Costa Rica Institute of Technology

Academical Area of Computer Engineering

I Semester 2020

Codename: **Hospital TECnológico (Pandemica Second Wave)**

Technical Documentation

Schlafenhase Development Team:

- **Jose D. Acuña** - *Web Functionality Manager & Design Assistant* - JoDaniel1412
- **Kevin Cordero** - *Lead Developer on Back-end and Connections* - Skryfall
- **Alejandro Ibarra** - *Lead Designer & Mobile App Developer* Assistant Project Manager- AlejandroIbarraC
- **Jose D. Sánchez** – *Database Administration & Definition* - JoseDavidSS
- **Jesús Yamir Sandoval** - *Project Manager & Web Developer* - shuzz22260427

Technical Advisor:

- **Marco Rivera** - *Teacher*

Course:

- **Databases** - CE 3101

Languages used:

- C#
- JavaScript / TypeScript
- MEAN stack tools (Node, Angular, Express)

A. Description of the data structure used (tables).

SQL Tables / Web Tables: Tables are database objects that contain all the data in a database. The information is logically organized in rows and columns. Each row represents a unique record, and each column represents a field in the record.

For the project in question, tables were used as the main structure to organize the server information, which provides said information, previously stored in the tables, directly to the web page.

Lists: These are used in the API and web apps to parse incoming data from the database. The main function of this data structure was to transport the information (which was organized in tables) to the front end. These types of lists follow the standard array concepts in C-like languages, which are well known in the programming scope.

B. Description of implemented Triggers/Stored Procedures/Functions

1. Stored Procedure (Used by the front end)

- *Usp_patient_procedure:* This SP function is to select all the procedures for the patient given ordered by date and returns it.
- *Usp_insert_reservation:* This SP function is to insert a value in reservation_procedure value.
- *Usp_insert_bed_equipment:* This SP function is to insert an equipment into a bed.
- *Usp_equipment_for_bed:* This SP function is to get the equipment information for the given bed.
- *Usp_update_equipment_from_bed:* This SP function is to update the information for a bed. Basically, takes the old equipment information and replaces it with newer information.
- *Usp_insert_bed_equipment:* This SP function is to delete any relationship between a bed and an equipment.
- *Usp_makes_reservation:* This SP function is to makes a reservation for a certain patient. It also must verify the hospital information in order to check if the reservation date is available.

- *Usp_update_reservation:* This SP function is to update the data for an old reservation. Basically, updating the data for an old reservation with newer dates for the same patient.

2. Triggers (Back End)

- *Hospital_delete:* This trigger was created in order to prevent a hospital from being deleted by accident.
- *Synchronization_trigger:* This is the most important trigger in the project. Its main function is to synchronize the database every 6 hours if there's an INSERT, UPDATE or DELETE on the Microsoft SQL Server Dataabse, linking the data of hospitals, patients and hospital staff. It should be noted that the reason for its creation was, once again, to keep the data as synchronized and updated as possible between the Microsoft SQL Server and PostgreSQL.

3. Functions (Front End)

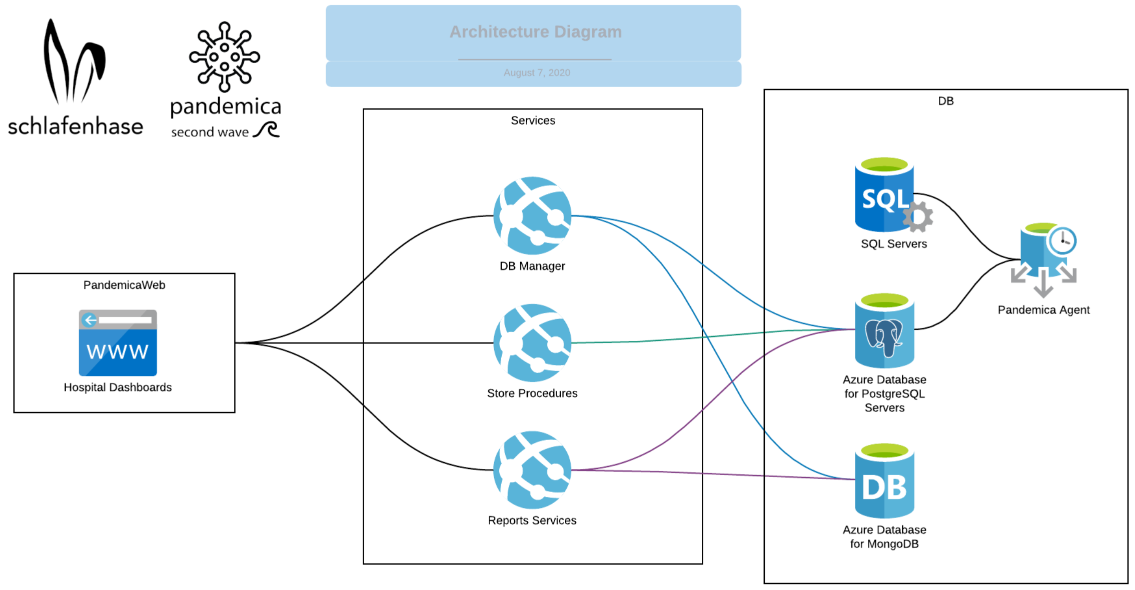
- *selectChange*: This function was created with the purpose of obtaining the information about the options chosen in the mat-select HTML element and in the Angular date picker. It works with an observed variable that notifies the TypeScript logic file of the actions that were carried out in the HTML, as the function encapsulates and stores the values in an observable data type to use as information for the server.
- *pop-up:* This function is responsible for opening local pop-up windows when the user clicks on add or edit. When doing this, an observable variable of the Boolean type is activated which indicates to the pop-up window that it should be opened or closed, depending on what is needed by the user in addition to the type of pop-up, so it can adjust the corresponding form correctly.
- *Update:* A method mainly used for the creation of the forms. Its function is to obtain the selected data from the list shown in the data form, this in order to then send only the data of the selected option to the server
- *Warning methods:* This is a set of methods implemented using the "SweetAlert2" library which is a direct dependency of JavaScript.
  The main function of these methods is to generate small tables with different warnings when the client makes the different options that they

have on the page. Its implementation is fully automatic thanks to the aforementioned dependency. To use this library, only a method was created in the TypeScript file and it was called from there

C. Architecture and Database Diagrams

Due to the differences in dimensions and structure of each of these diagrams, they are in the sub-folder called "Database Diagrams". These include the conceptual and relational database diagrams.

The architecture diagram and description can be found below:



Architecture description: Architecture is basically an interconnection of bases in synchrony. Mainly the main base is the one that stores the data, which in turn is supported by Azure and MongoDB, which are DBaaS in a cloud. A total synchronicity of all the data in the storage is always sought. The front-end and the back end must always maintain constant and uninterrupted communication, this in order to avoid data loss and maximize the updating of information on the website.

Basically, the website communicates with the various APIs. DB Manager handles the communication with various databases. Stored Procedures handles all packed information fetched via stored procedures. Finally, Reports Services handles the analysis and report generation.

D. Found problems.

1.1) SP problems with Entity Framework: When trying to call a Stored Procedure from Entity Framework and making this Stored Procedure try to return a table with columns, it was impossible to get it to be represented in a model class. For this reason an error was thrown.

1.2) Solution: To solve it, you had to go to the context of the database in Entity Framework and you had to add a special model class (kind of like a view), and indicate to the context that in order to understand that table of the Stored Procedure you have to use the class model that was made.

1.3) References:

- Core, R., & Harlow, D. (2020). [SOLVED] - Raw SQL Query without DbSet - Entity Framework Core. Retrieved 7 August 2020, from https://entityframeworkcore.com/knowledge-base/35631903/raw-sql-query-without-dbset---entity-framework-core

2.1) Problems with *NgFor: This was one of the least expected problems and was found when testing the CPU usage of the project.

The problem was that the *NgFor command (which is used to create a loop in HTML using the Angular web framework) consumed a lot of the resources on the PC (about 6GB of RAM and 187% of processor power)

2.2) Solution: It should be noted that this is not a problem caused by the group as such, but one due to the poor optimization of this tool (in this case the *NgFor by Angular and Google), so a total solution is basically impossible to achieve by part of the users unrelated to its maintenance and development. However, it is possible to mitigate if the resources that are made available when executing the problem is strictly regulated.

2.3) References:

- Improving Angular *ngFor Performance Through trackBy. (2020). Retrieved 3 July 2020, from https://medium.com/better-programming/improving-angular-ngfor-performance-through-trackby-ae4cf943b878

E. Known bugs.

Thanks to the experience gained so far through the course and the various works carried out, although some errors were found during the project, none that could not have been fully and satisfactorily resolved.

It should be noted that this was due to the great cohesion as a team and good management of resources in the different areas of the project.

F. Conclusions and Recommendations

- Conclusions.

-- Angular is one of the most powerful and useful tools for web development.

-- Aesthetics and ease of use is of utmost importance for creating a good website.

-- Diagrams are of vital importance when creating a database. An error in this process can have terrible consequences in the project's development.

-- Organization and care are extremely important when creating a database since the slightest error can cause great long-term damage to its implementation or use.

-- Triggers must be used with great care, as they can cause problems if they are not handled correctly.

-- Stored Procedures are extremely useful when performing various methods that can be reused multiple times.

-- MongoDB is a great tool with multiple services that simplify the use of databases.

-- Using DBaaS can greatly facilitate work in databases since it offers multiple tools that maximize the efficiency of certain processes.

- Recommendations.

-- Read the Angular documentation before even thinking of starting a web project.

-- Make a good conceptual and ER diagram before starting work on the database.

-- Try to implement Stored Procedures to achieve an efficient use of code.

-- Make a good task plan and a correct division of all team member's work.

-- Maintain good database administration by documenting all code blocks, tables, Stored Procedures, Triggers and more.

-- Learn how to use Entity Framework correctly

-- Read and understand the use of Azure tools before starting any procedure

-- Read and understand the use of MongoDB tools before starting any procedure

G. Blog and bibliography.

The following section aims to describe in a general way the tasks carried out by different members of the Schlafenhase main development team during the various periods of time involved in the project. If you want a more specific view of the tasks performed (time, manager, etc.), please review the attached work plan, located in "Docs/Second Wave/Technical" folder. It can also be found online on https://trello.com/b/7i1g4LhK/pandemica-second-wave-task-plan

15-07-20 to 18-07-20:

In the first days we started the investigation of DBaaS, specifically Azure and MongoDB. This was also substantially supported by the task previously carried out by the team, since it was about this topic, of creating a presentation to be shared for the whole student group.

Especially the members K.S Cordero, J.D Sánchez and J.A. Ibarra in MongoDB and J.D Acuña on the Azure side

Along with this, the members J.Y Sandoval and J.A Ibarra were in charge of the designs and initial documentation (Work plans and meeting agenda).

Lastly, in order to finish the initial preparations, the various diagrams were carried out as a group. Both the architecture diagram and the class diagram were made during a Discord call on the server "Régimen Corderista – Virtual Sessions" In this call, the 5 members of the group agreed and understood the structure of the project and their assigned tasks.

Sites used these days:

- Azure documentation. (2020). Retrieved 7 August 2020, from https://docs.microsoft.com/en-us/azure/?product=featured
- MongoDB Documentation. (2020). Retrieved 7 August 2020, from https://docs.mongodb.com
- Trello. (2020). Retrieved 7 August 2020, from https://trello.com/b/7i1g4LhK/pandemica-second-wave-task-plan

19-07-20 to 23-07-20:

The activities carried out the second week were the programming work in the database and the implementation of the front-end skeleton. Once the pertinent investigations had been carried out and with the diagrams ready and reviewed by all the members of the group, attempts to connect with Azure and MongoDB began. These tasks were assigned to the trio of members K.S Cordero J.D Sánchez and J.D Acuña.

At the other end, the members J.Y Sandoval and J.A Ibarra were the main managers of the organization of the website and the general design. The main tasks assigned were the creation of tables for information display and the creation of the necessary forms to fill the previously mentioned tables.

Being the first week of work as such (since the first was taken as organization and carrying out technical preparations), we tried to advance as quickly and organized as possible in order to leave a decent margin with respect to the defined deadlines, so if changes were to be made there would be not be a clash of dates

Sites used these days:

- Connect and query - Azure SQL Database & SQL Managed Instance. (2020). Retrieved 7 August 2020, from https://docs.microsoft.com/en-us/azure/azure-sql/database/connect-query-content-reference-guide Reading Excel files from C#. (2020). Retrieved 4 July 2020, from https://stackoverflow.com/questions/15828/reading-excel-files-from-c-sharp
- Connect to MongoDB. (2020). Retrieved 7 August 2020, from https://mongodb.github.io/node-mongodb-native/3.0/tutorials/connect/
- Team, A. (2020). Angular Material. Retrieved 7 August 2020, from https://material.angular.io/components/datepicker/overview
- HTML input type="radio". (2020). Retrieved 7 August 2020, from https://www.w3schools.com/tags/att_input_type_radio.asp

24-07-20 to 29-07-20:

The activities of this week were basically to implement previous investigations and work. Thanks to all the above, the connection with both Mongo and Azure was successfully made. With this done, we began to work with Entity Framework and the creation of Stored Procedures and Triggers necessary for the project (which was relatively simple thanks to the experience gained in previous projects)

Another important task done this week was creating the health-center dashboard view. This task was divided into two main areas, the first was the creation of the view and connectivity for the login and accounts system, this task was assigned to J.A Ibarra. The second great task, assigned to J.Y Sandoval, was the creation of the information displays and pertinent forms for each and every one of the components in said view (tables, entries for the information, buttons and any other similar)

Sites used these days:

- Problem with SP in Entity Framework 6. (2020). Retrieved 7 August 2020, from https://forums.asp.net/t/1974086.aspx?Problem+with+SP+in+Entity+Framework+6
- Entity Framework Core – validating data and catching SQL errors – The Reformed Programmer. (2020). Retrieved 7 August 2020, from https://www.thereformedprogrammer.net/entity-framework-core-validating-data-and-catching-sql-errors/
- issues, F., Snow, J., Murshed, R., & Snow, J. (2020). Firebase connection issues. Retrieved 7 August 2020, from https://stackoverflow.com/questions/54952012/firebase-connection-issues

30-07-20 to 24-08-20:

The final week was primarily devoted to creating the missing SPs and triggers, testing connections with Postgre and checking that everything was running in sync on both MongoDB and Azure. In addition to these connectivity tests, various synchronization tests were carried out to ensure that everything works on the desired line. The mentioned tasks were carried out by J.D Sánchez, J.A Acuña and K.S Cordero

In the case of the front-end, when all the necessary views and all the information displays had been completed, the user feedback system and aesthetic generalities were carried out. The feedback system was implemented using radio buttons of Angular Material. In the case of alerts, they were implemented when executing important actions which were linked to components (Delete or Edit button) For this task, the SweetAlert JavaScript library was used, which maximized the efficiency of the process incredibly.

At the end of all the necessary secondary tasks and tests, the last days of this period were used to organize the pertinent part of the documentation that had not yet been done. This documentation was done by all the members of the group, in order to enrich the writing and details with the experiences and work carried out by each part of the project.

Sites used these days:

- Miyoshi, T. (2020). Checkboxes, radio buttons, and menus. Retrieved 7 August 2020, from https://contactform7.com/checkboxes-radio-buttons-and-menus/
- SweetAlert2. (2020). Retrieved 7 August 2020, from https://sweetalert2.github.io
- sweetalert2/ngx-sweetalert2. (2020). Retrieved 7 August 2020, from https://github.com/sweetalert2/ngx-sweetalert2

F. General bibliography.

1. Miyoshi, T. (2020). Checkboxes, radio buttons, and menus. Retrieved 7 August 2020, from https://contactform7.com/checkboxes-radio-buttons-and-menus/
2. SweetAlert2. (2020). Retrieved 7 August 2020, from https://sweetalert2.github.io
3. sweetalert2/ngx-sweetalert2. (2020). Retrieved 7 August 2020, from https://github.com/sweetalert2/ngx-sweetalert2
4. Problem with SP in Entity Framework 6. (2020). Retrieved 7 August 2020, from https://forums.asp.net/t/1974086.aspx?Problem+with+SP+in+Entity+Framework+6
5. Entity Framework Core – validating data and catching SQL errors – The Reformed Programmer. (2020). Retrieved 7 August 2020, from https://www.thereformedprogrammer.net/entity-framework-core-validating-data-and-catching-sql-errors/

6. issues, F., Snow, J., Murshed, R., & Snow, J. (2020). Firebase connection issues. Retrieved 7 August 2020, from https://stackoverflow.com/questions/54952012/firebase-connection-issues
7. Connect and query - Azure SQL Database & SQL Managed Instance. (2020). Retrieved 7 August 2020, from https://docs.microsoft.com/en-us/azure/azure-sql/database/connect-query-content-reference-guide Reading Excel files from C#. (2020). Retrieved 4 July 2020, from https://stackoverflow.com/questions/15828/reading-excel-files-from-c-sharp
8. Connect to MongoDB. (2020). Retrieved 7 August 2020, from https://mongodb.github.io/node-mongodb-native/3.0/tutorials/connect/
9. Team, A. (2020). Angular Material. Retrieved 7 August 2020, from https://material.angular.io/components/datepicker/overview
10. HTML input type="radio". (2020). Retrieved 7 August 2020, from https://www.w3schools.com/tags/att_input_type_radio.asp
11. Azure documentation. (2020). Retrieved 7 August 2020, from https://docs.microsoft.com/en-us/azure/?product=featured
12. MongoDB Documentation. (2020). Retrieved 7 August 2020, from https://docs.mongodb.com
13. Trello. (2020). Retrieved 7 August 2020, from https://trello.com/b/7i1g4LhK/pandemica-second-wave-task-plan
14. Core, R., & Harlow, D. (2020). [SOLVED] - Raw SQL Query without DbSet - Entity Framework Core. Retrieved 7 August 2020, from https://entityframeworkcore.com/knowledge-base/35631903/raw-sql-query-without-dbset---entity-framework-core
15. Improving Angular *ngFor Performance Through trackBy. (2020). Retrieved 3 July 2020, from https://medium.com/better-programming/improving-angular-ngfor-performance-through-trackby-ae4cf943b878

schlafenhase

pandemica
second wave