ISTITUTO DI INGEGNERIA DEL MARE
INSTITUTE OF MARINE ENGINEERING

RAPPORTO TECNICO INTERNO

TITOLO	*Progetto e sviluppo della una piattaforma software Tidal Tools per la modellazione digitale di turbine idrocinetiche*

AUTORE/I	**Nome**	**Cognome**	**Matricola**
1	Pedram	Ghorbanpour	*n.a.*
2	Francesco	Salvatore	40493

PROGETTO/ COMMESSA	Progetto ULYSSES – ***U****nderpin* ***L****aborator****Y*** *for* ***S****tdies on* ***S****ea* ***E****nergy* ***S****ystems*	**Identificativo** DIT.AD019-040
RESPONSABILE/I	Francesco Salvatore	
COMMITTENTE/I	CNR	
PARTNER/S	-	
PAROLE CHIAVE	Energia rinnovabile da fonti marine; Correnti di marea; turbine idrocinetiche; piattaforma software; modelli digitali	

DIFFUSIONE	*Privata*		*Riservata*		*Pubblica*	X
NOTE					**Revisione**	

	NOTE GENERALI D'ISTITUTO	*23/06/2022*

CENTRO DI RICERCA DI APPARTENENZA	CNR – Consiglio Nazionale delle Ricerche		
DENOMINAZIONE ISTITUTO	CNR – INM *Istituto di Ingegneria del Mare*		
SEDE/INDIRIZZO	Via di Vallerano, 139 – 00128 Roma (RM)		
TEL	06 502991	**FAX**	06 5070619
E-MAIL	segreteria.inm@cnr.it	**PEC**	protocollo.inm@pec.cnr.it
WEBSITE	www.inm.cnr.it		
DIRETTORE D'ISTITUTO	Dott. Ing. IAFRATI Alessandro		

Visto, si approva
IL DIRETTORE
(Nota 15)

Alessandro Iafrati
27.06.2022
15:44:26
GMT+01:00

Sede principale: Via di Vallerano, 139 – 00128 Roma
P. IVA 02118311006 – C.F. 80054330586
E-mail: segreteria.inm@cnr.it PEC: protocollo.inm@pec.cnr.it Tel. 06-50299222 - Fax 06-5070619 http://www.inm.cnr.it

Sede di Roma "Sezione di Acustica e Sensoristica O.M. Corbino" Area della Ricerca di Tor Vergata Via del Fosso del Cavaliere, 100 - 00133 Roma

Sede di Genova Area della ricerca di Genova Via De Marini, 6 – 16149 Genova

Sede di Palermo Area della ricerca di Palermo Via Ugo La Malfa, 153 – 90146 Palermo

TABELLA DEI CONTRIBUTORI

Autore di Riferimento ai fini della gestione del prodotto

Nome	*Cognome*	*Matricola*
Francesco	Salvatore	40493

Autore/i

Nome	*Cognome*	*Matricola*
Pedram	Ghorbanpour	*n.a.*
Francesco	Salvatore	40493

Nota di Riservatezza:
nessuna

ISTITUTO DI INGEGNERIA DEL MARE
INSTITUTE OF MARINE ENGINEERING

INTERNAL TECHNICAL REPORT

TITLE	*Design and development of the Tidal Tools software platform for the digital modelling of hydrokinetic turbines*

AUTHOR/S	Name	Last Name	Service Number
1	Pedram	Ghorbanpour	*n.a.*
2	Francesco	Salvatore	40493

PROJECT	Project ULYSSES – ***U**nderpin **L**aborator**Y** for **S**tdies on **S**ea **E**nergy **S**ystems*	**ID Number**
		DIT.AD019-040
MANAGER/S	Francesco Salvatore	
CLIENT/S	CNR	
PARTNER/S	--	
KEYWORDS	Marine renewable energy; Marine currents; Hydrokinetic turbines; software platforms; digital models	

CONFIDENTIALITY	*Private*		*Restricted*		*Public*	X
NOTES					**Revision**	

	INSTITUTE DETAILS	23/06/2022	
ORGANIZATION	CNR – Consiglio Nazionale delle Ricerche		
INSTITUTE	CNR – INM *Istituto di iNgegneria del Mare*		
OFFICE/ADDRESS	Via di Vallerano, 139 – 00128 Roma (RM)		
PHONE	06 502991	**FAX**	06 5070619
E-MAIL	segreteria.inm@cnr.it	**PEC**	protocollo.inm@pec.cnr.it
WEBSITE	www.inm.cnr.it		
INSTITUTE DIRECTOR	Dott. Ing. IAFRATI Alessandro		

Approved
The DIRECTOR

CONTRIBUTORS TABLE

Corresponding Author

Name	***Last Name***	***Service number***
Francesco	Salvatore	40493

Author/s

Name	***Last Name***	***Service number***
Pedram	Ghorbanpour	*n.a.*
Francesco	Salvatore	40493

Confidentiality notes:
No.

Table of Contents

1 Introduction

The present report describes the development and implementation of a computational platform for the analysis and design of tidal energy turbines. The platform is called **TidalTools** and is a web application (website) which is designed to be accessible by remote connection using a common web browser. Like its name it deals with design, performance simulation, economic feasibility, etc. problems in the tidal energy industry and particularly it is designed to perform a horizontal-axis tidal turbine characterization from scratch with some simple inputs. Results of computational studies are made available as output files in various formats, plots, and a report of turbine key performance indicators and geometry details is provided.

The idea of developing this tool was first started on September 2020. The main objective was to make available to a wide community of users the digital models on tidal energy systems developed at CNR-INM (INSEAN, until 2018) and grounded on two decades of activity in the hydrodynamics modelling of rotary wing systems. Core of this computational suite is a Boundary Integral Equation Model (BIEM) for the analysis of tidal turbine performance and the mesh generation code (AUTOBLADE) to discretize 3D turbine models for the numerical solution by BIEM. More recently, these two codes have been integrated into a turbine parametric design procedure (X-Design) and annual energy production, techno-economic models have been included.

All the above-mentioned codes have been implemented in the Fortran environment and design and developing a user interface to these codes was not possible with the same programming language, so the solution was to use Django (Holovaty 2005), a high-level Python-written web development framework, to provide database management and graphical user interface to the computational models linked into the platform. Django is one of the most used programming languages among the scientists and developers.

One of the key features of this platform other than simulations and analysis is having a sophisticated database structure to manage all the data that are used as input and are generated as output in the calculations. In particular, one database is specifically designed to store all the important information regarding the different oceanic and tidal sites all around the world. These data can be the velocity, bathymetry, salinity, density measurements in particular periods of time along with metadata for name, location, coordinates, etc. of the site.

Besides providing the mentioned services to the users, the social media and forum functionality is also considered in this platform. The objective is to propose **TidalTools** as the first and only hub for students and scientists to get in touch, share, collaborate and together move toward further developments in the tidal energy field.

The platform is designed as a web application (website) accessible by remote connection using a common web browser. This layout provides several advantages from both users and developers sides:

- The users do not have to install any software on their local computers: this prevents compatibility issues with operating system, graphical libraries in the local computer
- No license keys or software is necessary: the user registers online to have access to the platform
- Developer's activity to update the software, to fix bugs, to introduce new features can be done in parallel to users access to the platform
- When a new platform version is released by the developers, the users do not have to upgrade or install new software. They just receive a message with notification of the new features.

The platform software is hosted by a server located at CNR-INM.

In the following sections of this report, detailed explanation of all tools and of the interfaces to manage input and output data through databases and a graphic user interface are given. Particular attention is given to describe the platform architecture that has been designed to enable a wide range of functionalities with a modular approach that allows a streamlined upgrade of the platform as new tools and features will be made available for integration. In fact, **TidalTools** is conceived as a long-term project, open to contributions from a wide community of users and developers.

2 TidalTools – An overview

TidalTools is an online web application with both computational, data management and social tools. This section illustrates the structure of the platform and the main tools that are integrated .

This online platform is developed to target the tidal energy sector (for now) but can be further developed to include all offshore energies. The aim in development of this tool is to facilitate the complicated design procedure of a tidal turbine by managing a very few but crucial data. The rest is being handled *in the back-end*. Besides the design and computational features, this tool benefits from a social part that is developed to connect the people in the tidal energy community. Everyone who is studying, training, working, and doing research in this field can look at this platform as a hub to exchange ideas, organize event, run polls, get in touch with each other through a fully encrypted messaging platform. Providing these two features both at the same platform is what makes the **TidalTools** unique and one of a kind.

The platform is designed to be accessible by remote connection using a common web browser from any device with any operating system. The advantage of this approach over the conventional offline approach is:

1- Users can always have access to the latest version of the code.
2- No huge file to download every time a new version of the platform is being published
3- Not being depended on the operating system
4- User can have access to the code from any device that has a connection to the internet i.e., Tablets and smartphones.

The platform combines several software elements dedicated to providing the users with specific services. However, the platform as mentioned, consists of two main parts which are:

1- The Computational Tools:
Where users can perform many tasks like design and performance evaluations of the horizontal axis tidal turbine.

2- The Social part:
Where users get in touch with each other and share their research with other and the admins and authors of the platform.

The computational tools available in this platform are developed in CNR-INM in over 20 years and all has been validated. In the current version of the platform a few numbers of tools are available. However, Other tools will include in the future by taking advantage of the modular structure of the platform architecture. The tools that are available now are:

1. **AUTOBLADE:**
This tool is used to build a three-dimensional (3D) model of a horizontal-axis turbine and to generate the computational mesh used by the BIEM (Boundary Integral Equation Model) solver for the prediction of turbine performance (see below). As input, geometry

parameters to build turbine blade and nacelle surfaces are required by using a specific data format. As output, turbine surface mesh and 3D models in formats compatible with popular graphics software (Tecplot, Paraview, Gnuplot) are given. The resulting turbine 3D model and computational mesh can be uploaded to the platform turbine geometry database.

2. **BIEM**
   This tool is used to evaluate the performance of a horizontal-axis tidal turbine by using a BIEM (Boundary Integral Equation Model) solver that is valid for steady or unsteady 3D flows. Turbine performance for user defined operating conditions (mean flow speed, TSR) and site conditions (vertical velocity profile) are evaluated over a TSR range or for a single TSR value. As input, the turbine surface mesh by AUTOBLADE is used. A simple viscous flow correction model allows to estimate performance up to stall, provided that blade section polars are given as input. Simulations results include thrust, torque, power curves, blade surface distributions of pressure, risk of cavitation on blade, radial distributions of induced velocity and effective angle of attack. The output files are formatted for visualization with popular graphics software. The resulting turbine performance curves can be uploaded to the platform turbine performance database.

3. **X-Design**
   This tool is used to perform a horizontal axis tidal turbine design from user defined objectives and constraints. The design follows by an iterative procedure in which X-Design is interfaced with the AUTOBLADE tool to generate 3D models and grids of guess turbines, whereas the BIEM tool is used to predict the performance of the guess turbine. As output the geometry and performance of the designed turbine are given. If the design is successful, the resulting turbine geometry can be uploaded to the platform turbines database

4. **SiteSim**
   This tool is used to estimate the power output, the operating conditions (mechanical loads, RPM) and Annual Energy production (AEP) of a user-defined turbine in a user-defined tidal site. Both turbine geometry and tidal site resource data are taken from dedicated platform databases.

5. **EcoSim**
   This tool is used to perform the economic feasibility analysis of a turbine in a particular site and generate a report that contains all the costs associated with constructing, operating, and maintaining it. The tool can also provide information like the Levelized Cost of Energy (LCoE), payback period, and other performance indicators that are used to perform a Life-Cycle Assessment (LCA) of a given tidal energy project.

Users when register on this platform, can benefit from all there modules. Besides, as mentioned above they can use social features considered inside the platform to share and contact others, these features are:

1. **Messaging feature**
   This is the classic messenger that allow users to communicate which each other

2. **Forum**
   Where users can share their research and results that they obtain by using the tools available in this platform, run polls, promote their research, etc.

In the present document, some of the platform modules will be referred to as *Vertical* like computational modules whereas some will be referred to as *Horizontal* like the users module. Vertical module is a module that is built to perform a single task and can work solo. However, a horizontal module is a module which is interacting with other modules and communicate with other vertical and horizontal modules[Figure 1]. The means of communication in the platform is provided by the data stored in the database. All the interactions and data storage and relations between stored data [Figure 2] is explained in the following section.

To summarize, **TidalTools** is not only an input/output platform but also provides means of communication and networking.

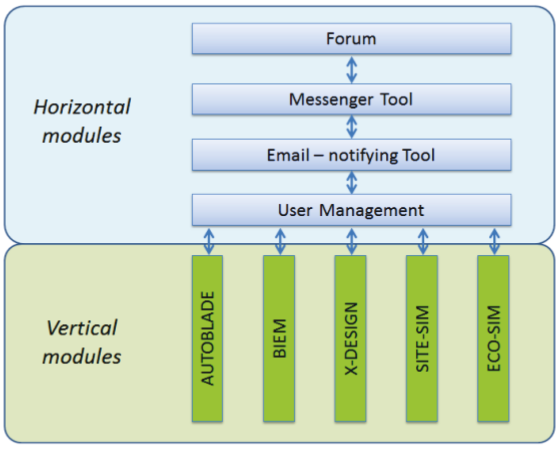

Figure 1 Vertical and Horizontal modules in the TidalTools platform

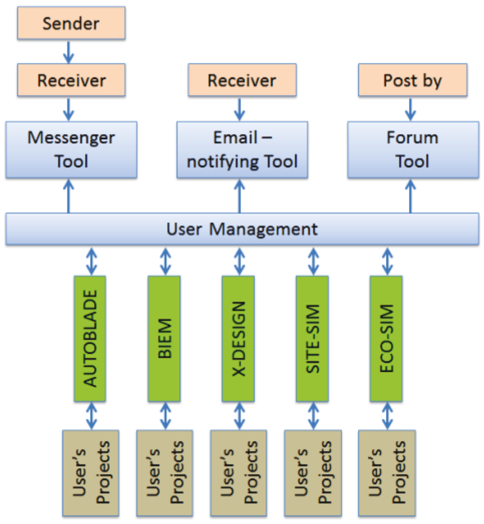

Figure 2 Relating databases in the TidalTools platform

3 The Platform Architecture

Before describing the platform architecture in detail, it is deemed useful to recall some general concepts that are common in the design and implementation of digital products like the one of interest here. Like every digital platform, **TidalTools** has an algorithm architecture and development process that this section aims to cover. The two most important parts for an application or in other word a computer program are:

1- **Back-end code**

Back-end refers to parts of a computer application or a program's code that allow it to operate and it cannot be accessed by a user. Most data and operating syntax are stored and accessed in the back-end of a computer system. Typically, the code is comprised of one or more programming languages. The back end is also called the data access layer of software and includes any functionality that needs to be accessed and navigated to by digital means.

2- **Front-end code**

The layer above the back end is the front-end and it includes all the software that is part of a user interface. Human or digital users interact directly with various aspects of the front end of a program, including user-entered data, buttons, programs, websites, and other features. Most of these features are designed by user experience (UX) professionals to be accessible, pleasant, and easy to use.

A back-end application or program supports front-end user services, and interfaces with any required resources. The back-end application may interact directly with the front end, or it may be called from an intermediate program that mediates front-end and back-end activities.

Usually, the back-end of a platform consists of the codes and libraries that are written in different programming languages. **TidalTools** also has its back-end developed using python and Fortran. The code in the back end consists of FORTRAN executables that are the solvers for different design and analysis and Python scripts that produce the whole structure of the platform and provides the means of communication between the user interface and the computational code in FORTRAN. It worth mentioning that using FORTRAN as a part of the back-end of a website was never seen before as one of the seniors of AWS states in Quora [1]. On the other hand, most of engineering software developed over the last decades is written in Fortran 77 or more recent versions. In both academia and industry, continuous developments on the software is being done using the same language. The necessity to develop interfaces to exploit a vast knowledge background into modern digital frameworks is then apparent. The computational platform project described here is an example of this type of effort.

In the present work, the platform has been designed for operation under Linux operating system.

3.1 Why Python

Python is a widely used general-purpose, high level programming language. It was created by Guido van Rossum in 1991 and further developed by the Python Software Foundation. It was designed with an emphasis on code readability, and its syntax allows programmers to express their concepts in fewer lines of code.

Python is a programming language that lets you work quickly and integrate systems more efficiently.

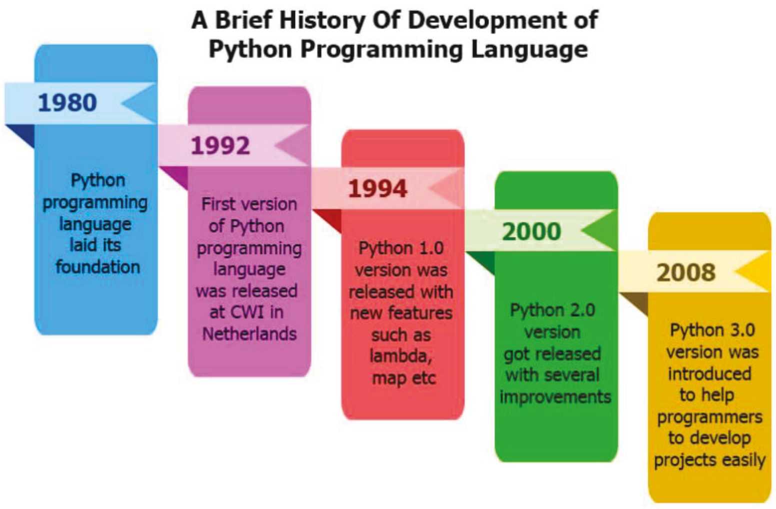
A Brief History Of Development of Python Programming Language
1980
Python programming language laid its foundation
1992
First version of Python programming language was released at CWI in Netherlands
1994
Python 1.0 version was released with new features such as lambda, map etc
2000
Python 2.0 version got released with several improvements
2008
Python 3.0 version was introduced to help programmers to develop projects easily

Figure 3 Python Development History

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built-in data structures, combined with dynamic typing and dynamic binding, makes it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python is simple, easy to learn syntax, it emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms and can be freely distributed. [2]

Nowadays, Python is very popular among the developers, engineers, and Data Scientists. That's why due to its huge community around the world many free source libraries have been developed are developing every day. Python is used in many areas such as Machine learning (ML), Artificial intelligence(AI), User interface development, big data analysis etc.

One area where Python shines is web development. Python offers many frameworks from which to choose from including bottle.py, Flask, CherryPy, Pyramid, Django and web2py. These frameworks have been used to power some of the world's most popular sites such as Spotify, Mozilla, Reddit, the Washington Post and Yelp. [3] For developing the **TidalTools** platform the Django Framework is used.

3.2 Back-end

As mentioned at the beginning of this section, this tool uses multiple languages at its back-end. Part of it is developed in FORTRAN and part of it is written in Python. The syntaxes and working principles of these two programming languages are not the same and that was the first barrier in the development of the platform. Luckily, there is a library called F2PY which interfaces FORTRAN into Python. However, unfortunately, this library proved not adequate to translate complex codes written in Fortran as those considered for inclusion into the TidalTools platform. Therefore, a library called TIDALDES has been developed from scratch to establish this interface between FORTRAN and Python in the back-end and provide the means of communication between their two parts of the back-end.

This library was developed to be able to generate the input file of the FORTRAN solver then access to the directory where the Fortran executable is located, execute the code, then read all the output generated by the code and finally process them in a way the python code can understand and interact with them.

The working principle of the TIDALDES module is more like a robot that executes some series of tasks when they are needed. In other words, it mainly and mostly is the python that is running. But is also important to know the logic behind this interface.

The files that this platform uses are all stored in a particular architecture to make them interact easily with each other. All the Fortran codes in the back-end are stored in a directory associated with the solvers name and all those directories are inside a directory called **Fortran_solvers [**Figure 4**]**. Then all the solvers are compiled using GFortran. Then , these solvers connected to the TIDALDES module.

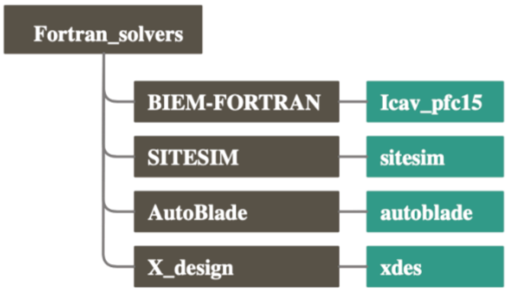

Figure 4 Gray boxes show the directory, and the green ones show the Fortran executable

When inside the platform any of the mathematical solvers are called, a subprocess starts in a particular directory. Then, results are processed by the Fortran interface and then passed to the Python and then to the database.

To develop the back-end of this platform a web framework called Django is used. This framework shapes the whole structure of the platform and provides an interface between the back-end and the database to dynamically get or send data from or to the database. The structure of the back-end of the platform is shown in Figure 5.

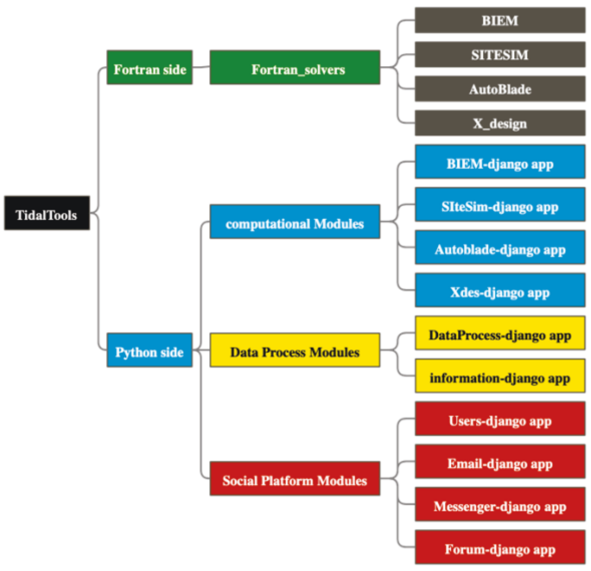

Figure 5 Back-end structure of the platform

3.2.1 Django Framework

The **TidalTools** platform as mentioned before is developed by using Django python. Django is a back-end framework, but this tool also support a front-end development tool that are not as good as the ones for the front-end. This tool can connect the back-end code and database directly to the html code of the UI and dynamically change data. However, for some of the features considered for future developments of the platform, front-end engine must be developed.

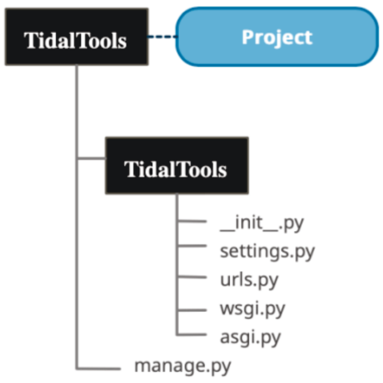

Figure 6 TidalTools directory and files generated by Django-admin

In the Django development the convention is to call the website a project and the features that are being developed individually to include in website, are called django applications. when a project starts, by using the `django-admin startproject project_name`' command some default directories and file generate automatically by a library called Django-admin. When a project is being instantiated using django-admin, a directory is being created with the name of the project as shown in Figure 6. Inside this directory, a directory has also been created with the same name as the project. This directory contains the configuration files of the project which are:

1. **__init__.py**
   This is an empty file as you can see below in the image. The function of this file is to tell the Python interpreter that this directory is a package and involvement of this __init.py_ file in it makes it a python project.

2. **Settings.py**
   It contains the Django project configuration

3. **Urls.py**
   URL is a universal resource locator; it contains all the endpoints that we should have for our website. It is used to provide you the address of the resources (images, webpages, websites, etc.) that are present out there on the internet. In simpler words, this file tells Django that if a user comes with this URL, direct them to that website or image whatsoever it is.
4. **Wsgi.py**
   WSGI stands for Web Server Gateway Interface, it describes the way how servers interact with the applications.

5. **Asgi.py**
   ASGI works like WSGI but comes with some additional functionality. ASGI stands for **Asynchronous Server Gateway Interface**. It is now replacing its predecessor WSGI.

Settings.py is the most important file, and it is used for adding all the applications and middleware applications. This is the main setting file of the Django project. This contains several variable names, and if you change the value, your application will work accordingly. It contains sqlite3 as the default database. We can change this database to MySQL, PostgreSQL, or MongoDB according to the web application we create. It contains some pre-installed apps and middleware that are there to provide basic functionality. In other words, all the settings related to the key features of the platform is being handled here.

Besides three different databases, an API interface is also developed for this platform in Django-rest framework that can be modified from settings of the platform. Whenever a new module or web application is being added to the platform it must be linked to the platform. This linking process is also can be done from the settings code. For this reason, every time an app is added the path to that app should be added to the list of installed_apps inside the settings.py

Django uses the concept of Projects and apps for managing the codes and presents them in a readable format. A Django project contains one or more apps within it, which performs the work simultaneously to provide a smooth flow of the web application. By running the 'python manage.py startapp application-name'

In the development of this platform, different applications were developed for different parts of the platform. In other words, each feature of the platform is developed as a django application that can be modified without effecting the other features of the platform. This kind of flexibility is prevised to help developers make tests without effecting the integrity of the platform. On the other hand, with this approach a particular feature can be shared with others without sharing the whole code. Among the available applications some are developed to handle the user log info, and some are developed to provide communicational services and data process. These modules are:

1. **User application**

   This module deals with user information and user logs. The database associated with this model is used as a foreign key in the other modules.

2. **Information application**

   This module deals with website content and contact us section of the website

3. **Data Process Module**

   This module deals with the storing the data that user publishes to the public and some machine learning features will later added here to train a model later used to replace the long process of the mathematical design

Beside the mentioned applications, others are developed to perform a link between the computational model and the interface. In this case, for each solver a different django application is developed which are as follows:

1. SiteSim Module
2. AutoBlade Module
3. BIEM Module
4. X-design Module
5. EcoSim

All the mentioned applications are made automatically by using new project command in the project root directory and they have the structure shown in Figure 7.

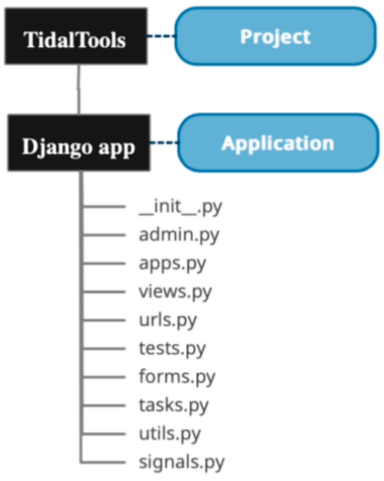

Figure 7 Django application files

The automatically generated django applications by default contain six important python scripts that are:

1. **__init__.py**

   This file provides the same functionality as that in the _init_.py file in the Django project structure. It is an empty file and does not need any modifications. It just represents that the app directory is a package.

2. **Admin.py**

   Admin.py file is used for registering the Django models into the Django administration.

   It is used to display the Django model in the Django admin panel. It performs three major tasks:

   a. Registering models

   b. Creating a Superuser

   c. Logging in and using the web application

3. **Apps.py**

   Apps.py is a file that is used to help the user include the application configuration for their app. Users can configure the attributes of their application using the apps.py file. However, configuring the attributes is a rare task a user ever performs, because most of the time the default configuration is sufficient to work with.

4. **Models.py**

   Models.py represents the models of web applications in the form of classes. It is considered the most important aspect of the App file structure. Models define the structure of the database. It tells about the actual design, relationships between the data sets, and their attribute constraints.

5. **Views.py**

   Views provide an interface through which a user interacts with a Django web application. It contains all the views in the form of classes. Some of these are Custom Filter Views, Class-Based List Views, and Detail Views.

6. **Urls.py**

   It works the same as that of the urls.py in the project file structure. The primary aim being, linking the user's URL request to the corresponding pages it is pointing to.

7. **Test.py**

   Tests.py allows the user to write test code for their web applications. It is used to test the working of the app.

Some of the modules due to the complexity of the back-end of them or demand for parallel computing have some additional scripts that are:

1. **Tasks.py**

   This script contains a module called Celery which is a parallel computing server manager developed in python that allows the routing happens in the main server and the calculations execute in the parallel server so when for example a design procedure is being executed the routing code and redirect the view to the home page or any other page. Without this the user should wait for as long as it takes for the code to execute which is not favorable from the user experience point of view.

2. **Utils.py**

   This part of the code contains the back-end logic of the module.

3. **Signals.py**

   This part of the code monitors the database operations and execute the code when some certain operation triggers it.

Each application has a folder called templates that contain all the html files that are needed by that application to load the user interface of the module. This part of the application will be deeply explained in **Front-end** sub section of this section.

3.2.2 Database Engine

Django attempts to support as many features as possible on all database back-ends. However, not all database back-ends are alike. Among them for this platform, SQLite, PostgreSQL databases connections were developed. As mentioned in the previous subsection all the configurations regarding the database this platform uses is handled in the settings.py.

In TidalTools, different databases were stablished. These settings allows the platform to change the engine of the database based on availability. These different databases are as follows:

1. **Local SQLite database(default)**

   SQLite is a database engine written in the C language. It is not a standalone app; rather, it is a library that software developers embed in their apps. As such, it belongs to the family of embedded databases. SQLite supports **an unlimited number of simultaneous readers**, but it will only allow one writer at any instant in time. For many situations, this is not a problem. Writers queue up. Each application does its database work quickly and moves on, and no lock lasts for more than a few dozen milliseconds. SQLite works great as the database engine for most low to medium traffic websites (which is to say, most websites). The amount of web traffic that SQLite can handle depends on how heavily the website uses its database. Generally speaking, any site that gets fewer than 100K hits/day should work fine with SQLite. The 100K hits/day figure is a conservative estimate, not a hard upper bound. SQLite has been demonstrated to work with 10 times that amount of traffic. [4]

2. **PostgreSQL (Local server)**

   PostgreSQL is a powerful, open-source object-relational database system that uses and extends the SQL language combined with many features that safely store and scale the most complicated data workloads. The origins of PostgreSQL date back to 1986 as part of the **POSTGRES** project at the University of California at Berkeley and has more than 30 years of active development on the core platform.

   PostgreSQL has earned a strong reputation for its proven architecture, reliability, data integrity, robust feature set, extensibility, and the dedication of the open-source community behind the software to consistently deliver performant and innovative solutions. PostgreSQL runs on **all major operating systems**, has been **ACID**-compliant since 2001, and has powerful add-ons such as the popular **PostGIS** geospatial database extender.

3. **Cloud PostgreSQL server Amazon Web Service (AWS)**

   Amazon web service provides many types of databases but for this particular case the settings for an online PostgreSQL database was implemented also inside the settings for future developments

As mentioned, (see above) the SQLite database can support a website with 100k users per day and is free and open source and is the default database of the Django. Therefore, for this platform at the beginning this database engine is chosen. However, the future migrations to any other database engine is also considered when developing the database configurations.

Each Django application in this platform has its own data tables in the database. The general visualization of the databases and how they are related to each other is shown in the Figure 8.

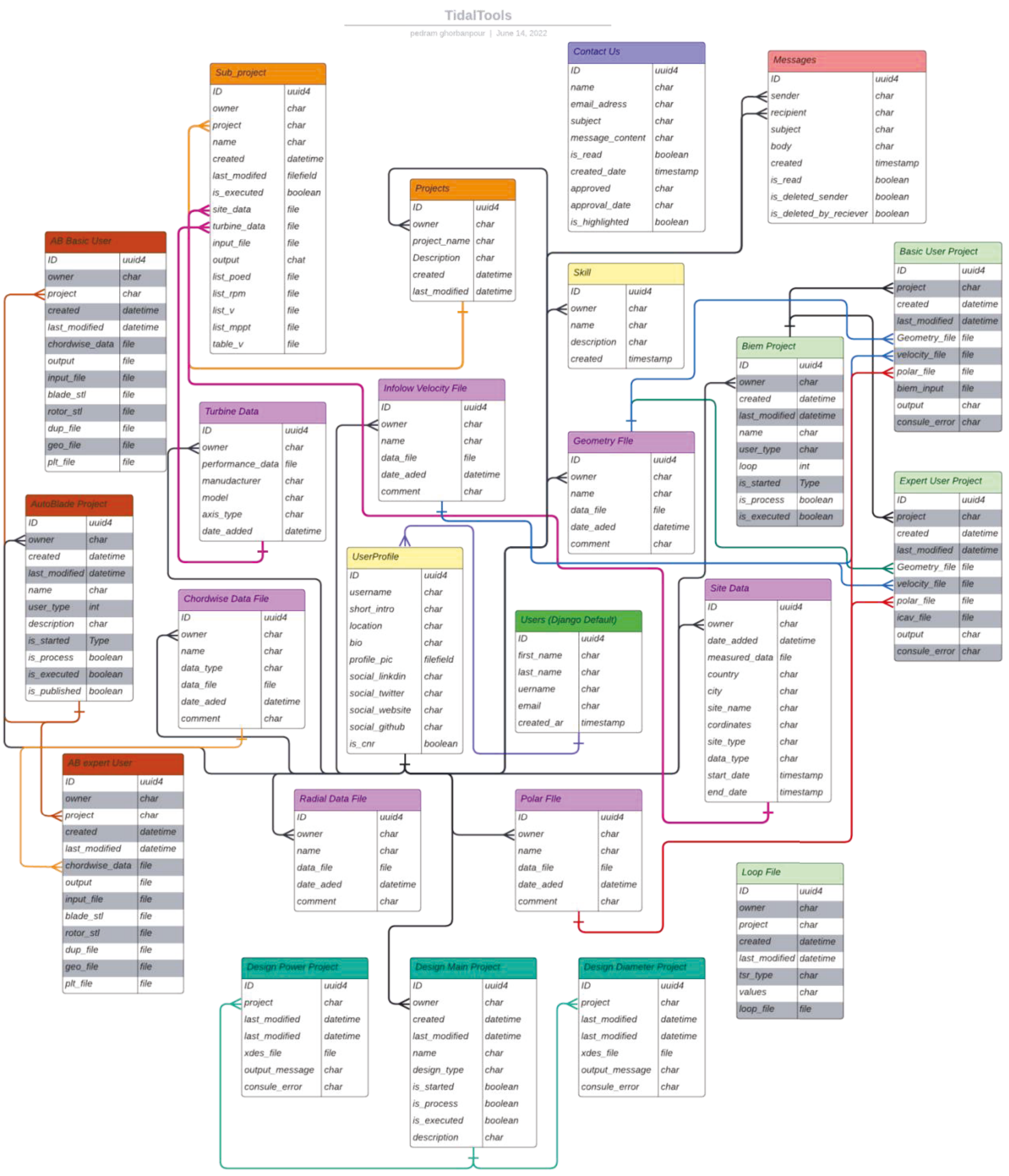

Figure 8 Database Entity Relationship Diagram

3.2.3 Celery Parallel server

The back-end of the **TidalTools** as mentioned consists of two parts, the framework developed in python and the mathematical solvers are developed in Fortran. The design algorithm (X-design) and performance simulation (BIEM) based on the number of TSR values and the dimension of the turbine, take between couple of minutes to half an hour. Therefore, for providing a smoother experience for the users a parallel computing code was developed. This code is developed by using the celery library of the python. This aspect is of primary importance in view of multiple users having access to the platform at same time.

Celery is an open-source asynchronous task queue or job queue which is based on distributed message passing. While it supports scheduling, its focus is on operations in real time. Task queues are used as a mechanism to distribute work across threads or machines. A task queue's input is a unit of work called a task. Dedicated worker processes constantly monitor task queues for new work to perform. Celery communicates via messages, usually using a broker to mediate between clients and workers. To initiate a task the client adds a message to the queue, the broker then delivers that message to a worker. A Celery system can consist of multiple workers and brokers, giving way to high availability and horizontal scaling. [5]

The approach used in the **TidalTools** platform development was to develop task managers for both design algorithm and BIEM code. The broker which is used to provide means of communication between celery and the platform is Redis. Redis is an in-memory data structure store, used as a distributed, in-memory key–value database, cache, and message broker, with optional durability. With this approach celery can assign a CPU core (Worker) per operation which will guarantee a robust and smooth calculations for the users of the platform. In other words, celery will manage threads instead of operation system.

3.2.4 Key Python Libraries

For developing this online tool, many pythons' open-source libraries is used. Django and Celery were the most important ones since the foundation of the system was built using those libraries. However, some libraries also used to data process, data visualizations', and automatic pdf or excel report generations these libraries are as follows:

1. **Pandas**
   pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series. [6]

2. **NumPy**
   NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. [7]

3. **Matplotlib**
   Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK. [8]

4. **Seaborn**

Seaborn is a library that uses Matplotlib underneath to plot graphs. It will be used **to visualize random distributions**. [9]

5. **PyPDF3**
   PyPDF3 is a pure-python PDF library capable of splitting, merging together, cropping, and transforming the pages of PDF files. It can also add custom data, viewing options, and passwords to PDF files. It can retrieve text and metadata from PDFs as well as merge entire files together. [10]

3.3 Front-end

So far, the architecture of the code and the directories for developing the back-end of the platform explained. However, the most powerful back-end is useless without a powerful front-end. In development of TidalTools, no particular front-end engine used. Django framework has a tool implemented in it which uses jinja template engine that can connect the back-end to the front-end and dynamically change the information that are being displayed. Also, the information entered by the user when filing a text field or a form is handled at the front-end and then to the back-end.

To develop the front end of this code different front-end languages were used which are:

1- **HTML**
   The Hypertext Markup Language or HTML is the standard markup language for documents designed to be displayed in a web browser.

2- **CSS**
   Cascading Style Sheets is a style sheet language used for describing the presentation of a document written in a markup language such as HTML. CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript.

3- **JavaScript**
   JavaScript, often abbreviated JS, is a programming language that is one of the core technologies of the World Wide Web, alongside HTML and CSS. Over 97% of websites use JavaScript on the client side for web page behavior, often incorporating third-party libraries.

4- **Jinja Template engine**
   Jinja is a web template engine for the Python programming language. It was created by Armin Ronacher and is licensed under a BSD License. Jinja is like the Django template engine but provides Python-like expressions while ensuring that the templates are evaluated in a sandbox.

In order to develop the front-end of a website, all of the above-mentioned parts shod be considered in the development phase. For **TidalTools** since a jinja template engine is used, html files were developed in a different manner. In **TidalTools** a reference html and CSS was developed. The base html codes were developed for navbar and html body. In these two html files all the dependencies and inheritances from CSS and JavaScript codes were done. Then, different jinja blocks were defined inside the template html files for later interactions with different applications.

Then, for every sub application that **TidalTools** has, different html files were developed but the complexity of these html files were not like that of the main template html because thanks to ninja template engine, all the settings and CSS configurations are inherited from the base template. To develop views on the platform a similar approach was used for almost all of the different pages which is as follows:

First, Models.py is where the database is created then tables are added to admins.py to make them available in the admin panel of the platform. Then if the view should display any forms to get data from user, the forms associated with the database tables are made inside the forms.py.

Then a view function for each one page the application is developed in views.py and bonded with the associated template html file inside the "templates" directory. Finally, these views are connected to a URL inside the urls.py file.

The data to be displayed on the browser is passed by views.py file to the html template through JSON metadata. first the metadata is handled by jinja and all the jinja variables in html file are replaced by the associated values given in the metadata

This approach is how every application of the platform was developed. Then these modules are called inside the main engine of the platform inside the setting.py and directly all the functions, routings, database tables, etc. has been added to the platform.

In this section a general explanation of the code and GUI architecture was given. However, in the following parts of this section more detailed explanation about the modules, database tables and the back-end functionality of each solver is given.

4 Horizontal modules

As mentioned in the previous section, horizontal modules are those that are interacting with other modules and communicate with other vertical and horizontal modules. In TidalTools, five horizontal modules were developed which are:

1- User Management
2- Messaging application
3- Emailing application
4- Data management application
5- Forum application

These modules (Django Applications) are interacting with each other to keep the track of operations and manage that users information and all computational tools (Vertical modules) are connect to them in different manners, as explained in this section and in the Vertical Modules section.

4.1 Users Management Module

Django by default has a user module. However, this module is mainly being over-written or inherited from to write a new one for projects because it takes a limited amount of data. In **TidalTools** platform the approach was to develop its own user module and use the native module as a parent and inherit the methods such as user registry, login, logout, and user authentications from the django user model.

Therefore, a module (django application) was developed to deal with user management using django users module as a parent to this module.

4.1.1 Database Structure

The database which is particularly developed for storing the user data is called "**UserProfile**". This table is connected to the users module from username field of it (foreign key). This inheritance provide the platform with having different user access levels and more which is shown in Figure 9.

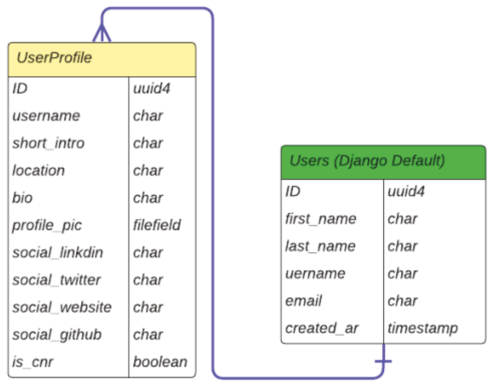

Figure 9 User Module Database entity relationship diagram

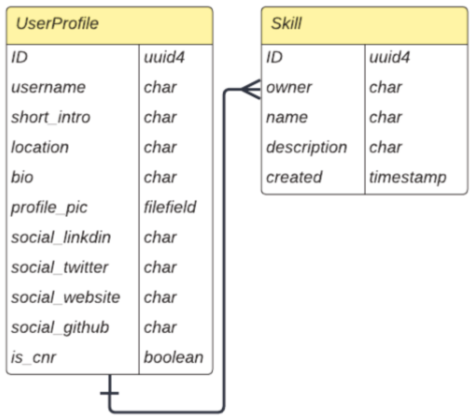

Figure 10 Skills Database entity relationship diagram

All the information regarding users and their skill is stored in the mentioned tables in the database.

4.2 *Messaging application*

This module is developed to provide communication means among the users of the platform. Users can use this feature to exchange ideas and get in touch with the admins and developers of the platform. For this reason, a table in database was developed to store the needed information.

4.2.1 Database Structure

The database structure of the messaging application is demonstrated in Figure 11. This table uses UserProfile as foreign key for sender and receiver fields.

Other filed inside Messaging app database are as follows:

1. **Subject**
   This field is a character field that contains the subject of the message
2. **Message Body**
   This field is text field that contains the main message body of the message
3. **Creation Date**
   This field is date-time field that contains the creation date of the message
4. **Is_read**
   It is a Boolean field that by default is False but if the receiver reads the message becomes true.
5. **Is_deleted_by_sender**
   It is a Boolean field that by default is False but if the sender deletes the message becomes true.
6. **Is_deleted_by_receiver**

It is a Boolean field that by default is False but if the receiver deletes the message becomes true.

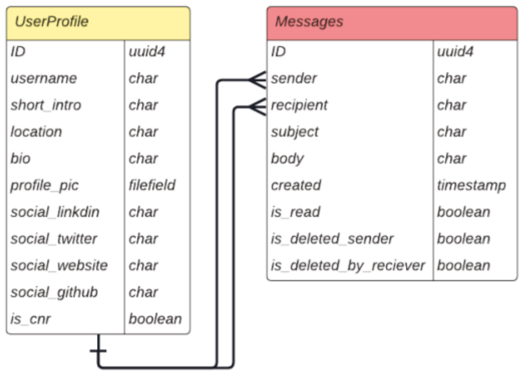

Figure 11 Messaging Module Database entity relationship diagram

4.3 Automatic Emailing Application

This application was developed to automatically send emails to the users based on the operations they perform. This application doesn't have any database tables since it doesn't need to store any sort of data. However, it is connected to the users table because it uses the email address provided by the user at the registration to send them emails. The operations that trigger this application are:

1. Registering on the platform.
2. Receiving a message inside the platform form a user.
3. Successful termination of the Design process.
4. Successful termination of the BIEM code. -

4.4 Data management application

This application is developed the data of the materials that was published by the users after using any tool inside the platform. In other word, if a user for example use X-design to design a turbine and decides to publish the resulting performance information to the community after publishing it the data will referenced in this table so other users can use it to perform other analysis.

4.4.1 Database Structure

The database designed for storing the related data for this application has seven tables. As said before, when user publish a project the output of the project is being added here.

Besides, this application handles the tidal sites data which are used by computational tools. Users can upload site data that they have and use them in the projects. The tables available in this database are shown in Figure 12.

The tables are:

1. **Turbine Data**

This table store the turbine performance data which are generated by X-design tool. More datils are in next subsection.

2. **Site Data**
   This table stores the tidal or oceanic site data. This table is populated with default data and the data provided by the user.

3. **Inflow Velocity Profile**
   This Table stores the velocity profile data.

4. **Geometry File**
   This table stored turbine computational grid data which is generated by AutoBlade tool as input for BIEM. More details are in next subsection.

5. **Polar Data**
   This table stores the blade profile data which is generated by the X-foil application

6. **Radial Data**
   This table stores basic information to build a 3D model of turbine geometry as input for AUTOBLADE. Specifically, radial distribution of blade geometry parameters are given.

7. **Chordwise Data**

This table stores basic information to build a 3D model of turbine geometry as input for AUTOBLADE. Specifically, chordwise distribution of blade sectional offsets are given.

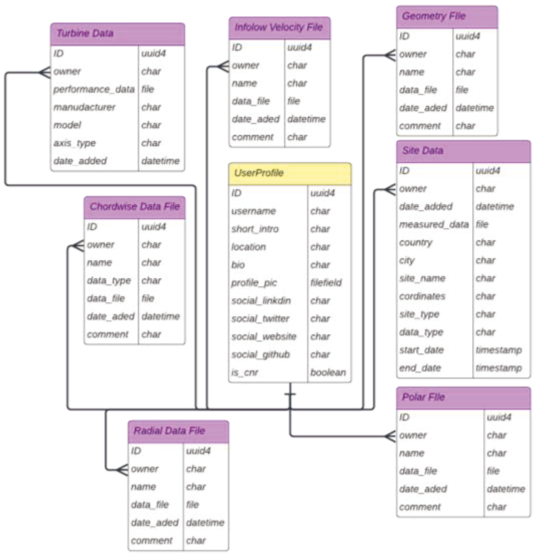

Figure 12 Data Management Database entity relationship diagram

4.5 *Forum Application*

This application is not fully developed yet but it is foreseen inside the platform architecture to provide the means of communication, networking, and knowledge exchange for the users of the platform.

5 Vertical Modules

Vertical module as explained in previous sections is more of a particular module that only deals with specific problems and doesn't interact with other modules to perform a task. In the current version of the **TidalTools** platform there are four computational modules (Django applications) developed as explained in this section. All the vertical modules are interacting with the Users Management Module to organize the users and their projects accordingly.

5.1 Turbine Geometry and Mesh Tool (AutoBlade)

The ***AUTOBLADE*** tool allows to generate a 3D geometry with a fully automated mesh over the blade and hub of a horizontal-axis turbine. This computational tool can be both a part of a turbine design procedure (see below) or can be used solo. As explained in the introduction part, **TidalTools** aims to reduce the complexity of using the computational codes that are linked in it for users who do not have a specific theoretical knowledge. To this aim, a minimal input structure has been designed, whereas input parameters requiring some knowledge of the methodology implemented into the software are not requested.

5.1.1 Database Structure

This Module uses a complex relative database due to its different features for **Expert** of **Basic** user. The term expert in this platform refers to a module access procedure that uses more input data since it is developed for more professional use. However, the basic user type just work with few key input data. Figure 13 shows the entity relationship diagram of the AutoBlade database.

This database also interacts with the Data management application database. This tool uses the data stored in Chord wise data and Radial data from Data management database.

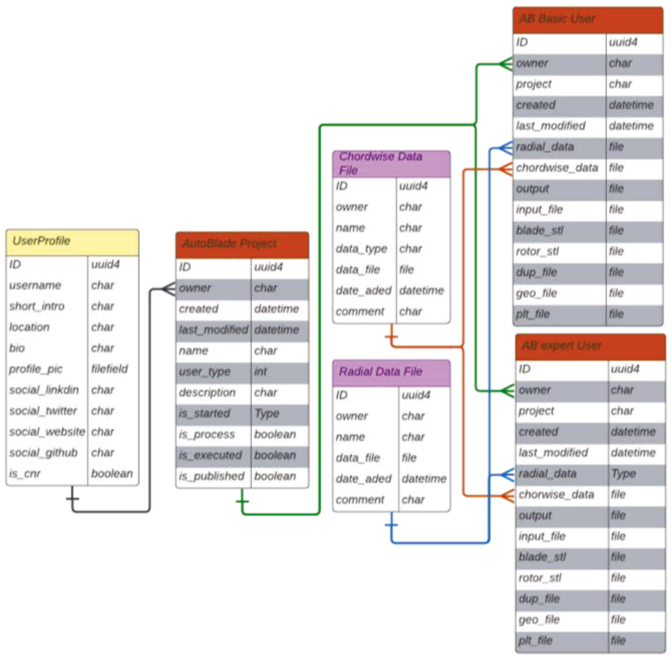

Figure 13 AutoBlade Computational Tool Database Entity Relationship Diagram

5.2 Turbine Performance Simulation (BIEM)

This tool is developed to perform turbine performance simulations for user's defined operating conditions. Results include thrust, torque and power curves associated with the turbine and other quantities describing blade loading, risk of cavitation, induced velocity. This tool like the AutoBlade provides two different types of input data one simplified for Basic users and the other one advanced for professional use. Besides two different input options, this code can also run either for a single operating condition (single TSR value) or for a range of operating conditions (list of TSR values).

5.2.1 Database Structure

The database of this tool consists of 3 tables however it also interacts with the database of the Data management application database and uses data stored inside the following tables:

1. Geometry files data
2. Inflow velocity profile
3. Polar file

The tables inside the database of this module(Django-application) consists of 3 tables which are :

1- BIEM Project
Store the main project data which later other two tables will connect to this through a foreign key.

2- Basic user project
Stores the project with simpler input data

3- Expert user project
Stores the project with more complex input data

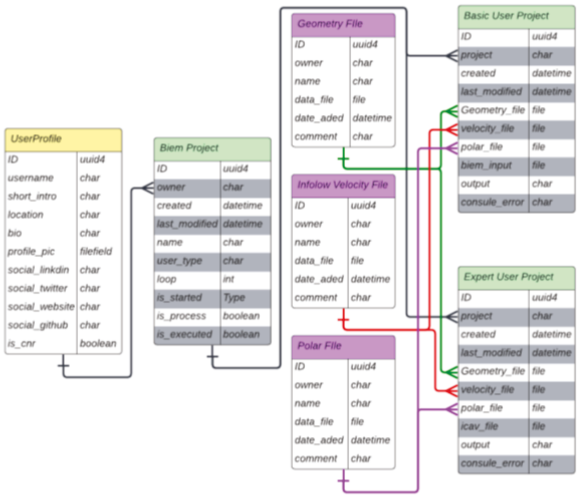

Figure 14 BIEM Computational Tool Database Entity Relationship Diagram

5.3 Turbine Design Tool (X-design)

The design algorithm used in the back-end of the platform is a robust design algorithm developed from scratch in CNR-INM. Like every design algorithm it involves a complex iterative procedure in which multiple solvers interact by exchanging input/output data. In order to adapt design parameters to a wide range of design problems, the design algorithm can manage a complex input data structure (files and parameters). However, as in the case of the AUTOBLADE and BIEM solvers, a simplified input structure is provided to basic users, whereas the parameters that are less case dependent are controlled by a dedicated interface in the back end. This way, a non-expert user can perform a full design of a turbine from scratch.

5.3.1 Database Structure

Since this project can perform the design based on a target power [kW] or a target dimension [rotor diameter] two different tables were developed for storing the input data. Then these two tables are connected to the main project table on the Project field to connect the input data to stablish the link between the project and its input file. The Database Entity Relationship Diagram is shown in Figure 15.

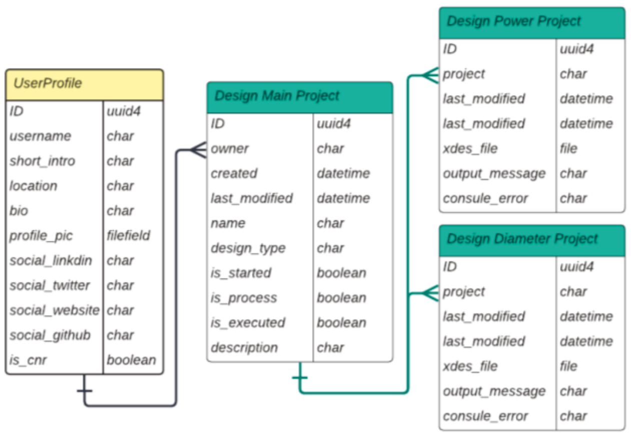

Figure 15 X-Design Tool Database Entity Relationship Diagram

5.4 Turbine Annual Productivity (SiteSim)

This tool is developed to perform the analysis of the turbines annual energy production using the power curve generated by BIEM code and by using the data available in the site resource database of the website. The tool also evaluates a number of turbine mechanical performance parameters during operation in variable speed (Maximum Power Point Tracking, MPPT) and in overspeed (rated power) regimes. However, in the future development plans users can also use their own site resource data to perform these simulations. In this module when user wants to start a project, they have a chance to have as many sub projects as they want for each project. The idea behind this approach was to give users an opportunity to organize their projects based on sites or turbine types. In other words, a project can start for a site and then in sub projects user can evaluate the performance of the different turbines in that site and keep them all in one project or vice versa.

5.4.1 Database Structure

The data base of this module interacts with the Data management module and uses the data stored in Turbine data and Site data tables. Other than that, it has two tables that are project and sub project. Figure 16 shows the Database Entity Relationship Diagram for this module.

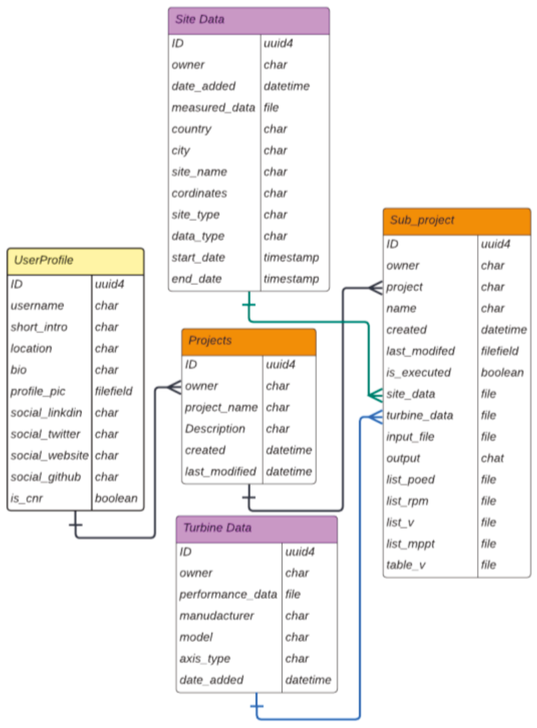

Figure 16 SiteSim Tool Database Entity Relationship Diagram

5.5 *Economic Feasibility Calculation Tool(EcoSim)*

This tool is used to perform the economic feasibility analysis of a turbine in a particular site and generate a report that contains all the costs associated with constructing, operating, and maintaining it. The tool can also provide information like the Levelized Cost of Energy (LCoE), payback period, and other performance indicators that are used to perform a Life-Cycle Assessment (LCA) of a given tidal energy project.

This tool is still at development stage and is not available yet inside the platform, but it will be available soon after validation tests will be completed. However, the platform structure to host this tool has been implemented in the same fashion of all the other computational tools that are already available.

6 Conclusions

In this report, the development and implementation of a computational platform for the analysis and design of tidal energy turbines has been described. The platform is called **TidalTools** and is a multi-user web application (website) which is designed to be accessible by remote connection using a common web browser. It deals with design, performance simulation, economic feasibility, etc. problems in the tidal energy industry and particularly it is designed to perform a horizontal-axis tidal turbine characterization from scratch with some simple inputs. The platform integrates computational tools that have been developed and extensively validated at CNR-INM (formerly INSEAN) over the last two decades. Through the platform, these tools will be made available to users from academia and industry in the tidal energy sector.

The work described in the report is part of a long-term project in which the present platform version represents the first release of a digital tool that in the future can be extended to deal with a wider range of applications in the renewable energy sector.

7 Bibliography

[1] J. Andrews, "Quora," [Online]. Available: https://www.quora.com/How-do-popular-websites-use-multiple-programming-languages-for-their-backend-system-How-do-these-programming-languages-communicate-with-each-other-in-a-single-website.

[2] P. S. Foundation, "Python," [Online]. Available: https://www.python.org.

[3] R. Python. [Online]. Available: https://realpython.com/tutorials/web-dev/.

[4] SQLite, "SQLite," [Online]. Available: https://www.sqlite.org/whentouse.html.

[5] Celery, "Celery," [Online]. Available: https://docs.celeryq.dev/en/stable/getting-started/introduction.html.

[6] P. Documentation, "Pandas," [Online]. Available: https://pandas.pydata.org.

[7] Numpy, "NumPy," [Online]. Available: https://numpy.org.

[8] Matplotlib, "Matplotlib," [Online]. Available: https://matplotlib.org.

[9] Seaborn, "Seaborn," [Online]. Available: https://seaborn.pydata.org.

[10] SFneal, "PyPDF3," [Online]. Available: https://github.com/sfneal/PyPDF3.

[11] https://www.postgresql.org/about/. [Online].