DUY TÂN UNIVERSITY
International School

CAPSTONE PROJECT 1

CMU-SE-450 / CMU-IS-450 / CMU-CS-450

ARCHITECTURE DOCUMENT

Version 1.2

Date: 12 - Aug - 2020

SMART DASHBOARD APPLICATION

Submitted by

Vo Van Hoa
Pham Van Tin
Ky Huu Dong
Tran Thanh Kieu

Approved by

Capstone Project 1 - Mentor:

Name	Signature	Date
Binh, Thanh Nguyen _____		_____14 - Dec- 2020

Name	Signature	Date
Huy, Truong Dinh _____		

PROJECT INFORMATION			
Project Acronym	SDA		
Project Title	Smart Dashboard Application		
Project Web URL	https://sda-research.ml/		
Start Date	12 - Aug - 2020		
End Date:	15 - Dec - 2020		
Lead Institution	International School, Duy Tan University		
Project Mentor	PhD Binh, Nguyen Thanh; MSc Huy, Truong Dinh		
Scrum Master	Hoa, Vo	hoavo.dng@gmail.com	0935.193.182
Team Members	Tin, Pham Van	tinphamvan123@gmail.com	0932.535.175
	Dong, Ky Huu	kyhuudong@gmail.com	0898.246.980
	Kieu, Tran Thanh	thanhkieutran391@gmail.com	0358.583.251

DOCUMENT INFORMATION			
Document Title	Architecture Document		
Author(s)	Team C1SE.06		
Role	[SDA] Architecture_v1.2		
Date	13 - Dec - 2020	File name	[SDA] Architecture_v1.2
URL	https://github.com/sdateamdtu2020/sda-documents		
Access	Project and CMU Program		

REVISION HISTORY

Version	Person(s)	Date	Description	Approval
Draft	Hoa, Vo	12 - Aug - 2020	Initiate document	x
1.0	All members	20 - Sep - 2020	Finish content of document	x
1.1	All members	15 - Nov - 2020	Update content	x
1.1.1	Tin, Pham	16 - Nov - 2020	Add C&C, Module View Client	x
1.1.2	Hoa, Vo	16 - Nov - 2020	Add Module View Server, Low Level Architecture	x
1.1.3	Hoa, Vo	20 - Nov - 2020	Add Quality Attributes	x
1.2	All members	13 - Dec - 2020	Update C&C, Module View diagrams	X

TABLE OF CONTENTS

1. INTRODUCTION

1.1. PURPOSE

The purpose of the Architecture document is to:

- Define the architecture needs and technology in detail.
- Provide solutions for business needs.
- Provide overview about resources, schedule, solution and budget for the project.

The architecture merely introduces the project to the student development teams, and provides the up-front information necessary for the team to develop a specification.

1.2. DEFINITIONS, ACRONYMS AND ABBREVIATIONS

Acronyms	Definitions
SDA	Smart Dashboard Application
GUI	Graphical User Interface
SDK	Software Development Kit
ES	ECMAScript language

1.3. DOCUMENTS REFERENCES

No.	Reference
1	Product Backlog Document for SDA
2	Project Plan Document for SDA

2. PROBLEM STATEMENT

2.1. PROJECT OVERVIEW

Our environment is always changing. However, at the current rate of urbanization and industrialization, outside of the natural factors, the change of environment is mainly due to human factors. Emissions, population explosion, industrial solid waste, ... are the main causes leading to negative effects on the global environment. To address this at a holistic level, data analysis and aggregation are the first important tasks to be done.

However, analyzing and aggregating data from many different sources takes a lot of effort and money. To solve this problem, based on our knowledge of big data systems, we have built an intelligent data processing system that can be run on a website-platform with an intuitive and easy-to-use dashboard. This system is a prospective and useful tool for environmental experts and policy makers in Vietnam in particular, and worldwide in general. It will collect, analyze and synthesize data about all the factors that can affect the environment, thereby helping users to come up with quick and accurate solutions to solve problems that related to the environment.

2.2. BUSINESS DRIVERS

Business problem:

Our environment is always changing. However, at the current rate of urbanization and industrialization, outside of the natural factors, the change of environment is mainly due to human factors. Emissions, population explosion, industrial solid waste, ... are the main causes leading to negative effects on the global environment. To address this at a holistic level, data analysis and aggregation are the first important tasks to be done.

Business need:

Smart Dashboard Application have specific uses :

- Help users to come up with quick and accurate solutions to solve problems that related to the environment.
- Help users represent data in many types of charts, diagrams and maps, thereby providing an overview to solve the problem.
- Help users can get the information and insights they need at a glance.
- Giving business predictions based on the chart, diagrams, and maps.

All the things above are based on the functionality of the Smart Dashboard Application system. SDA fully meets these requirements. Therefore, the development of SDA is very necessary and meaningful.

2.3. PROJECT GOAL

Smart Dashboard Application - SDA - means that a dashboard that is convenient for users to analyze and review data. It will include several hundred datasets about real-time information of the environment, historical aerial photographs, measurement data of air pollutants... SDA will connect and analyze data from multiple sources in ways you've never imagined. Then reveal the insights you've been missing...in just a matter of minutes.

With smart suggestions and an intuitive visual interface, SDA makes it easy for any user to combine data and discover hidden insights in one place...without the usual scripting, coding, and IT hand-holding. With this dashboard, individuals or any subjects can take advantage of environmental data to be able to decide the best relevant policies.

3. ARCHITECTURE DRIVERS

3.1. HIGH-LEVEL REQUIREMENTS

(Refer to the Product Backlog document for SDB)

3.2. SYSTEM CONTEXT

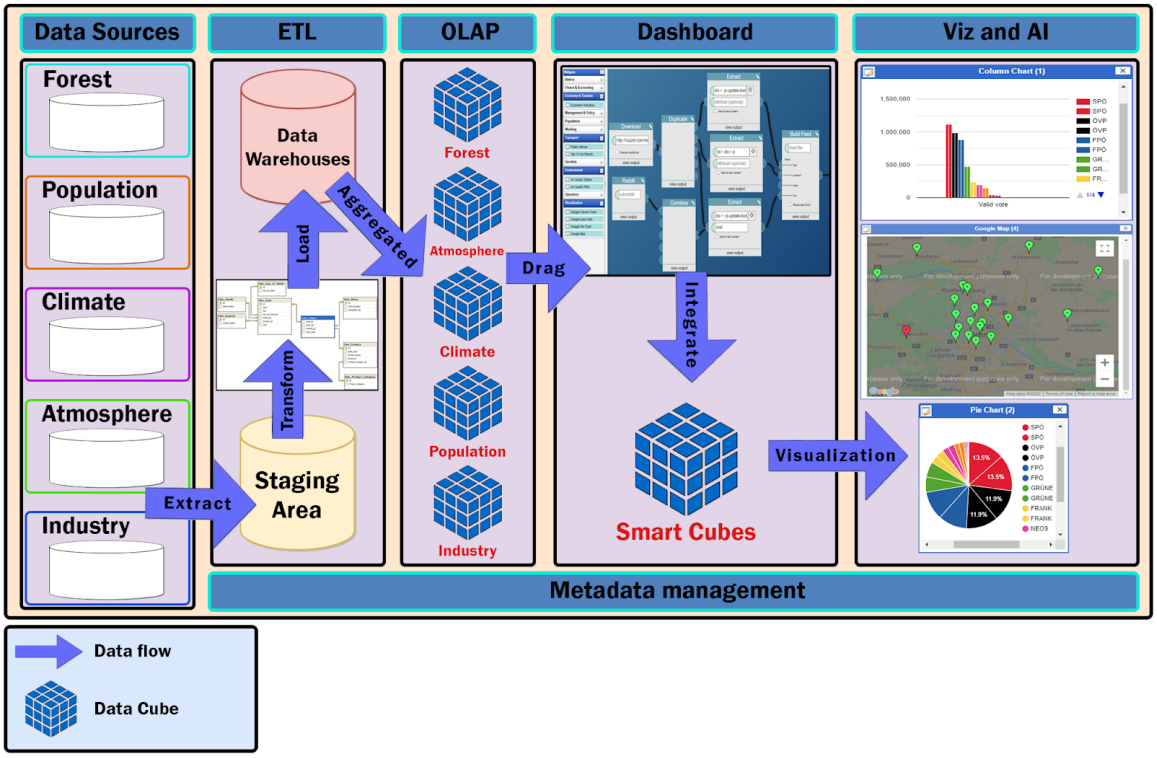


Figure 3.2: Context Diagram of System

Prose:

- **Data Sources :**
  - Collect from environment open data platforms of the governments and NGO organizations.

    - Use web crawling techniques to crawl data from related environment websites.
    - Data Format : CSV, JSON, XML.
- **ETL (Extract, Transform, Load)**
    - **Extract :**
        - In this step, data is extracted from the source system into the staging area. Transformations if any are done in the staging area so that performance of source system is not degraded. Also, if corrupted data is copied directly from the source into the data warehouse database, rollback will be a challenge. Staging area gives an opportunity to validate extracted data before it moves into the Data warehouse.
    - **Transform :**
        - Data extracted from the source server is the raw data and not usable in its original form. Therefore it needs to be cleansed, mapped and transformed. In fact, this is the key step where the ETL process adds value and changes data such that insightful reports can be generated. In the transformation step, we filter , clean, split and integrate the data to match with the system requirements and data warehouse architecture.
    - **Load :**
        - Loading data into the target data warehouse database is the last step of the ETL process.
- **OLAP**
    - In this step, we export the data from the data warehouse as RDF data cubes for better query performance, data binding and scalability, in addition for information transparency.
- **Dashboard**
    - In this step, the user can drag any data cubes that appear as items on the sidebar and drop onto the main content board, then connect between them and use the operator such as statistics merge, geo merge to build a new data cube that matches the user requirement.
- **Viz & AI**
    - This step will perform the data cube which was created by the user with the form they want. It can be a map, a column chart, a line chart or a pie chart.

3.3. QUALITY ATTRIBUTES

ID	QA01
Quality Attributes	Performance
Stimulus	Uses the statistics merge operator for merging multiple data cubes.
Source(s) of stimulus	User
Artifacts	System
Environment	Normal mode
System response	The SDA returns a new data cubes as fast as possible
Response measure(s)	Within 3 seconds

Table 3.3.1: Quality Attributes: Performance

ID	QA02
Quality Attributes	Availability
Stimulus	The power is off while server is running
Source(s) of stimulus	Power
Artifact	During peak usage load
Environment	Hardware and software
System response	System will use the cloud server to save the work so we don't need to worries about power incident occurred
Response measure(s)	All work always can be saved

Table 3.3.2: Quality Attributes: Availability

ID	QA03
Quality Attributes	Availability
Stimulus	Can't get a specific data cubes when select it from widget
Source(s) of stimulus	User
Artifact	System
Environment	Normal mode
System response	System will log the fault immediately
Response measure(s)	Within immediately

Table 3.3.2: Quality Attributes: Availability

4. CONSTRAINTS

4.1. BUSINESS CONSTRAINTS

- Project will be started on: 12 - Aug - 2020
- Project will be finished on: 15 - Dec - 2020
- Duration: 17 weeks

4.2. TECHNICAL CONSTRAINTS

Main Programming Language: Javascripts, Python.

Data Warehouses:

- Programming Language: Python.
- Database: PostgreSQL.
- Library: Psycopg2, CSV, Unidecode.

Data Cubes

- Programming Language: RDF-Graph, SPARQL.
- Tool for converting from Datawarehouse to RDF Data Cube : OpenRefine.
- Storing Platform & Endpoint : GraphDB.
- Network Accessing: RDF-REST API.

Server:

- Programming Language: Javascripts.
- Framework: ExpressJS (NodeJS).

- Libraries: Node-Postgres.
- Operating System: Windows, Linux, MacOS.
- Deployment Environment: Google Cloud with App Engine and SQL Services.
- Network Accessing: HTTP methods (POST, GET) via RESTful API.

Client:

- Programming language: HTML, CSS, Javascripts.
- Framework: React, Redux.
- Libraries: Material-UI, react-dnd, beautiful-react-diagrams, highcharts
- Deployment Environment: Google Firebase Hosting
- Operating System: Windows, Linux, Mac OS.
- Web Browser: Chrome, Firefox, Microsoft Edge, Coccoc
- Network Accessing: World Wide Web (WWW), HTTP methods (POST, GET) via RESTful API

5. HIGH-LEVEL ARCHITECTURE

5.1. COMPONENT AND CONNECTOR VIEW (C&C VIEW)

The diagram below shows the overview architecture including components and other related components. We have representations and behaviors for import components in the following sections

References C&C View on attached page.

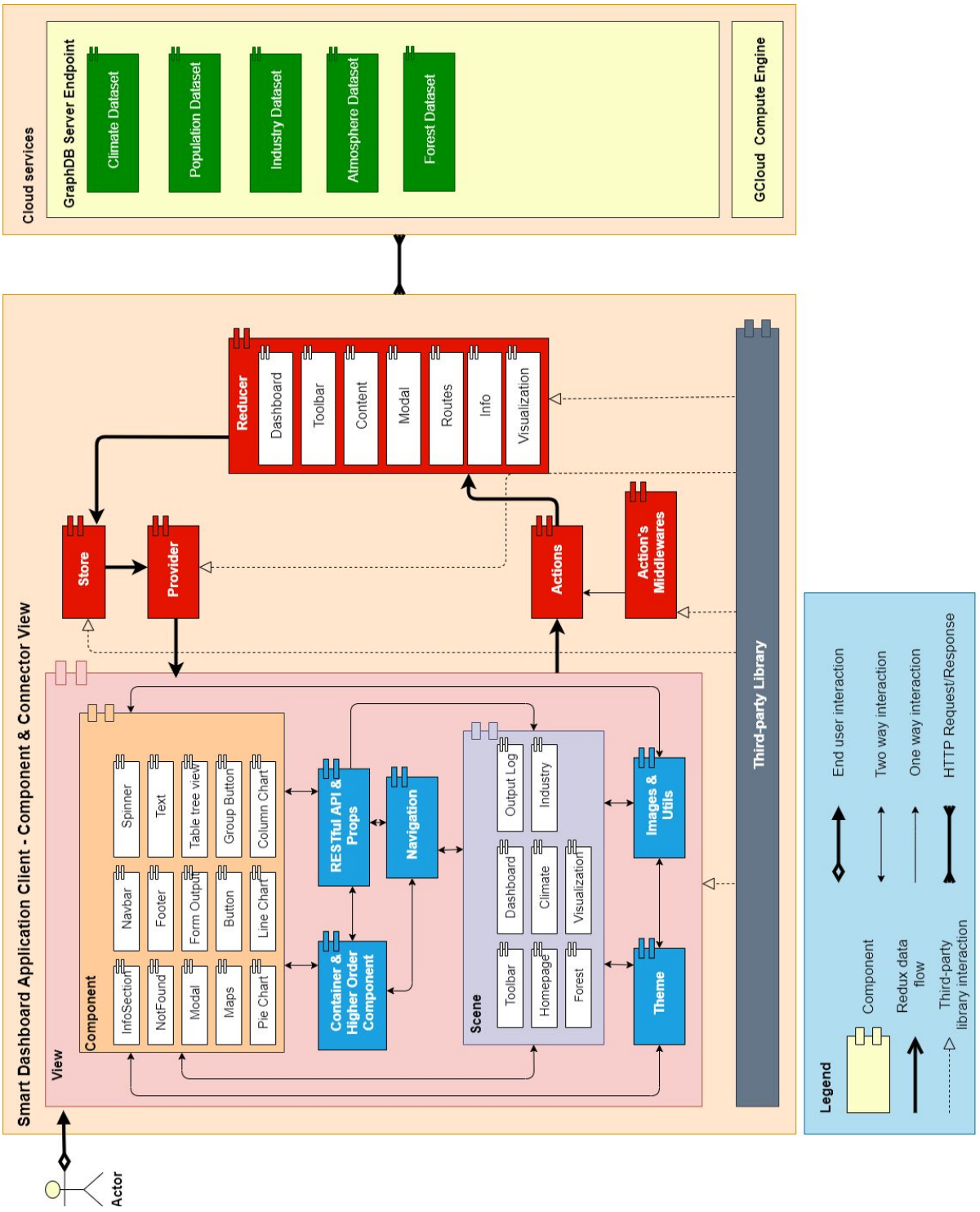


Figure 5.1: C&C View

Prose:

When the end-user opens the application, GUI will appear. Users will interact with the View component. The View component includes some small components like Toolbar, Content, Description and Util, Scene, Navigation, Container and Higher Order component, and some other components (The View component will be detailed in Model View). If User has any action, View will call the Actions component. Actions will format the requirement and send it to the Reducer. The Reducer will send a state tree to the Store. The Store will replace the old state tree with a new one. The Provider is a component that helps connect Store and View, it will get the new state from Store and send it back to View. Third party libraries will help the application run faster and reduce the code time.

Any time when the application needs to use data, it will call Cloud service. Cloud service includes Graph server endpoint. We place our server on GCloud and implement the application API on it.

Role & Responsibility	Description
View	
Component	Components let you split the app into independent, reusable pieces, and think about each piece in isolation.
Scene	The Scene transform represents the app in UI.
Container & Higher Order Component	Pattern that has proven to be very valuable for several React libraries
Restful API & Props	Provides a data in your API & render in child
Navigation	Provides an easy to use navigation solution,
Theme	Defining set of styles.
Images & Utils	Storing images or utils library.
Store	The Store is the object that brings all together.
Provider	Make the store available to all container components in the application without passing it explicitly.
Actions	Actions are payloads of information that send data from your application to your store.
Actions's Middlewares	It provides a third-party extension point between dispatching an action, and the moment it reaches the reducer.
Reducer	Actions describe the fact that something happened, but

	don't specify how the application's state changes in response. This is the job of reducers.
Third-party Library	
GraphDB Server Endpoint	Build and ship our APIs faster and more consistently. Not having to worry about authentication, performance and status monitoring has reduced the time and effort we need to build great APIs
API:	Set of functions and procedures that allow the creation of applications which access the features or data of an operating system, application, or other service.

5.2. MODULE VIEW

5.2.1. Module View based on Web Application

References Module View on attached page.

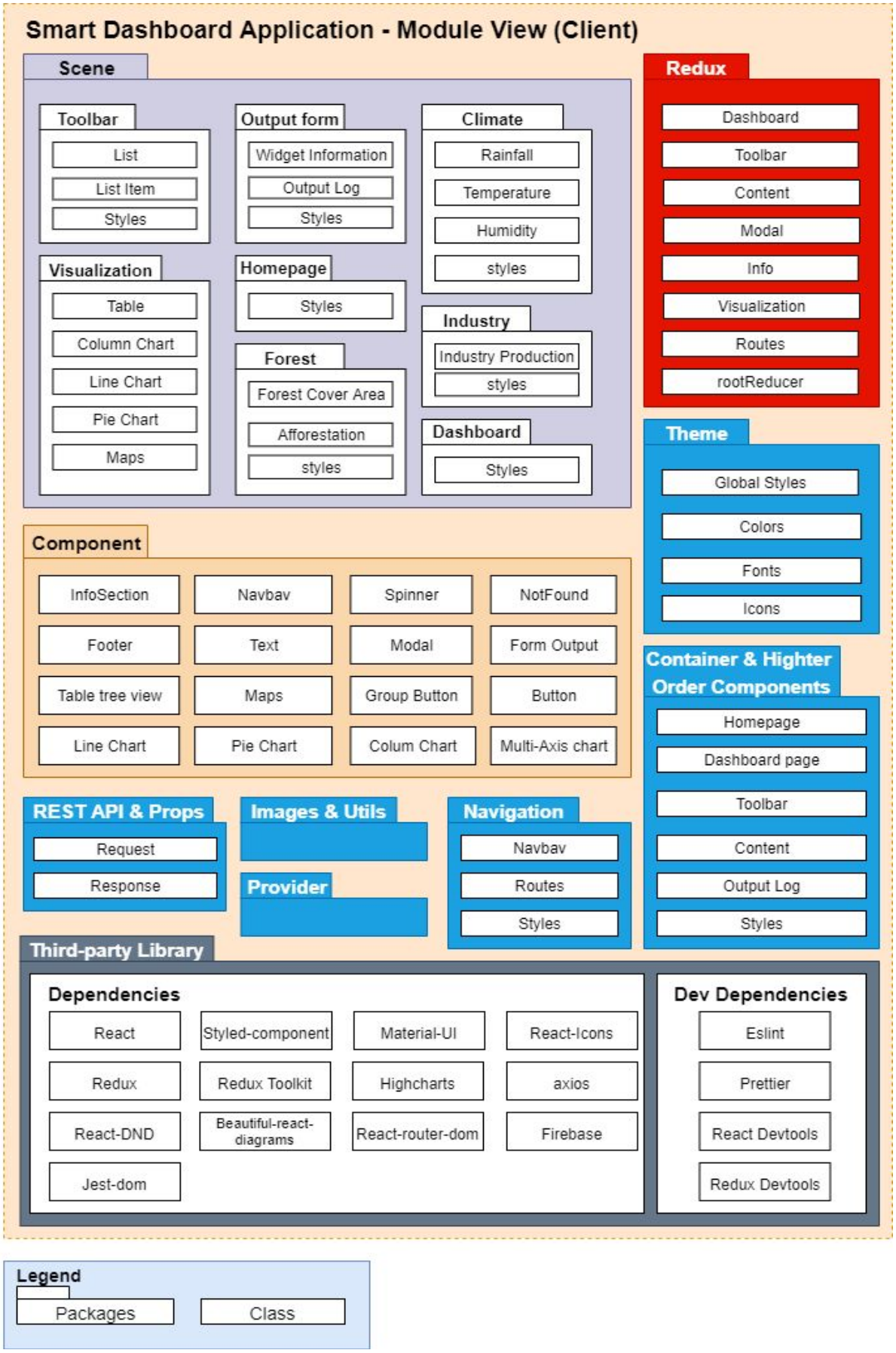


Figure 5.2.1: Module View Client

Prose:

The SDA client application includes 10 packages that help the app run effectively.

The Scene package includes

Component package which has 16 classes which is often used and we custom it to fit our requirement. We also have Theme package to define the app format. It will define the app color, font, icon, and metric.

Navigation package contains a Navbar, Routes and its Style.

Redux package also has Dashboard, Toolbar, Content, Modal and root Reducer class to manage the state. Our Component and Higher Order Components is an advanced technique in React that is used in reusing components. Higher Order Components are not part of the React API. Specifically, a higher-order component is a function and it takes the argument as a component and returns a new component.

Images and Utils package contain the app image and functions to solve app general problems.

In third-party libraries, we have Dependencies and Devdependencies class.

Beside it, We have Dev Dependencies class like EsLint, Prettier, React Dev Tool and Redux Dev Tool to manage the code and make sure that our code follows the general format.

Finally, when the app wants to use data, it will connect to cloud service. Cloud service includes GraphDB and GCloud service

Role & Responsibility	Description
Styles	Defining styles
Scene	
Toolbar	Display list of datacube & dataset
Output form	Widget Log
User Guide	Help user can use app easily
Homepage	Home page of web
Dashboard	Higher Order Component that wrap toolbar, mainboard, Information Widget
Climate	DataCube
Industry	DataCube
Forest	DataCube
Component	

InfoSection	Section Information
Navbar	Navigation
Spinner	An UI when page is loading
NotFound	An UI when Page when 404 error
Footer	Footer
Form Input	A text field is used for form
Text	A general text is used for all text in this application
Modal	Modal Component is inherited Modal of Material-UI
Button	Button
Group Button	List of button
Custom Node	Widget of each datacube & visualization
Line Chart	Visualization data from data cube in chart
Column Chart	Visualization data from data cube in chart
Pie Chart	Visualization data from data cube in chart
Maps	Visualization data from data cube in maps
Redux	
Dashboard	A Redux store & reducer for handling state of Dashboard
Toolbar	A Redux store & reducer for handling state of Toolbar
Content	A Redux store & reducer for handling state of Content
Modal	A Redux store & reducer for handling state of Modal
Routes	Define routes of scenes for navigation
rootReducer	A combination of all defined reducer & third party reducer
Theme	
Global Styles	Defining styles
Container & Higher Order Components	
Homepage	Responsible for handling homepage, about, contact routes
Dashboard page	Responsible for handling dashboard routes &
Toolbar	Handle list item & drag & drop from Toolbar to Content
Content	Handle widget node & connector in Mainboard

Outputlog	Handle widget information & data log
Rest API & Props	
Request	Used to send request to get data from server
Response	Used to get data when requesting success, such as humidity, temperature,....
Images & Utils	
Provider	
Navigation	
Navbar	A List of buttons such as: New, Help, Example1,...
Routes	Define routes of scenes for navigation
Third-party Library	
Dependencies	A list of dependencies that is required for operating the application
DevDependencies	A list of dependencies that is required for development environment

5.2.2. Module View based on Server

References Module View - Server on attached page.

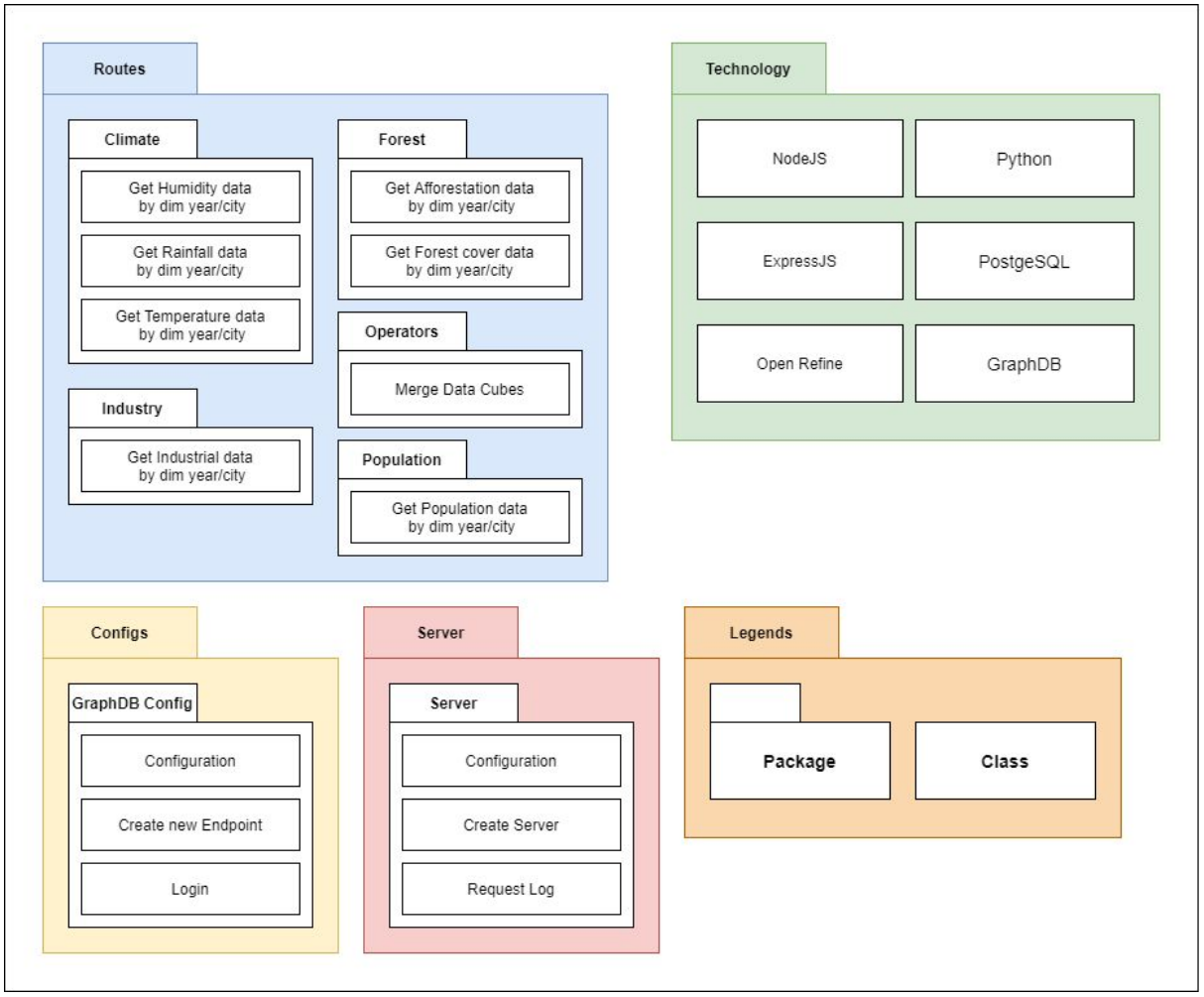


Figure 5.2.2: Module View - based on Server

Prose:

Role & Responsibility	Description
Routes	GET HTTP actions
Climate	Get data of climate data cube (humidity, rainfall, temperature values) by dimension year and city
Forest	Get data of forest data cube (afforestation area, forest cover value) by dimension year and city
Industry	Get data of industry data cube by dimension year and city
Population	Get data of population data cube by dimension year and city
Operators	Execute merging 2 or 3 data cubes for creating a new data cube

Configs	
GraphDB Config	Initiate configuration values for GraphDB, create a new SPARQL Endpoint, login to GraphDB
Server	
Server	Initiate configuration values for creating an ExpressJS server, log the requests from clients
Technology	
ExpressJS	Library for programming API
NodeJS	Library for programming API
Open Refine	A reconciliation service for registered SPARQL endpoints, a graphical user interface(GUI) for exporting data of Google Refine projects in RDF format. The export is based on mapping the data to a template graph using the GUI.
Python	Programming Language for building ETL process

5.3. ALLOCATION VIEW

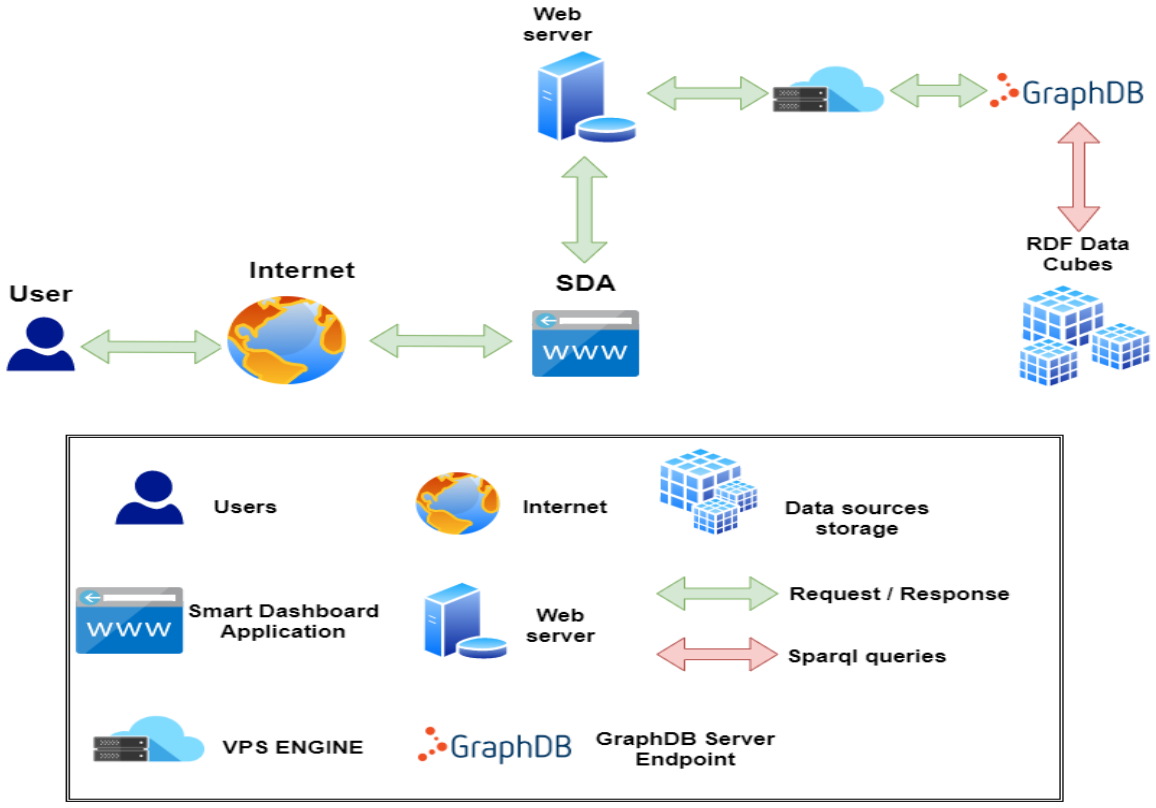


Figure 5.3: Allocation View

Prose:

The user can access our web app via the internet. When they use it, the web app will connect to the GCloud server to get the data via the internet.

Role & Responsibility	Description
Data sources storage	RDF Data Cubes is stored on GraphDB
User	User that interact with SDA
Web server	HTTP Server with NodeJS
Smart Dashboard Application	Our application.
VPS Engine	Store and Execute SDA's server
GraphDB Server Endpoint	Stage to received and run SPARQL for interacting with the data cubes
Request / Response	Get request metadata from client and response the data to client
SPARQL queries	Query language for execute the method of RDF Data Cubes

6. LOW-LEVEL ARCHITECTURE

6.1. Database Design

6.1.1. Data warehouse architecture

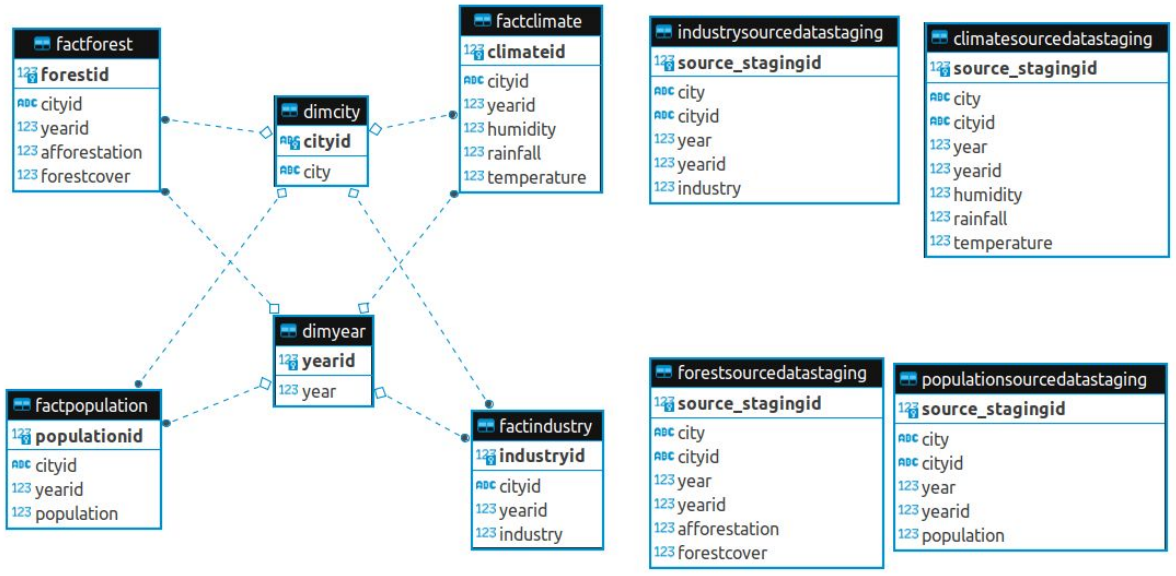


Figure 6.1.1: Data warehouse Schema

A data warehouse is subject oriented as it offers information regarding a theme instead of companies' ongoing operations. These subjects can be sales, marketing, distributions, etc.

A data warehouse never focuses on the ongoing operations. Instead, it put emphasis on modeling and analysis of data for decision making. It also provides a simple and concise view around the specific subject by excluding data which is not helpful to support the decision process.

6.1.2. Star schema, Fact Tables and Dimension Tables

The star schema architecture is the simplest data warehouse schema. It is called a star schema because the diagram resembles a star, with points radiating from a center. The center of the star consists of a fact table and the points of the star are the dimension tables.

A fact table typically has two types of columns: foreign keys to dimension tables and measures those that contain numeric facts. A fact table can contain fact's data on detail or aggregated level.

A dimension is a structure usually composed of one or more hierarchies that categorizes data. If a dimension hasn't got hierarchies and levels it is called flat dimension or list. The primary keys of each of the dimension tables are part of the

composite primary key of the fact table. Dimensional attributes help to describe the dimensional value. They are normally descriptive, textual values. Dimension tables are generally smaller in size than fact tables.

Typical fact tables store data about sales while dimension tables data about geographic region (markets,cities), clients, products, times, channels.

SDA database system is designed based on the data warehouse platform. In SDA data warehouse, there are three main concepts (table types), those are staging area table, dimensional table and fact table with star schema:

- Staging area tables: Climate Staging area, Forest Staging area, Industry Staging area, Population Staging area.
- Dimensional tables: dimyear table, dimcity table.
- Fact tables: factclimate table, factforest table, factindustry table, factpopulation table.

6.2. RDF DATA CUBES

6.2.1. RDF DATA CUBES DESIGN

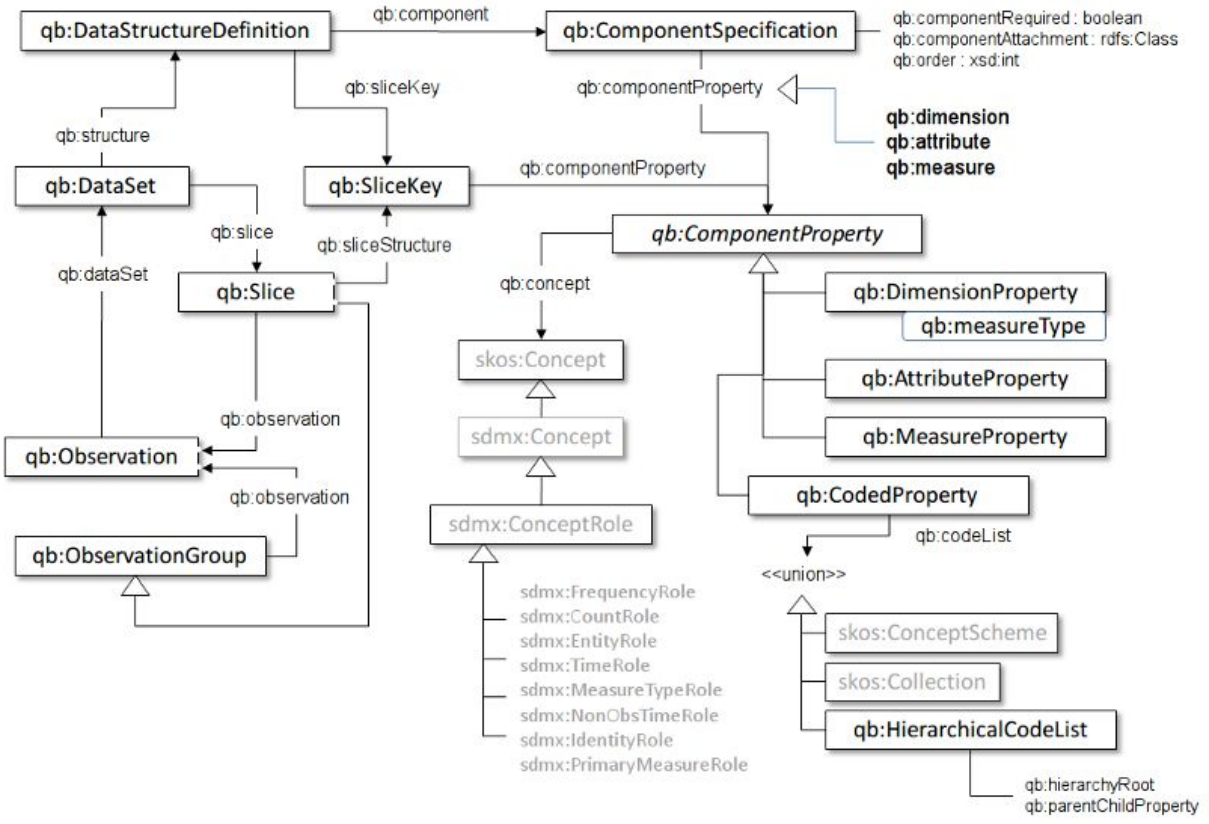


Figure 6.2: Pictorial summary of key terms and their relationship.

6.2.2. RDF Data cubes Architecture

The vocabulary of the RDF Data Cube was recently proposed as a W3C recommendation for the destruction of various mathematical data using the RDF format. The model under this vocabulary is based on the SDMX model, the ISO standard for representing statistical data that can be shared between organizations. . The data cube, called the dataset, is an example of class qb:Dataset. Cube data cells, called observation, are the conditions of phase qb:Observation. Each look can have one or more attributes, sizes, and dimensions of a structure. Features defining qb type steps: MeasureProperty. This positive relationship correlates with recognition and measurement values (e.g., value of humidity), which have a range of numbers. Qualities and sizes of species qb:AttributeProperty and qb:DimensionProperty, respectively. The size represents the viewing context (e.g., the city in which the humidity is calculated), while the attributes provide additional information about the measure, such as the unit of measure.

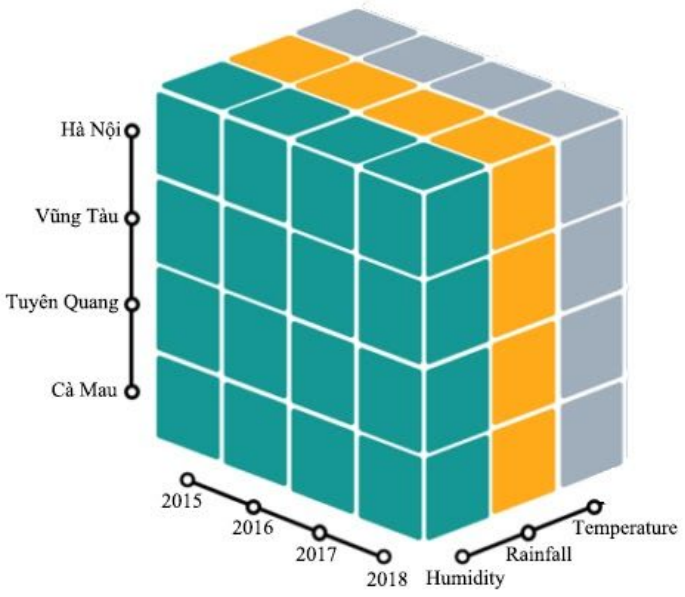


Figure 6.2.1: Climate Data Cube in 3D

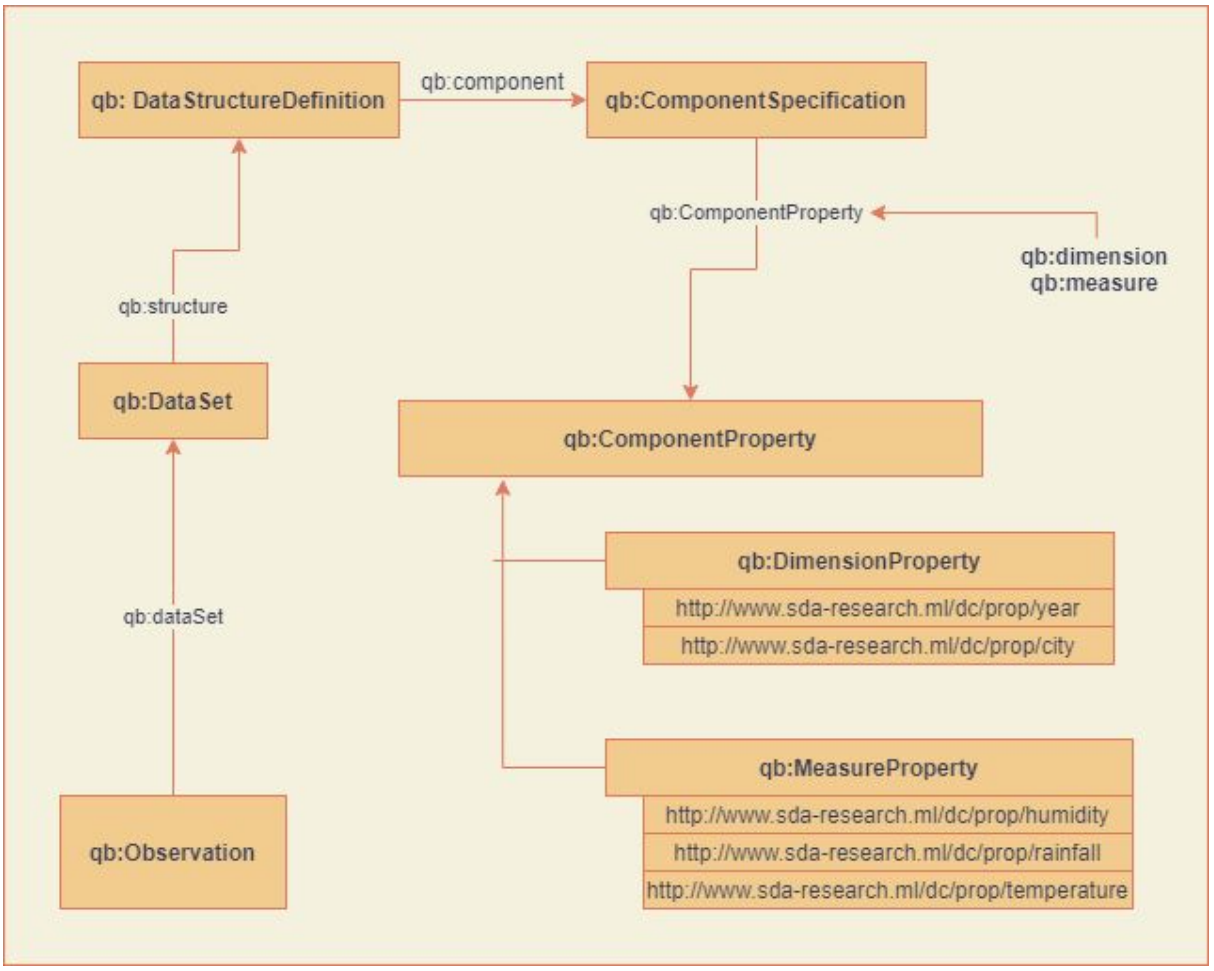


Figure 6.2.2: Climate Data Cube Structure

As the schema above, the Data Structure Definition has a child node ComponentSpecification, where qb:ComponentProperty is configured. The qb:ComponentProperty includes two main concepts, qb:DimensionProperty and qb:MeasureProperty - the DSD definitions for each dimension and measure in the data warehouse. Each class has an specific URI that is regulated for querying with SPARQL Endpoint and makes the Linked Data Structure for the Data Cube.

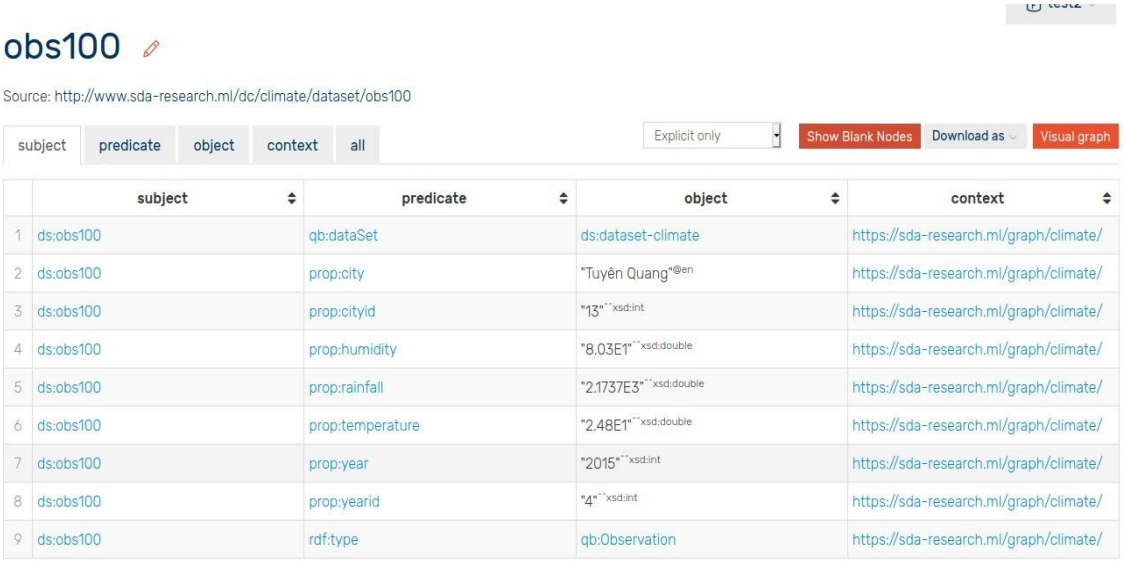
For instance, The Climate Data Cube structure has:

- Dimensions:
  - City : Hà Nội, Vũng Tàu, Tuyên Quang, Cà Mau,etc...
  - Year : 2015, 2016, 2017, 2018,etc...
- Measures:
  - Humidity
  - Rainfall
  - Temperature
- Observations:

- Humidity value of Hà Nội in 2015, 2016.
- Rainfall value of Cà Mau in 2015, 2016.
- Temperature of Tuyên Quang in 2017, 2018.
- ...etc...

When a user need to figure out a set of statistical data that depends on multiple categories, such as humidity of an area/city in a specific time period (year), the data cube will refer to the statistics value (observation) of the measure (humidity) that match the dimension (city, year).

For instance, this is the observation of the rainfall value of Tuyen Quang in 2015.



obs100

Source: http://www.sda-research.ml/dc/climate/dataset/obs100

subject | predicate | object | context | all

Explicit only | Show Blank Nodes | Download as | Visual graph

	subject	predicate	object	context
1	ds:obs100	qb:dataSet	ds:dataset-climate	https://sda-research.ml/graph/climate/
2	ds:obs100	prop:city	"Tuyên Quang"@en	https://sda-research.ml/graph/climate/
3	ds:obs100	prop:cityid	"13"^^xsd:int	https://sda-research.ml/graph/climate/
4	ds:obs100	prop:humidity	"8.03E1"^^xsd:double	https://sda-research.ml/graph/climate/
5	ds:obs100	prop:rainfall	"2.1737E3"^^xsd:double	https://sda-research.ml/graph/climate/
6	ds:obs100	prop:temperature	"2.48E1"^^xsd:double	https://sda-research.ml/graph/climate/
7	ds:obs100	prop:year	"2015"^^xsd:int	https://sda-research.ml/graph/climate/
8	ds:obs100	prop:yearid	"4"^^xsd:int	https://sda-research.ml/graph/climate/
9	ds:obs100	rdf:type	qb:Observation	https://sda-research.ml/graph/climate/

Figure 6.2.3: Observation 100th.

6.3. APPLICATION APIS

For interacting between client and server, we choose to deploy SDA onto a Linux Virtual Private Server. Using Linux VPS, it is very easy and simple to develop, maintain and especially run the server in the most stable way.

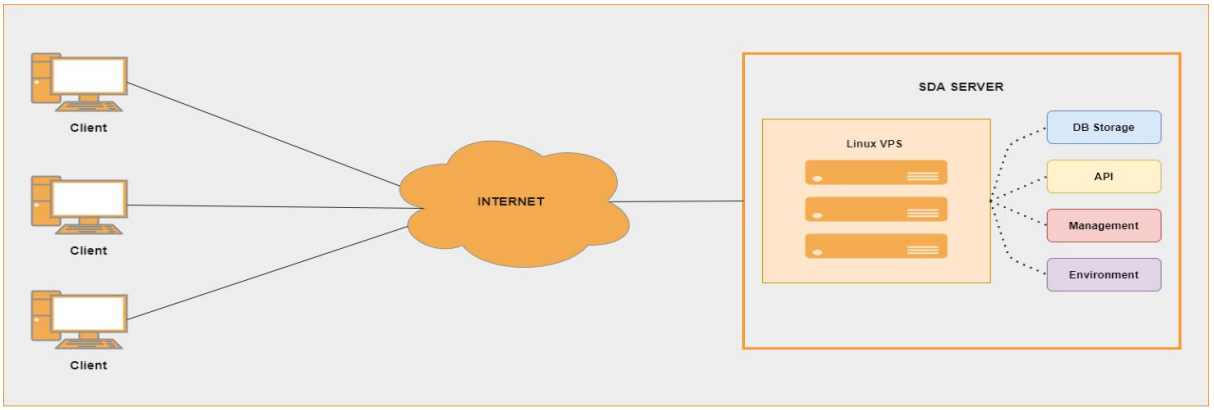


Figure 6.2.4: Deployment Diagram

We write the APIs of SDA system with enapso-graphdb library, a Javascript library that give developers the most flexible way to configure, connect and interact with OntoText GraphDB platform, and the help of Express.JS, the most famous also strongest NodeJS library for building REST API server.

In the SDA system, because that is a data representing and integrating system, thus the main HTTP method we use in API development is the GET method. Specifically, with each data cube, we design separate GET methods for querying the data cube based on the requirements of clients. Below is the routes map of the SDA system.

```
GET /climate/humidity/city/:cityid
GET /climate/humidity/year/:year
GET /climate/humidity/city/:cityid/year/:year
GET /climate/humidity/city/:cityid/fyear/:fyear/tyear/:tyear
GET /climate/rainfall/city/:cityid
GET /climate/rainfall/year/:year
GET /climate/rainfall/city/:cityid/year/:year
GET /climate/rainfall/city/:cityid/fyear/:fyear/tyear/:tyear
GET /climate/temperature/city/:cityid
GET /climate/temperature/year/:year
GET /climate/temperature/city/:cityid/year/:year
GET /climate/temperature/city/:cityid/fyear/:fyear/tyear/:tyear
GET /industry/industry/city/:cityid
GET /industry/industry/year/:year
GET /industry/industry/city/:cityid/year/:year
GET /industry/industry/city/:cityid/fyear/:fyear/tyear/:tyear
GET /forest/afforestation/city/:cityid
GET /forest/afforestation/year/:year
GET /forest/afforestation/city/:cityid/year/:year
GET /forest/afforestation/city/:cityid/fyear/:fyear/tyear/:tyear
GET /forest/forestcover/city/:cityid
GET /forest/forestcover/year/:year
GET /forest/forestcover/city/:cityid/year/:year
GET /forest/forestcover/city/:cityid/fyear/:fyear/tyear/:tyear
GET /population/population/city/:cityid
GET /population/population/year/:year
GET /population/population/city/:cityid/year/:year
GET /population/population/city/:cityid/fyear/:fyear/tyear/:tyear
GET /merge/dc1/:dc1/dc2/:dc2/s1/:s1/s2/:s2/city/:cityid/fyear/:fyear/tyear/:tyear
GET /merge/dc1/:dc1/dc2/:dc2/dc3/:dc3/s1/:s1/s2/:s2/s3/:s3/city/:cityid/fyear/:fyear/tyear/:tyear
```

7. REFERENCES

- Server Endpoint: http://server.sda-research.ml
- Draw.io: https://www.draw.io
- ETL (Extract, Transform, and Load) Process: https://www.guru99.com/etl-extract-load-process.html
- Extract, transform, load: https://en.wikipedia.org/wiki/Extract,_transform,_load
- What is Extract, Transform, Load? Definition, Process, and Tools: https://www.talend.com/resources/what-is-etl/
- ETL - Extract, Transform, Load: https://www.webopedia.com/TERM/E/ETL.html

- What is OLAP? Cube, Operations & Types in Data Warehouse: https://www.guru99.com/online-analytical-processing.html
- What is OLAP: https://www.ibm.com/cloud/learn/olap
- OLAP cube: https://en.wikipedia.org/wiki/OLAP_cube
- OLAP (Online analytical Processing): https://techterms.com/definition/olap
- The RDF Data Cube Vocabulary: https://www.w3.org/TR/vocab-data-cube/
- Optimizing RDF Data Cubes: http://ceur-ws.org/Vol-1426/paper-02.pdf
- CubeQA—Question Answering on RDF Data Cubes: https://www.researchgate.net/publication/313073658_CubeQA-Question_Answering_on_RDF_Data_Cubes
- RDF Data Cube - graphical representation : https://www.researchgate.net/figure/RDF-Data-Cube-graphical-representation_fig3_265690180
- Linked clinical data cube architecture aligned with the RDF data cube: https://www.researchgate.net/figure/Linked-clinical-data-cube-architecture-aligned-with-the-RDF-data-cube-Depicts-the_fig2_274700091
- The RDF Data Cube: https://www.w3.org/TR/eo-qb/#Datacube
- Data Warehouse Architecture: Types, Components, & Concepts: https://www.astera.com/type/blog/data-warehouse-architecture/
- Data Warehousing - Architecture: https://www.tutorialspoint.com/dwh/dwh_architecture.htm
- Data Warehouse Architecture: Traditional vs. Cloud: https://panoply.io/data-warehouse-guide/data-warehouse-architecture-traditional-vs-cloud/