**Invoice Issuer**

**About the app:**

Invoice issuer is an application that lets sends invoice information to customer’s email.

The app provides the sender with the ability to send the invoice using three different invoice templates.

The app provides the sender with a history of his/her previous invoices and it is categorized by customers’ emails.

**Technical view of the app:**

**App Functions:**

The app must be able to provide the following functions:

1. The ability to choose a customer email from list of emails.
2. The ability to choose an Invoice Template among three different templates.
3. The app must provide the sender with a history of his/her previous invoices , it must be categorized or grouped by the customer.
4. The app must provide a simple report of the history in a PDF format.

**Analytical Design:**

The app is data-driven based upon having a database that stores data needed by the app in order to perform it’s main functions.

In the following sections we are going to discuss about Database Design

**Database Design:**

The main entities that the app must be able to track its data and process it is:

1. **Users**
2. **Invoices**

Users are divided into two main sections:

* + **Senders**
  + **Receivers**

All have the following attributes:

* **User name**
* **User address**
* **User phone**
* **User email**
* User type (sender – receiver)

Invoices have the following attributes:

* **Invoice number**
* **Invoice QR code**
* **Invoice due date**
* **Invoice payment status**
* **Invoice sender email**
* **Invoice receiver email**

Invoices are created by items that the customer bought

Therefore their must be a new table that describes the items

Of the invoice lets name it Invoice Items

Invoice Items have the following attributes:

* **Item**
* **Unit Price**
* **Quantity**

**Breif Overview of how django API is working:**

Django uses the MVT (Model-Veiw-Template) architecture to design and develop web applications that it is more flexible and easy to maintain later.

Django generally divides the code into modules called Apps which works as units that is used as structures to make the code more maintainable.

These apps are divided into the following files:

* **Models file:**  is responsible for representing the data models of the system, which is translated to the tables of the database and its the core of data layer.
* **Views file**: is responsible for represinting the views classes , which is the core of the business layer.
* **Templates files**: are responsible for the representation layer.
* **Url file**: is responsible for creating URLs of the web app.

Django supports the creation of the Restful APIs by using a dedicated framework drived from it which is Django Rest Framework.

**Django Rest Framework:**

It uses the same structure of the basic Django Framework

In addition to the following files:

* **Serializers files**: are responsible for representing the translation of the data models into serializable objects.