 JavaScript engine;



2. **NPM** - package manager for JavaScript, and is the default for Node.js;



3. **Docker** - is an open platform for building, shipping and running distributed applications.



4. **Nodemon** – utility that will monitor for any changes in the source and automatically restart the server;

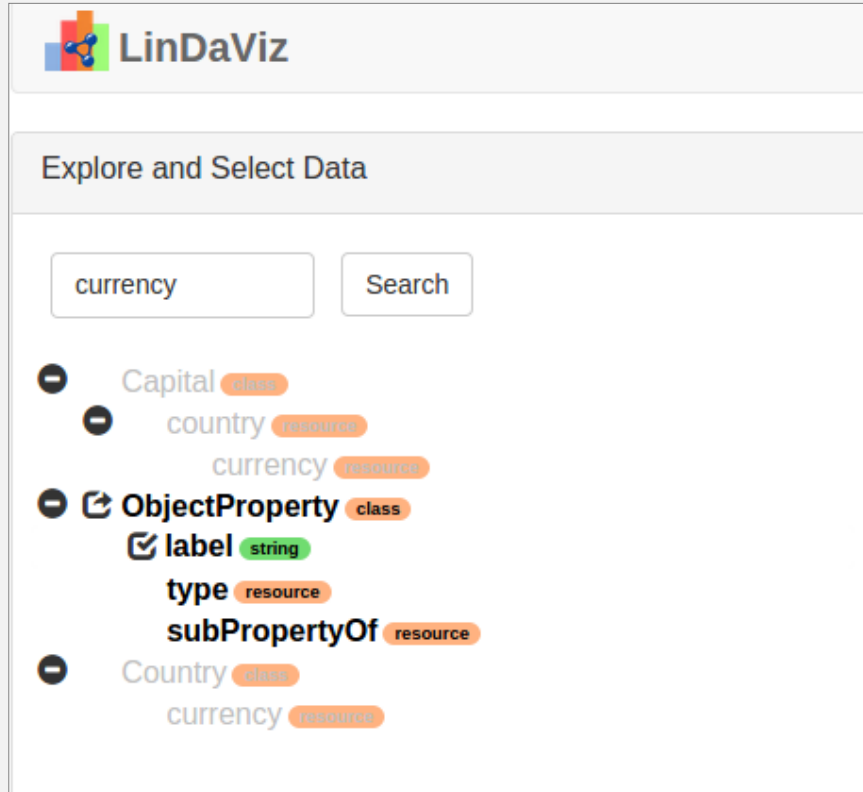


5. **Ember cli** - Ember.js command line utility, provides a powerful add-on system for extension;



6. **Virtuoso** - is a middleware and database engine hybrid.

Keyword-based search



Challenges:

- Support of different **ontologies** and **vocabularies**;
- **Depth** of the search;

How it works: a user types a keyword in a text field and presses a button “search” if there is a such word in the dataset then a new tree with searched results will be displayed, otherwise “No matches” will be displayed. If the user enters too short word, a hint will be displayed

Filtering

The screenshot shows the LinDaViz interface. On the left, under 'Explore and Select Data', there is a search bar and a list of data fields: DataEntry (class), frequency (string), value (number), location (string), source (string), type (resource), state_fips (number), state (string), data_item (string), and year (number). On the right, the 'Preview Data Selection' panel shows a table with columns: frequency, value, location, source, state_fips, state, and data_item. The table contains three rows of data. Below the table, there are 'Filters' for 'value' (Min: 300, Max: 800) and 'state_fips'. An orange circle highlights the 'value' filter dropdown and its input fields, with an arrow pointing to a zoomed-in view of the filter.

frequency	value	location	source	state_fips	state	data_item
END OF DEC	371	STATE	CENSUS	15	HAWAII	GEESE - INVENTOR
END OF DEC	722	STATE	CENSUS	24	MARYLAND	GEESE - INVENTOR
END OF DEC	556	STATE	CENSUS	50	VERMONT	GEESE - INVENTOR

Challenges:

- Triggering of **any changes** inside the tree;
- The necessity of tracking only **numerical values** from the tree;
- **Automatic update** of the data table after applying of filters;

This is a zoomed-in view of the 'Filters' section. It shows a dropdown menu for 'State Fips' with options 'State Fips', 'Year', and 'Value'. The 'Value' option is selected and highlighted in orange. To the right of the dropdown are input fields for 'Min: 300' and 'Max: 800'. A 'Visualize' button is located at the bottom right of the filter section.

How it works: a user checks data which he wants to visualize, the data appears in a data table, and the filter is being filled with numerical values in a drop-down list. A user then chooses a list item, types in min and max values and sees how the table changes in real-time.

Horizontal Scrolling

The screenshot shows the LinDaViz interface. On the left, the 'Explore and Select Data' panel lists various data properties: DataEntry (class), State Fips (number), Year (number), Source (string), State (string), type (resource), Value (number), Frequency (string), Data Item (string), and Location (string). Below these are Property (class), Dataset (class), and Document (class). On the right, the 'Preview Data Selection' panel shows a table with columns: State Fips, Year, Source, State, Value, Frequency, and Data Item. The table displays three entries, with the third entry (State Fips 4, Year 2007, Source CENSUS, State ARIZONA, Value 858, Frequency END OF DEC, Data Item GEESE - INVENTORY) highlighted. A horizontal scrollbar is visible below the table, indicating that the table is wider than the container. The scrollbar is circled in orange. Below the table, it says 'Showing 1 to 3 of 3 entries (filtered from 50 total entries)'. At the bottom, there is a 'Filters' section with a dropdown for 'State Fips' and input fields for 'Min: 1' and 'Max: 4'. A 'Visualize' button is at the bottom right.

State Fips	Year	Source	State	Value	Frequency	Data Item
1	2007	CENSUS	ALABAMA	3231	END OF DEC	GEESE - INVENTORY
2	2007	CENSUS	ALASKA	92	END OF DEC	GEESE - INVENTORY
4	2007	CENSUS	ARIZONA	858	END OF DEC	GEESE - INVENTORY

How it works: a user selects a dataset for exploration and visualization then selects a data property. The more data the user selects the larger the table gets and grows out of proportion.

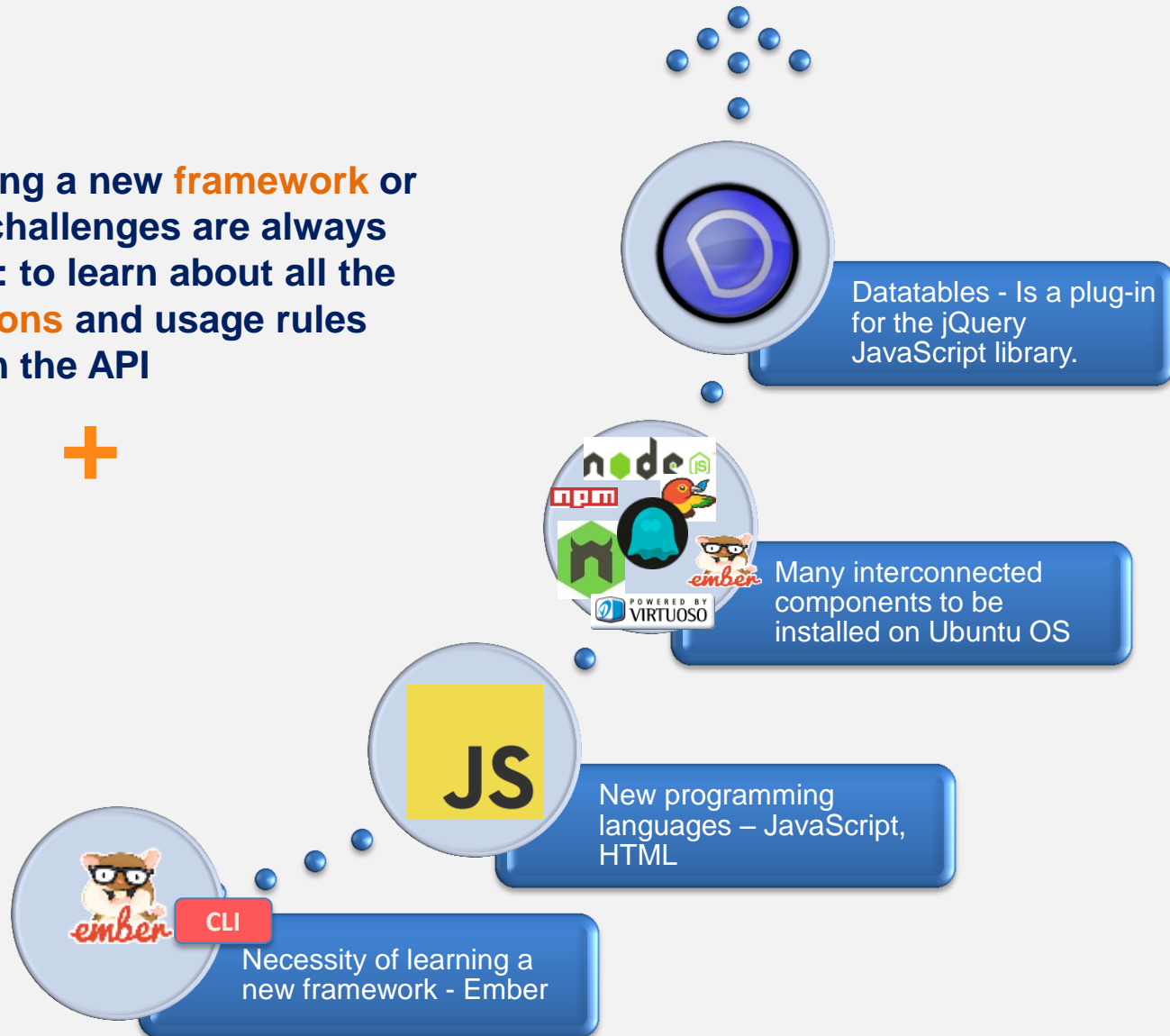
Test cases

No	Component	Expected Result	Test Result (passed/failed/blocked)
1	Searching	Possibility of seeking keywords through the tree (searching for a particular keyword inside the tree, added exceptions for non-found words and too short words);	Passed
2	Filtering	Filtering numeric data in the table (with min and max values, once the filters are chosen, the data table changes automatically);	Passed
3	Scrolling	Installed plug-in for the data table with ability to scroll large data sets	Passed

Challenges:

While using a new **framework** or **API**, the challenges are always the same: to learn about all the **expectations** and usage rules implicit in the API

+





Result after implementation:

The screenshot displays the LinDaViz web application interface. On the left, the 'Explore and Select Data' panel shows a tree view of data fields: DataEntry (class), frequency (string), value (number), location (string), source (string), type (resource), state_fips (number), state (string), data_item (string), and year (number). A search bar with the text 'dataentry' and a 'Search' button is highlighted with an orange circle. An orange callout box labeled 'Key-based search' points to this search bar. Below the field list, an orange callout box labeled 'Filters' points to the 'Filters' section at the bottom right. This section shows a dropdown menu with 'value' selected, and input fields for 'Min: 300' and 'Max: 800', which are also circled in orange. On the right, the 'Preview Data Selection' panel shows a table of data entries. The table has columns: frequency, value, location, source, state_fips, state, and data_item. The first three rows are visible, showing entries for 'END OF DEC' with values 371, 722, and 556, located in 'STATE' (CENSUS) for 'HAWAII', 'MARYLAND', and 'VERMONT' respectively. An orange circle highlights the table area, with an orange callout box labeled 'Scrolling' pointing to it. The table footer indicates 'Showing 1 to 3 of 3 entries (filtered from 50 total entries)' and includes 'Previous' and 'Next' navigation links. A 'Visualize' button is located at the bottom right of the interface.

frequency	value	location	source	state_fips	state	data_item
END OF DEC	371	STATE	CENSUS	15	HAWAII	GEESE - INVENTOR'
END OF DEC	722	STATE	CENSUS	24	MARYLAND	GEESE - INVENTOR'
END OF DEC	556	STATE	CENSUS	50	VERMONT	GEESE - INVENTOR'



+

Thank you!