Instituto Tecnológico de Costa Rica

Área de Ingeniería en Computadores

CE3101 – Bases de Datos

Tarea Corta #1

Profesor:
Marco Rivera

Link hacia el repositorio de Github:
https://github.com/jonDorito/Feria_Virtual_TEC

Integrantes:
Alvaro Vargas - 2018085151
Sergio Marín Zuñiga 2018171984
Daniel Umaña Monge 2018225216
Jonathan Gonzalez - 2018118366

II Semestre 2020

Implemented methods description **2**
Color code for creator: 2
Front-End: 2
Back-End: 2
Data structures used **5**
Back-End: 5
LinkedList: 5
List: 5
Front-End: 5
Known problems **6**
Front-End: 6
Back-End: 6
Found problems **7**
Front-End: 7
Back-End: 7
Work plan **8**
Work plan timeline **9**
Group Logs **10**
Conclusions and recommendations **11**
Bibliography **12**

Implemented methods description

Color code for creator:

Front-end	Back-end	Not Owned

Front-End:

Function Name	**Description**	**Parameters**	**Return**
selecterProducerType	Select a producer of a list	- username: username - type: type user: user	void()
onSelect	Change the current view		void()
onSubmitCategory			void
onSelectView			void
showView			void
login	Log an user to the application	username: username	void
onSubmit [producer-form.components]		attribute: attribute value: value	void
getProducts	Get all the products on the database		

Back-End:

Function name	Description	Parameters	Return
registerUser	Registers a user in the database	- newUser: user being register	Void
validateCredentials	Validates the credentials for a user	- username: username for the user - email: email for the user - password: password for the user	boolean whether or not the credentials are correct
checkUserAvailability	Checks whether or not a username is available	- user: user	bool whether or not it is available
checkEmailAvailability	Checks whether or not an email is available	- email: email	bool whether or not it is available
doesUsernameMatchesType	Matches username to a type of user	- username: username - type: type	bool did it match type
getTypeByUsername	Retrieves a type by username	- user: user	string of the type of user
modifyAttribute	Modifies a given attribute in a given user	- username: username - attribute: attribute - value: value	bool whether or not the change was made
saveUsers	Save current users to a json file in database	- users: Users store in memory	Void
saveProduct	Serialize the product data and save it to DB	- products: Linked list that contains all	Void

		the products	
retrieveUsers	Retrieve the users form DB	None	All the users loaded in DB
retrieveProducts	Retrieve the products form DB	None	All the products loaded in DB
generateToken	Generates a token for a given user	- username: username	token
doesUserHasToken	Checks whether or not the user already being assigned a token	- username: username	bool whether or not there is a valid token
validateToken	validates if the token is valid	- token: token	Whether or not token is valid
GetHash	Generates a hash in bytes for the source string	- inputString: source string	Array of bytes
GetHashString	Generates a hash string for the source string	- inputString: source string	hash string
registerCategory	Registers a category in the database	- newProduct	None
saveProduct	Serialize the product data and save it to DB	Linked list that contains all the products	none
saveOrder	Serialize the order data and save it to DB	Linked list that contains all the or	None
retrieveUsers	Retrieve the users form DB	None	All the users loaded in DB
retrieveProducts	Retrieve the products form DB	None	All the products loaded in DB
convertStringToList	Convert string to list	string in JSON format	List of products

registeredProducts	Registers a product in the database	newProduct	None

Data structures used

Back-End:

LinkedList:

LinkedLists act as easily expandable storage for the Users and Products stored in the data. They create little overhead for adding new items to the list and are dynamically managed by the memory manager.

List:

Lists are used for temporary storage of known lists of items of known value, such as when retrieving a type of user from a file which has to be then combined with a list of other types.

Front-End:

On the front-end we didn't actually implement any data structure of relevance, apart from the most basic lists, mainly because this part was in charge of directing the user commands and info to the back-end.

Known problems

Front-End:

The editing is not available:

The editing functionality was not implemented, so there is no way of actually modifying products, categories, orders or any other object from the front-end.

Lacking communication:

Some of the parts of the front-end are not working properly because the communication with the back-end could not be made. Components like some of the list and such are not working because they lack the required GET function and others like the created or add components are not working because they lack the proper POST function.

Back-End:

Front-end can’t add parameters to the request’s body:

Because the front-end was unable to find a way to introduce parameters in the body of the request, the approach was changed from being read from the body to being read directly as parameters from the URI.

Found problems

Front-End:

Back-End:

Compatibility with Cors was a bit tricky, requiring downgrading the controller for the rest api to support it.

Work plan

Starting Date: 23/9
Deadline: 14/10
Time: 21 days

Sprint	Tasks	Length (days)	Date
1	**Goal: Planning and Investigation** ● C# ● ASP.NET ● Angular ● Bootstrap	7	23/09
2	**Goal: Develop the basic front-end and basic back-end functions** ● Views ● Data modeling ● Server HTTP ● JSON storage	5	30/09
3	**Goal: Integration** ● HTTP Request (GET,POST,DELETE, ...) ● Security ● Server view integration ● Android Studio app startup	4	5/10
4	**Goal: Complete the main features of the app** ● Data visualization ● User interaction ● App general view and data visualization	4	9/10
5	**Goal: Test and documentation** ● General testing ● Internal and external documentation	2	13/10

Work plan timeline

Task	**Description**	**Group**	**Dev responsable**
inserte aqui task	*inserte aqui task*	Front-end	Alvaro
inserte aqui task	*inserte aqui task*	Front-end	Daniel
REST Api resources and database development	Build and configure web api resources and features	Back-end	Jonathan
REST Api orders and products development	Build and manage orders within the database of the web api	Back-end	Jonathan

Group Logs

Activity	Date	Duration	Participants
Initial reunion for talking about the general specs and task division of the project	24/09	1h	All
Sprint planning meeting	28/09	1h	All
REST API Structure	01/10	1h	Back-end
REST API Structure for LogIn	05/10	30min	Back-end
REST API Resources division	09/10	1h	Back-end
REST API authentication methods	12/10	1h 30min	Back-end
REST Api Storage solution	17/10	1h	Back-end
REST Api Postman test	18/10	2h 30min	Back-end

Conclusions and recommendations

Front-end frameworks like angular are surprisingly large topics, if you've made it this far, though, you're more than well on your way to front end development magic. You should be significantly more comfortable breaking down a webpage into its component pieces and then coding them with HTML and CSS. You have the tools necessary to identify an effective visual layout and then bring it to fruition.

There are still plenty of ways you can make your workflow better or improve your knowledge of best practices, but you've got everything you need to build beautiful websites. In simple words, Front-End is that part of the web, where we are experiencing the elements that can be visualized or interacted with. It majorly comprises two segments the designs and the Front-End built-up or its development. Earlier when it was discussed anywhere about the development, the conclusion or interpretation was directly related to the back end. But now the scenario differs and the need to separate both domains is critical where design creators that really turn around magnificent objects through only Photoshop and the other brains who work out the coding phases to make it work. All that is achieved with a mixture of HTML, CSS and JavaScript including objects like buttons, fonts, indexes, menus, notification bar, any sliders or tabs, etc. Making all these things function dynamically and to secure all the data that is laid in the Front-End objects, is when the back-end comes in picture.

Bibliography

AngularJS (2019-02-05). Recuperado de: https://angular.io/

Bootstrap Themes & Templates (2019-02-05). Recuperado de: https://getbootstrap.com/docs/4.1/getting-started/introduction/