Information Systems and Databases

1^{st} **Semester 2020/2021**

Project - Part 3

Instituto Superior Técnico

Mestrado em Engenharia Electrotécnica e de Computadores

Monday, 15h30, Lab 14

December 18, 2020

Group 15:

Francisco Rodrigues, nº 86993, 12 horas, 33%

Isabel Castelo, nº 87023, 12 horas, 33%

Ricardo Antão, nº 87107, 12 horas, 33%

Professor:

Prof. João Marques

Web Application Architecture

The web application developed can be accessed through the link `https://web2.tecnico.ulisboa.pt/ist187107/`. A sketch of the architecture is available in figure 1. All the code created is explained in section 2 and is available in the zip file in the *web* folder together with this report.

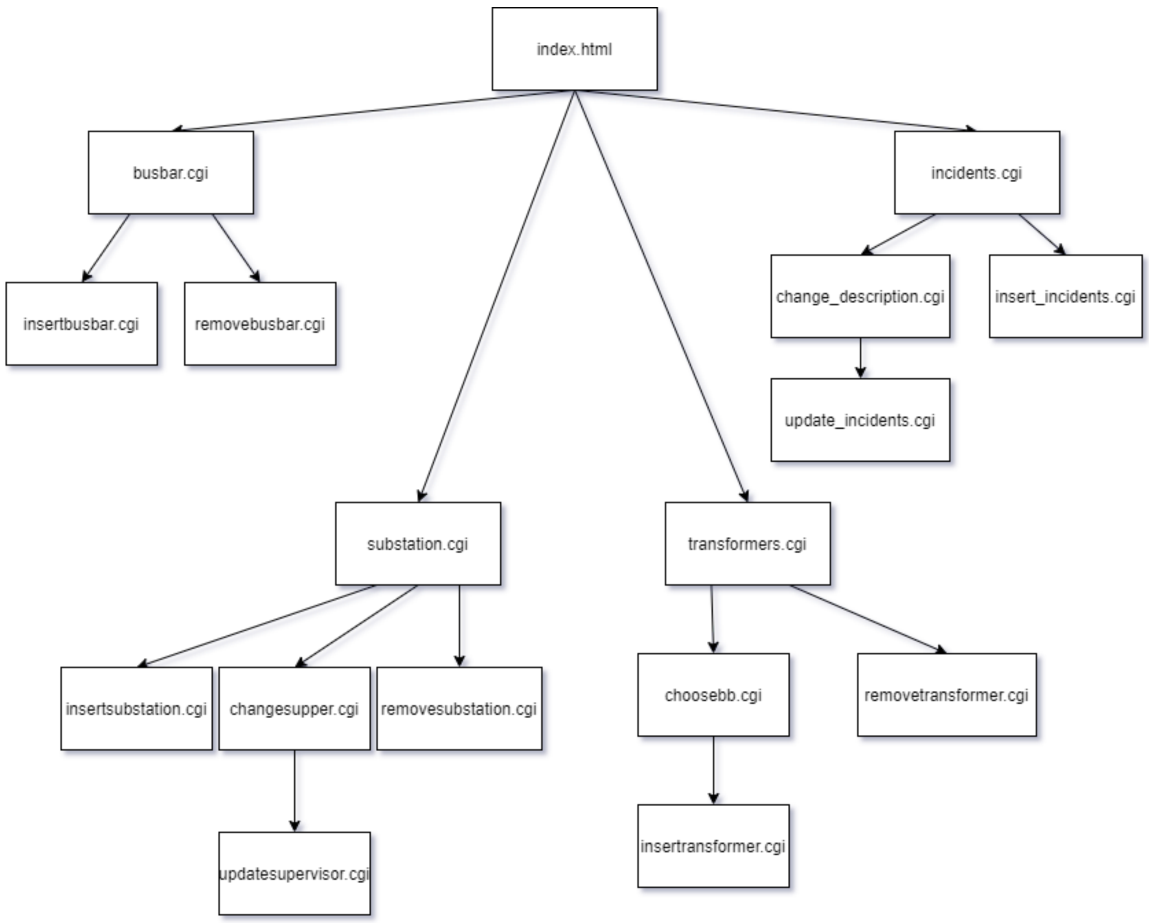


Figure 1: Simplified Web application scheme

File Relation

Index.html

Index.html creates the main menu of our web application, shown in Figure 2, where a user can choose one out of four options:

- Bus Bars - to insert, list and remove bus bars;
- Substations - to insert, list, remove and change the supervisor assigned to substations;
- Transformers - to insert, list and remove transformers;
- Incident Registering - to add non-line incidents and edit their description.

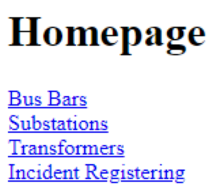


Figure 2: Main Menu

busbar.cgi - Busbars section

By choosing the option Bus bars in the main menu, the application goes to the page generated by busbar.cgi file. This file creates a new page where the user can insert a new element id and a voltage in order to create a new bus bar; see a list of all existing bus bars; or delete any busbar already created, as Figure 3 shows.

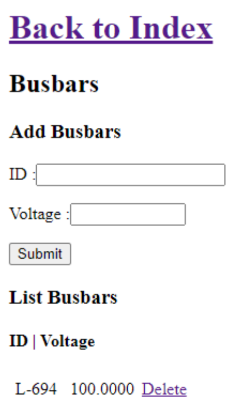
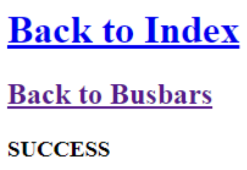


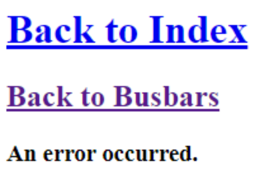
Figure 3: Insert and Delete of Bus Bars

Clicking the submit button with the all the fields filled will call the file **insertbusbar.cgi**. This file will execute the insertion in the database. To guarantee the atomicity of the operations the insertbusbar.cgi will execute an insert of the ID in the *element* table and the insert of the new busbar in the *busbar* table sequentially. If the 2 operations are successful, a success message will appear, as in figure 4a. Otherwise, an error message appears if the element id inserted already exists, as in 4b. Clicking the delete button in a specific bus bar will call the **removebusbar.cgi** file which will remove the bus bar from the database and show a success message like Figure 4a, unless the delete operation violates a foreign key constraint, in that case an error message will appear, like Figure 4b. Similar to **insertbusbar.cgi** the **removebusbar.cgi** will execute the delete of the busbar from the *busbar* table and the delete of the busbar's ID from the *element* sequentially.



SUCCESS

(a) Success message



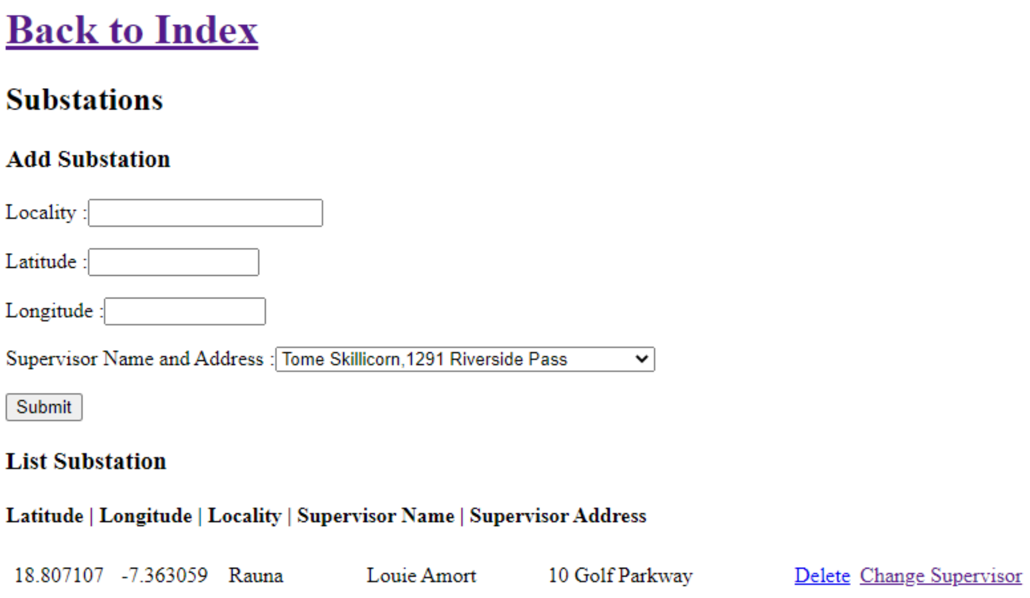
An error occurred.

(b) Error message

Figure 4: Outputs possible in the insertion of a new bus bar

substation.cgi - Substations section

By choosing the option Substations in the main menu, the application goes to the page generated by the **substations.cgi** file. This file creates a new page where the user can insert a locality, latitude, longitude and supervisor in order to create a new substation; see a list of all existing substations; delete already created ones; and can change the supervisor assigned, as Figure 5 shows.


Back to Index

Substations

Add Substation

Locality :

Latitude :

Longitude :

Supervisor Name and Address : Tome Skillicorn,1291 Riverside Pass

Submit

List Substation

Latitude | Longitude | Locality | Supervisor Name | Supervisor Address

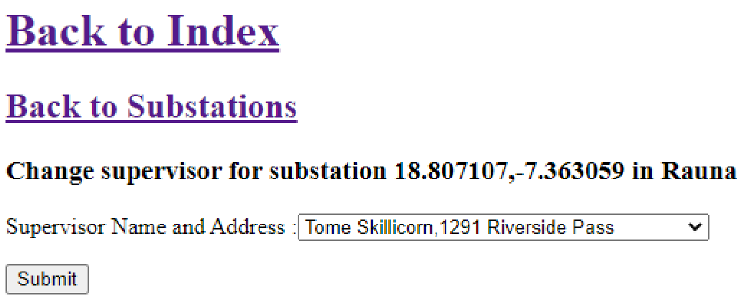
18.807107	-7.363059	Rauna	Louie Amort	10 Golf Parkway	Delete Change Supervisor

Figure 5: Insert, Update and Delete of Substations

The functioning of the application in Substations is the same as mentioned before in BusBars. The submit will call the **insertsubstation.cgi** file which will give a success or an error message identical to the ones showed in Figure 4. This time, only an insert in the *substation* table will be executed. To prevent wrong inputs by the user, longitude and latitude were restricted to their respective ranges (-90 to 90 and -180 to 180) and to 6 decimal places (to match with the schema datatype). Additionally the Supervisor is chosen from a drop down menu (which has all the entries from *supervisor* table) to prevent foreign key constraint violations. Clicking the delete button in a specific substation will call the **removesubstation.cgi** file, which will remove the substation in the database and show a success message as well. However, the substations option has another feature which consists in changing the supervisor assigned. By choosing this feature (clicking in the **Change Supervisor** link), the file **changesupper.cgi** is called.

changesupper.cgi - Choosing the new Supervisor

The **changesupper.cgi** file redirects to another page, visible in Figure 6, where the user can choose the new supervisor assigned to the substation picked. When the user picks the new supervisor, the file **updatesupervisor.cgi** is called, which will update the substation in the database. The **updatesupervisor.cgi** will execute the update command with the parameters from the html form. It is important to note that, by clicking on the **Change Supervisor** link, we have made a parameterized call of the **changesupper.cgi** file, where the parameters are the attributes from the substation which we want to alter the supervisor. These parameters are passed to the **updatesupervisor.cgi** as hidden inputs of the html form.


Back to Index

Back to Substations

Change supervisor for substation 18.807107,-7.363059 in Rauna

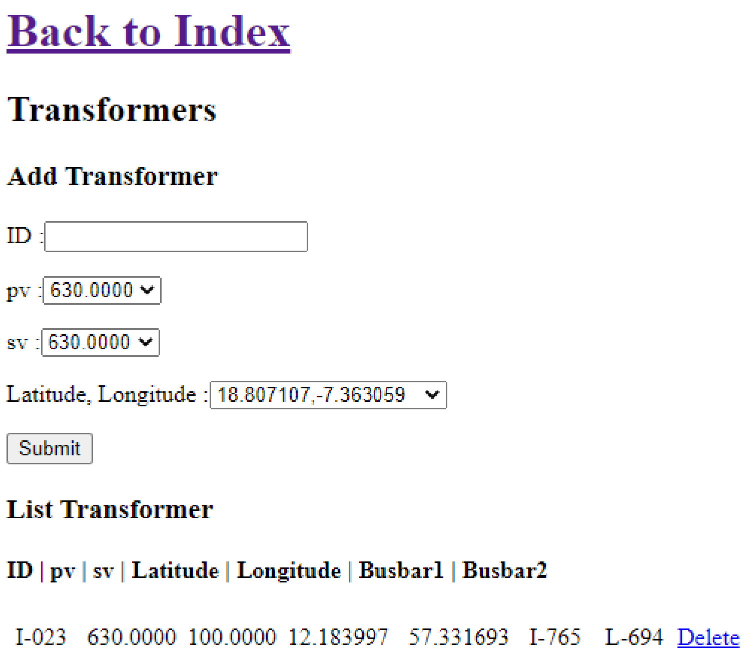
Supervisor Name and Address : Tome Skillicorn,1291 Riverside Pass

Submit

Figure 6: Update Supervisor of a substation from available

transformers.cgi - Transformers section

By picking the option Transformers in the main menu, the application calls the **transformers.cgi** file. This file creates a new page where the user can insert an element id, primary voltage, secondary voltage, latitude and longitude in order to create a new transformer; see a list of all existing transformers; and delete already created ones, as Figure 10 shows.


Back to Index

Transformers

Add Transformer

ID :

pv : 630.0000

sv : 630.0000

Latitude, Longitude : 18.807107,-7.363059

Submit

List Transformer

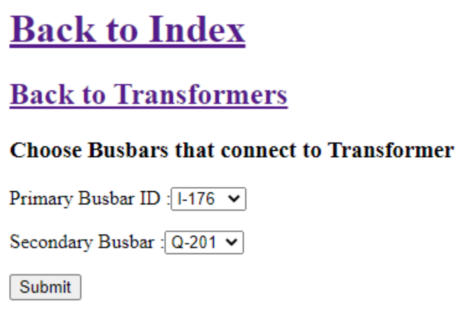
ID	pv	sv	Latitude	Longitude	Busbar1	Busbar2	
I-023	630.0000	100.0000	12.183997	57.331693	I-765	L-694	Delete

Figure 7: Insert and Delete of Transformers

Choosing to add a new transformer to the database by clicking the submit button with all the fields filled, will call the file **choosebb.cgi**. Just like the other menus described, the delete feature will call the **removetransformer.cgi** which will remove the picked transformer of the database. Like the **removebusbar.cgi** the **removetransformer.cgi** will remove both entries from the *transformer* and *element* tables sequentially.

choosebb.cgi - Choosing the connecting busbars

Submitting the html form to add a new transformer will redirect to a new page which is shown in Figure 8. This new page will ask the user to pick the busbars that will connect to the transformer, where the primary voltage of the transformer has to be the same as the voltage of the primary bus bar, and the secondary voltage of the transformer has to be the same as the one from the secondary bus bar. This is done so that the user does not violate the integrity constraint imposed by a trigger. After picking the busbars, the file **insertransformer.cgi** will be called and will insert in the database the new transformer. This insertion will result in a success or error message as the ones described before. Analogously to the **removetransformer.cgi**, the atomicity of operations is guaranteed in a similar fashion to the busbar processes.


Back to Index

Back to Transformers

Choose Busbars that connect to Transformer

Primary Busbar ID : I-176

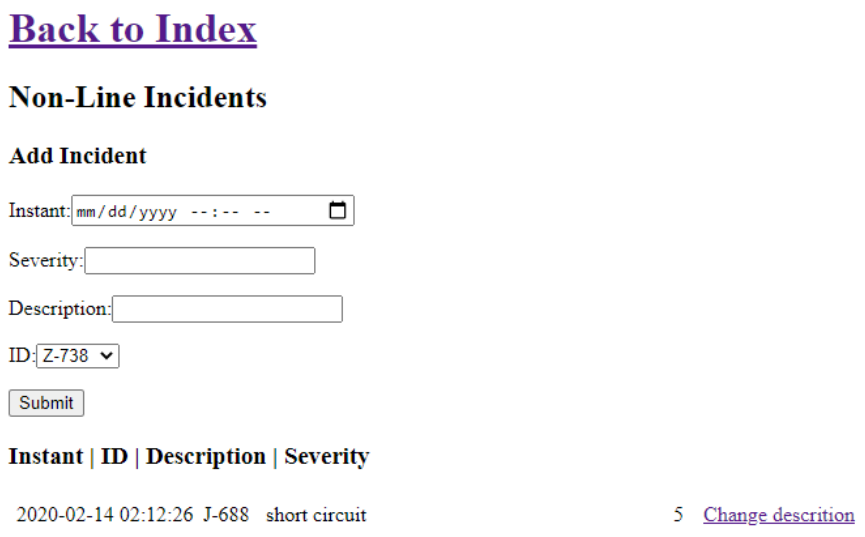
Secondary Busbar : Q-201

Submit

Figure 8: Insert Busbars connected to Transformer from available

incidents.cgi - Incidents section

By picking the option Incident Registering in the main menu, the application calls the **incidents.cgi** file. This file creates a new page where the user can insert a datetime, severity, description and element id in order to register a new incident; see a list of all the incidents related to non-line elements; and change their description. The html will give the user freedom to insert any instant, severity and description but the element id chosen will come either from the *busbar* or *transformer* table.


Back to Index

Non-Line Incidents

Add Incident

Instant: mm/dd/yyyy --:-- --

Severity:

Description:

ID: Z-738

Submit

Instant | ID | Description | Severity

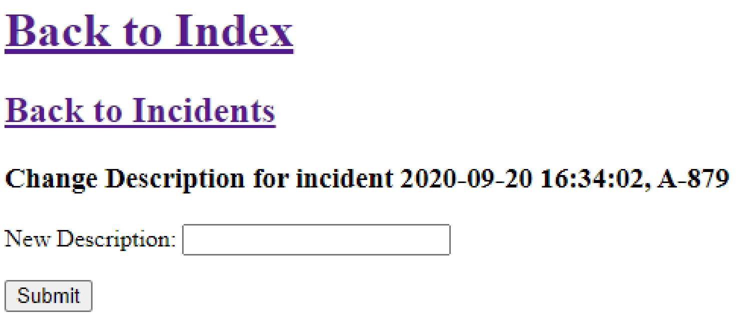
2020-02-14 02:12:26 J-688 short circuit 5 Change descrition

Figure 9: Add and Update Non-Line Incidents

By filling the form and submitting it **the insert_incidents.cgi** file will be called. This file will insert a new incident on the *incident* table. The response in case of success or failure of the operation will be identical to the ones in Figure 4. Additionally clicking on the **Change description** link will call the **change_description.cgi** file.

change_description.cgi - Elaborating the new description

The page generated by the **change_description.cgi** file is composed by an html form prompting the user to write a new description. Analogously to the **changesupper.cgi** file the **change_description.cgi** file is called with a parameterized call. This parameters are the attributes of the incident which the user wants to alter the description.


Back to Index

Back to Incidents

Change Description for incident 2020-09-20 16:34:02, A-879

New Description:

Figure 10: Update Incident's Description

After writing a new description and submitting the form the **update_incidents.cgi** file will be called. This file will execute the update command altering the description of the corresponding incident. Again the success or failure messages will appear like in Figure 4.

Indexes

6.1

Query to be optimized:

```
SELECT locality, COUNT(*)
FROM transformer NATURAL JOIN substation
WHERE pv = <some value>
GROUP BY locality
```

For the first query, we decided to use 2 different indexes:

```
CREATE INDEX ON substation(locality);
CREATE INDEX ON transformer USING HASH(pv);
```

The first index is a B+ tree index on the table *substation* on the *locality* attribute. The purpose of this query is to speed up the *GROUP BY* operation. This is a secondary, dense and unclustered index. A B+ tree type of index will help the *GROUP BY* operation thanks to the index entries being ordered by the search key which will be the attribute to be aggregated by the GROUP BY.

The second index will be a hash index on the table *transformer* on the attribute *pv.* This index is used to help with the equality constraint on the *pv* attribute. This is a secondary and dense index. The hash index is great to speed up equality searches since you just need to check one of the buckets of the hash index (the one corresponding to the value in the equality constraint).

The conjunction of these 2 indexes will speed up the aggregation operation and the equality search. However the hash index on *pv* won't be of much use if the equality condition qualifies high percentage of records (high selectivity).

6.2

Query to be optimized:

```
SELECT id, description
FROM incident
WHERE instant BETWEEN <t1> AND <t2>
AND description LIKE '<SOME_PATTERN>%'
```

For the second query, we decided to use 1 index.

```
CREATE INDEX ON incident(instant, description);
```

The index is a B+ tree index on the table *incident* with a composite search key over the attributes *(instant, description).* The purpose of this query is to speed up the range searches on both attributes. Normally a B+ tree index is not used to help the **LIKE** operator search, but in this case, since the constraint is on the prefix of the description, this type of index is able to help.

Alternatively a index could be created leading to an index-only scan.

```
CREATE INDEX ON incident(instant, description, id);
```

This index would be a B+ tree on the table *incident* over the *instant*, *description* and *id.*The objective of this query would be to create an index only plan. The problem is that the cost of maintaining such an index is very high. This is a primary clustered index. Additionally after some testing (using **EXPLAIN**),it was verified that the cost of the index-only plan (provided by Postgres) was identical to the cost of the plan using the first index.

Notes

There are some things to note in the application:

- Every page generated has a link back to the index.html and (if possible) a link back to the section in which the user is found (transformer.cgi, incidents.cgi, busbar.cgi or substation.cgi)
- To not alter the file schemaPart3.sql we are working with the ranges of latitude and longitude switched around.

- To keep the database from having impossible locations, a check constraint was added to the *substation* table so that the coordinates are always in the correct ranges (although they are still switched).
- In order to not ignore the incidents with lines and busbars a new (fictional) entry was added to the *d_location* table. This entry will be the one associated to the incidents with busbars and lines in the *f_incident* table.