Campus Guidebook iOS

Authors: Isaak Wheeler, Haythem Odeh, Othman Djuliarso

Creation Date: 6/1/2022

Last Updated: 6/5/2022

Version: 1.0

Contents

[Database and Infrastructure 3](#_Toc104981857)

[DatabaseHelper 3](#_Toc104981858)

[Templates 7](#_Toc104981859)

## Database and Infrastructure

Introduction:

The database relies on one main class. The DatabaseHelper. This class creates tables if they do not exist already, gives access to methods for querying the database, and allows you to grab the open database reference just in case you need to use it.

### DatabaseHelper

|  |  |
| --- | --- |
| Usage | Explanation |
| let <ObjectName>: DatabaseHelper = DatabaseHelper() | When the object is initialized, the DatabaseHelper will run the methods CreateDB() and CreateTable(). CreateDB() will look for a database on your device and if it cannot find one, it will create one. This method returns the reference to the database. CreateTable() will run the first of the SQL. it has queries for setting up the specified tables in the database if none exist. This method calls initTable() which will run the queries.  The database is structured with 3 tables. Clubs, Sustainability, and Events. These tables store the information related to the Clubs, Sustainability, and Events pages. There is no connection between the tables at this time. |

#### Modification

|  |  |
| --- | --- |
| Desired Modification | Instructions |
| Adding Columns to the tables: | 1. Modify the creation queries located in the DatabaseHelper class in the method CreateTable(). 2. You will have to go to the object classes and change the values TableColumns, InsertableValueCount, and add variables that describe the new values before the enumerations. 3. You will have to add the new values to the enum CodingKeys in order to maintain the codable state of the object. Forth, you will need to add the values to the initalizer of the class. 4. Make sure InsertableValueCount has an integer that represents the total number of columns minus the ID column and TableColumns has a list of value names exactly the same as the create table query. 5. Go to the DatabaseHelper class and look for the add method for the table and modify the columnArray variable with the new values in order.   Note: Consistency is important. The program will not add values to the table in the correct order. After you add the columns to the create query, make sure the order is reflected in the init of the related object and TableColumns. |
| Adding a new table: | 1. Navigate to the CreateTable() method in DatabaseHelper. 2. You will have to create a new variable to represent the new create query. Format the name as such: Create<TableName>Table. The queries are structured as such: CREATE TABLE IF NOT EXISTS <TableName> (id INTEGER PRIMARY KEY AUTOINCREMENT, <ColumnName> <DataType>); 3. Create a new class that describes the table. There is a basic template located in the templates section of the document named “Table object class” 4. Create a new method in the DatabaseHelper class as such:   add<TableName>Row(<ObjectName>: <ObjectName>?)  Reuse the logic of other similar methods and modify values accordingly. columnArray will need to be edited. Use other methods for reference. |

#### Methods

|  |  |
| --- | --- |
| Method format, usage example, return type | Explanation |
| Method format:  RemoveDBTables()  Usage example:  <ObjectName>.RemoveDBTables()  Return type:  None | This method will remove all the database tables that were created by CreateTable(). This method is intended for debugging or testing but may be useful elseware. Note that this will delete all data in the database so be careful. This method makes use of the same CreateTable() method but passes delete queries instead of creating queries. |
| Method format:  GetAllTableContents(tablename: String)  Usage example:  <ObjectName>.GetAllTableContents(tablename: <NameOfTable>)  Return type:  Array of Arrays of type Any | This method will return all rows and columns from the specified table as a multidimensional array. The first index in the subarray will always be the ID of the row.  Format:   |  |  |  | | --- | --- | --- | | Array | 0 | 1 | |  | |  |  |  |  | | --- | --- | --- | --- | | 0 | 1 | 2 | … | | ID: Int | col2 | col3 | col4 | | |  |  |  |  | | --- | --- | --- | --- | | 0 | 1 | 2 | … | | ID: Int | col2 | col3 | col4 | | |
| Method format:  GetOpenDB()  Usage example:  <ObjectName>.GetOpenDB()  Return type:  None | This method returns a reference to the open database for use in queries that do not need to be included in the DatabaseHelper class. |
| Method format:  addClubRow(Club: Club?)  Usage example:  <ObjectName>.addClubRow(Club: <ClubObject>)  Return type:  None | Once you have created a club object, you can call this method to automatically add the club to a new row in the database. |
| Method format:  addEventRow(Event: Event?)  Usage example:  <ObjectName>.addEventRow(Event: <EventObject>)  Return type:  None | Once you have created a Event object, you can call this method to automatically add the Event to a new row in the database. |
| Method format:  addSustainabilityRow(Sustainability: Sustainability?)  Usage example:  <ObjectName>.addSustainabilityRow(Sustainability: <SustainabilityObject>)  Return type:  None | Once you have created a Sustainability object, you can call this method to automatically add the Sustainability to a new row in the database. |
| Method format:  removeRowByID(tableName: String, id: Int)  Usage example:  <ObjectName>.removeRowByID(tableName: <NameOfTable>, id: <IntValue>)  Return type:  None | This method will remove a specified row from a specified table based on the row ID. |
| Method format:  removeRowByName(tableName: String, Search: String)  Usage example:  <ObjectName>.removeRowByID(tableName: <NameOfTable>, Search: <NameToSearch>)  Return type:  None | This method will remove a specified row from a specified table based on the name in the row. This method may not delete anything if there is no row with the specified name. |
| Method format:  getRowByID (tableName: String, id: Int)  Usage example:  <ObjectName>.getRowByID (tableName: <NameOfTable>, id: <IntValue>)  Return type:  Array with values in the database row | This method will return array with the values of the row with a specified ID.  Format:   |  |  |  |  | | --- | --- | --- | --- | | 0 | 1 | 2 | … | | ID: Int | col2 | col3 | col4 | |

## Templates

|  |  |
| --- | --- |
| Template Name | Template |
| Table object class | //  // <TableName>.swift  // Campus-Guidebook-ios  //  // Created by <First Last> on <Date>.  //  import Foundation  import SQLite3  class <TableName>: Codable{  var TableName: String = "<TableName>" var TableColumns: String = "<NameOfColumnsInOrderSeperatedBy “, ”>"  var InsertableValueCount: Int = <NumberOfTableColumns>  Var <VarNameBasedOnColumnName>: <VarDataTypeBasedOnTableDataType>    private enum CodingKeys: String, CodingKey {  case <VarNamesSeperatedBy “,” ThatAreInTheDBTable>  }    init(<LowerFirstLetterVarName>: <DataType>?){  <VarName> = <LowerFirstLetterVarName> ?? ""  }  } |
|  |  |

## Category List Buttons – Collection View

|  |  |
| --- | --- |
| Content | Explanation |
| Introduction | It’s an object that manages an ordered collection of data items and presents them using customizable layouts. Using the collection view, you are able to create a layout for the category buttons. It is very similar to table view but with collection view you are able to create different layouts. |
| Functions | The collection view object has functions like numberOfItemsInSection, which you are able to define how many items in the view. Theres also cellForItemAt, which you can use to create, configure, and return the appropriate cell (each specific button)  for the given item. |

## Events, Clubs, and Sustainability - View Controller

|  |  |
| --- | --- |
| Content | Explanation |
| Introduction | A view controller manages your interface and facilitate navigation around your app's content. Its a page where the user can interact with the contents inside it. |
| Related category content in a single view | Having the view controller show the content belonging to the category in a single view controller was challenging yet possible. One of the ways it was done was, since the category buttons were in a collection view, one of its functions is didSelectItemAt, which tells me exactly which button i pressed by using the index. When redirecting to the view controller of the single page, we use the indexes to present the relative content to any of the above categories. |
| Sample data from the database | The data was created with lists for each category. Using a function from the database helper class like addClubRow, addEventRow, addSustainabilityRow, it was very easy to add to the database. Then using function from database helper class, getAllTableContents(tablename: "Club"). This will get all the data from the table name and store it into a list. Then using the function cellForRowAt, from table view which you can use to create, configure, and return the appropriate cell for the given item. This will show the sample data in a list. |
| Images | There are 3 ways of storing images into the database. One is storing the URL, second is the image name, and finally no image at all but empty “ ”. When checking for the image, there are a couple functions used. The first one is checking if its a URL. Creating an extension to String, isValidURL which checks if the image is a valid URL. Using another function getImg, I was able to check if the image had a valid URL, it it did, then it will use an extension of UIImage withContentsOfUrl, which will show the image from it’s URL in the view controller. Otherwise it will check if its not a URL, the function will check if it returned anything, if the image is blank, if so it will return a default image from the assets folder and load it into the view. If its not empty it will check the assets folder for the image and return the appropriate image. |

## Card Layout and Custom Card Cells - Table View

|  |  |
| --- | --- |
| Content | Explanation |
| Introduction | The table view displays and presents data using rows in a single column which features a prototype cell functionality. Using a table view allows you to scale the content of the table for smaller or larger sizes screen, rather than a fixed content when using other UI Views. |
| Features | The table view object features a function to return a reusable table-view cell object for the specified reuse identifier with the dequeReusableCell function. The table view object also features a function to autofill the number of rows in a specified section with the numberOfRowsInSection function argument based on the number of elements for that specific section. |

## Card View Layout and Custom Card Cell - A Table View to display the Data in a List

|  |  |
| --- | --- |
| Content | Explanation |
| Table View and Controller | - The UI Table View is the second-level view following the Main Navigation Collection View, as displayed on the landing page.  - The Table View takes the entire screen which has a prototype cell (the card layout itself) that will display an image, a title, and a short description of a specific Club, Sustainability, or Event table’s row.  - This table view is connected to a view controller that allows you to automatically fill how many rows exist in the database in each table.  - This table view also acts as a delegate to the Detailed Card View scene that displays the full descriptions of a table row’s content by using the didSelectRowAt argument to tell the delegate that the specified row is selected to navigate to the Card Details View object with that selected row index path. |
| Custom Cell Class | - The UI Table View Cell is a visual representation of a single row in a table view.  - The table view cell object is a specified and customizable type of view that manages the content of a single table row.  -  Any table view must have at least one type of cell to display the content.  - The Card Cell object allows you to configure the cell’s content and layout.  - The prototype of the cell is configured to get each of the specific row’s data (such as the image, title, description) of a Club, Event, or Sustainability table,  by using the database helper methods: by first adding the data to the table, such as addClubRow, addEventsRow, addSustainabilityRow, then using the method to display that content for that selected cell, such as getAllTableContents.  - The card cell’s configure function allows the user of this function to display the image, title, and description of a table row with a customizable layout and behavior. |