Developing a Website comparing: Django, Wagtail and Wordpress

Filipe João Lucas Inês
Instituto Supeior Técnico, Lisboa, Portugal
June 2022

***Abstract*—The goal of this thesis is to compare multiple solutions (django, django + wagtail, wordpress) to rewrite the website https://www.arctel-cplp.org/. The website belongs to ARCTEL which is a law association that promotes information sharing and exchange of experiences between communication's regulators of the Portuguese speaking countries. The website is used in different ways by the organization and was multiple sections like news, training center, observatory, reports, etc.**

The website has over a decade and although is stable and it still serves well the organization it has some parts of the code that need to be updated. The code is written in PHP and relies on libraries that are no longer maintained and that are not ported to the most recent versions of PHP which is an obstacle to the maintainability and feature development of the website. The database schema was made on an old MySQL version and does not implement some of the modern features of SQL databases. The rewrite of the website needs to take in account the existing features and requirements and needs to port the database and old files to the new website. The rewrite includes redesigning the database and the backend.

***Index Terms*—Django, Wagtail, Wordpress, Web Development**

I. Introduction

The goal of this thesis is to rewrite and develop the ARCTEL website comparing multiple solutions like Django, Wagtail and Wordpress discussing pros and cons of each approach.

The main analyze will be on Django and Wagtail, although some parts and functionalities of the website will have a proof of concept based on Wordpress so the different approaches can be compared.

Django is a mature web framework released in 2005 and is often compared with Ruby on Rails, Laravel and Java Spring. All of these frameworks follow a battery included philosophy, the term battery included means these frameworks comes with a set of tools and libraries required for common use cases out of the box like ORM, middlewares, authentication, i18n, templates engine, etc, and are suitable for the majority of the web applications giving freedom to the programmer to customize most of the behavior of the frameworks.

Right away one of the disadvantages of Django compared with Wordpress is the CMS functionality and the user friendly editor that comes out of the box with it. The user friendly editor is mandatory to manage the website for the non tech people and could be one of the main drawbacks of Django since developing an editor to the level of Wordpress would be not feasible for the development of the new website, to solve this drawback the new website developed with Django will include a popular python CMS called Wagtail. Wagtail is a CMS released in 2014 also developed with Django, this CMS can be used as standalone application or incorporated ininto an existing Django project, Wagtail is used by organizations like NASA [11] and British NHS [12] providing an interface for non tech people manage the website and create content like posts, news, etc.

II. Arctel

A. Organization

ARCTEL is an organization created in 2008 that potentiates the share of information and knowledge between the multiple regulators of CPLP countries (country that speak Portuguese language like Angola, Brasil, Cabo Verde, Guiné Bissau, Guiné Equatorial, Moçambique, Portugal, São Tomé e Príncipe and Timor Leste) and was formed with the collaboration of the multiple telecommunications authorities of each country (INACOM, ANATEL, ANAC, ICGB, INCM, ANACOM, AGER and ARCOM) with the goal to contribute to the development of the telecommunication sector.

The organization organizes conferences and forums that involves the telecommunication entities of each country to discuss technology, ideas and standards of the telecommunication industry. The organization also publish important news about the field.

As the beginning of the organization in 2009 the website https://www.arctel-cplp.org was created to communicate with its audience.

1) Original Tech Stack: The website was created in 2009 based on a small open source framework called SpellBook, the development of this framework started in 2006 with PHP4 (https://code.google.com/archive/p/sbook/source/default/commits), it is a MVC framework that used the PEAR repository to download packages. PEAR was the most popular PHP code repository however nowadays composer [13] became de facto standard package manager composer for PHP.

2) Old Website Structure: From the old website these are the most relevant webpages:

- Contact: Basic contact information.
- News: News about the ARCTEL related fields. The news should be group by year and be managed through backoffice.
- Publications: sections with publications about the sector.

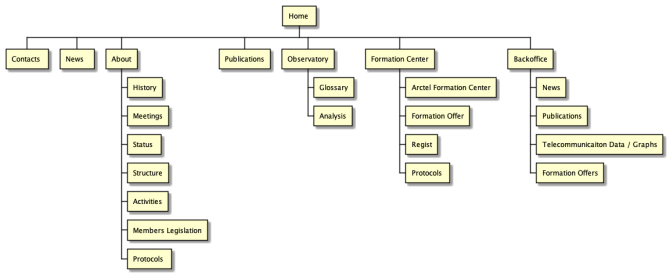

Fig. 1. Old Website Map

- About: info about the organization with multiple subsections (History, Activities, Structure, etc. . .).
- Observatory > Glossary: Collections of terms used in the field.
- Qualification Center > ARCTEL Qualification Center: brief explanation about qualification offers.
- Qualification Center > Protocols: ARCTEL academic partners.

B. Requirements

The people managing the website do not have a tech background so the backoffice should be friendly to non tech people, preferably the website should have a WYSIWYG editor and a CMS system to manage the website Pages and News system. None of the pages should be hardcoded like About, History, Glossary and other pages that are considered static or change not very often, the admin should always have the possibility to change the content and be independent of the programmer even if the pages change once every year or so.

News: The News system should allow the admin to create a New with basic fields like title, date and body, then the News should appear in a listing page grouped by year and with a pagination system. It is also important the website imports the News from the old database because since 2009 the old website had more than 9000 News.

Publication and Directories: The Publication and Directory systems are related to each other and should allow the website admin to upload files and list them in a specific webpage. The Publication is the entity that stores the file, and the Directory is the entity that groups multiple Publications (one Directory to multiple Publications relationship), then each Publication appears on a specific webpage based on which Directory it belongs. The reason this system exists is because Arctel uses the website to publish PDFs reports like activities planned for the each year, accounting reports, telecommunication sector publications, etc, across the website, so as there are dozens of PDFs it is important to have a dedicated and centralized management for these files.

Qualification Center: Arctel organizes qualification offers related to the telecommunication sector together with the partners of each country and uses the website to publish them. Each qualification offer has a dedicated page with a description and a form to register. The qualification offers have a start and end date for enrollments and preferably the website should show or hide them automatically based on their enrollment period without need manual intervention from the admin. The backend renders a qualification offer by checking if the current date is between the start date and end date of the enrollments period.

Telecommunication Metric Graphs: There is also a section with multiple multiple metrics related with technology and telecommunications to show the reality of the sector in different countries. The metrics and data are shown in graphs, in the old website the graphs were manage by an external application and imported as html snippets into the page, the new website should have preferably this system integrated in the database and not rely on external services.

C. Data Models

The database schema from the Arctel old website was analyzed and adapted, some of the tables were dropped since they were no longer used and some are integrated into the database like the data_points table to store the metrics data shown on the graphs. Below there is a list of the multiple entities that should exist on the website:

- users - entity responsible for login credentials so users can access the backoffice to manage the website and edit other entities
- news - entity to store News
- publications - entities that store information and files to be displayed on webpages
- directories - directories group publications
- qualification_offer - qualification offers by Arctel, each qualification offer should have a start and an end date where registrations are open
- qualification_registry - stores the registrations for each qualification offer
- data_points - store data and metrics about the telecommunication sector

III. Solutions

The following 3 technologies alternatives Django, Wagtail and Wordpress were analyzed to create the website. This section explains the basic functionalities and features of each one. It is also discussed how each framework uses the database, how it create the pages, how flexible and easy it is to modify and extend with other libraries.

A. Django

1) Framework Presentation: Django [7] is a python web framework, it has been around since 2005 and it's one of the most popular web frameworks out there, it is "batteries-included" meaning it has built in the most basic features a web application needs, like authentication, ORM, URL mapping, template engine, etc.

This framework is often compared to similar frameworks like Ruby on Rails, Laravel and Spring, which have all similar concepts and similar ways of working. These frameworks are often referred as MCV frameworks (Model, Controller, View), and basically, they all give a way to define URL's that are then

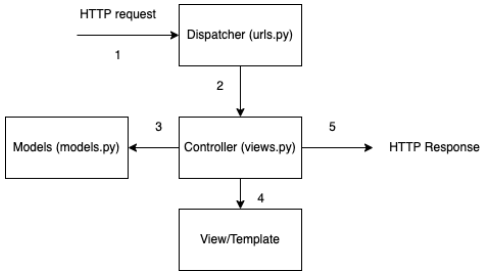

Fig. 2. Django Workflow

associated with a controller that calls data models and execute logic which then uses data to render templates.

Django is basically a software constituted by multiple libraries that receive http requests and return http responses. As already said Django is a MVC framework and the basic flow 2 is receiving the request, pass it to the url dispatcher, getting the controller, ask data from models and then call the view to return the response.

Views/Controllers: The Views (controllers) in Django are responsible to receive web requests and return a web response, this web response can be a string, a html document, a json response, a redirect or any other valid web response. The View is usually the place where the business logic of the application and the data models are called, in the end the View returns a web response by calling a template or just returning data in some format (ex: json, xml). In case of big projects the access to data models and business logic should be behind service layers instead of being directly on the View.

In the project all controllers inherit a custom class based view that can convert POST requests to DELETE or PUT requests, this functionality is a workaround to the fact that is not possible to send DELETE or PUT requests directly from a form submission, so to simulate this methods the forms have a hidden input (example: <input type="hidden" name="_method" value="DELETE"/>) so the POST request is transformed to a PUT or DELETE request in the dispatcher.

Templates: Django templates correspond to the second letter of the MVC term (View) and are a way to generate HTML dynamically. Django templates are only meant to express presentation and do not allow an easy way to execute business logic inside the templates, all business logic should come from the controller, although Django templates allow common "if" and "loop" statements. Besides "if" and "loop" statements the templates can print variables, run tags that contain logic and can for example print output content, as an example the "% csrf_token %" tag prints the csrf token to be used inside forms and validate the POST submissions, there is also filters that transform variables, as an example in this project there is a custom filter called "dict_item" that allows access items from a map inside templates since Django templates do not have a way to access values from the map, this tag can be used as "mapldict_item:'key'".

Django templates allow the reuse of html across files by extending templates, this is useful for example to reuse html that is common to all pages like navbar and footer without the need to rewrite these on all templates. Usually the developer creates a base layout that have common html code to all pages that then extended by other templates.

It is possible to use other engine templates like Jinja2 [6], this template engine is commonly used in Django applications since it is compatible with Django templates and has useful features that Django templates lack, as an example Jinja2 templates allow call methods and functions with arguments and also allows access map items using keys.

Models: Django Models are used to manage the database creating and modifying tables, the data models also store and retrieve data from the database. This models are classes and create the database tables based on the class fields the developer defined.

The models act as source of true, so when the fields are modified the Django application creates database migrations to change the corresponding table. It is possible to create the migrations locally on the dev environment to test, and then apply them on production with ease. The migrations that the database needs to run are stored as files in "migrations" folder and django creates a database table called "django_migrations" to record which migrations were already run in the database.

Scalability: Django is very scalable by having a "shared nothing" architecture, meaning that the application can be developed to be independent of the node that runs on. The fact Django applications do not depend and do not store state on the node they run on, allows Django applications to scale by just adding nodes to a cluster that serves the requests. However the developer needs to actively develop the application to not save state on the node if it is necessary to achieve the "share nothing" architecture and run the application in multiple nodes. To not save state on node the developer can not save files on the disk, this means not saving uploaded files in the file system or not using sqlite as database. Django provides modular and flexible libraries so it is easy to create a sane development environment without compromise the "shared nothing" architecture. As an example a sqlite database can be used locally and changed to a database like postgres or mysql on production by changing the database configuration for dev and prod.

Similar to database configuration, the storage backend (that can be used to store uploaded files) can also be tuned for development and production, as an example the storage backend can use the filesystem for local development and an aws s3 or ftp backend.

There are big companies using Django with huge traffic loads into their applications. The following real world uses show well how Django can scale to handle huge traffic loads:

- Instagram [8]
- Disqus [9]
- Mozilla [10]

B. Django and Wagtail

Wagtail is a CMS written in Python and based on Django, it was created in 2014 by Torchbox [14]. Wagtail can be used as a standalone application or can be integrated in an existing Django project, this thesis talks about Wagtail in the perspective of integrate it into an existing Django project.

The reason Wagtail is integrated in the project is because Django does not provide CMS functionality to create and manage pages on the site, besides the pre defined pages that are populated dynamically by Django ORM models there is no way for the admin to edit or create new pages. It could be possible to create a system from scratch that allowed the administrator to creates pages just with Django, but that would be complex and not on pair with existing CMS systems.

The News system on the website uses a rich editor called TinyMCE [5], the TinyMCE editor is a rich text WYSIWYG HTML editor that provides a simple way to write html, and while it is suitable to create news by providing basic blocks like heading, list, bold text, url, etc is not the ideal solution to create new website pages, since there is no trivial way to create more complex structures.

Creating a powerful editor to non tech people manage website pages is a complex task, as so the Wagtail seem to fit this problem by providing a rich editor and a CMS system to edit the websites pages.

To integrate Wagtail in the existing Django project it is needed to define Wagtail Models to create Pages, but thats basically the only work there is to do in order to get a powerful editor and CMS system trough Wagtail admin section.

There are well known organizations using wagtail like nasa jet propulsion laboratory https://www.jpl.nasa.gov/ [11] and the National Health Service in UK. The NHS UK [12], serves more than 380 millions visits a year and went through a huge migration of the content to the new Wagtail CMS system, ending up migrating more than 4000 pages from the old system.

C. Wordpress

1) Framework Presentation:: Wordpress [4] is a CMS written in PHP created in 2003 for blog systems. Although wordpress was created in mind for blogs, is nowadays used for many different multipurpose websites like for example ecommerce (with the popular Woocommerce plugin) or news (ex: https://www.observador.pt uses wordpress [3]).

Wordpress powers 42.9% of all web [1] being the most popular CMS and tool to create websites. This popularity was achieved because it was one of the first CMS system available that allowed people to create and maintain a website without need to code. Wordpress has a powerful admin interface that allows users to manage, create and edit pages.

2) Admin Dashboard:: The admin interface has a section to create pages and other section to create posts which are basic functionalities for a web blog (which was the primary reason wordpress was created). The admin dashboard has an editor so users can edit the pages and posts without the need to code. Until wordpress version 5 the integrated editor used was TinyMCE editor, however with the wordpress version 5 in 2018 the new editor Gutenberg took place giving a new and more visual way to edit pages with blocks.

The admin interface also has the templates section that allows users to download wordpress themes to change the website visual without coding. The plugin section is another important section from the admin interface, it allows users to download plugins to add or extend wordpress functionalities.

3) Templates: The templates system provides the structure by which the content is displayed, by getting information from the database and render it in html.

The basic wordpress page structure consists of 3 essential blocks, the header, the content, and the footer. The most basic wordpress page is done by creating a file "index.php" with the following content:

```
<?php get_header(); ?>
<?php get_the_content(); ?>
<?php get_footer(); ?>
```

The "<?php get_header(); ?>" and the "<?php get_footer(); ?>" will call respectively the template header.php and footer.php and the "<?php get_the_content(); ?>" will render the information from the database.

Templates are stored in the folder "templates" inside the theme folder. Wordpress follows a set of rules to decide which template to render based on the request.

- homepage: it renders home.php, if this file does not exist it renders index.php
- front page: it renders front-page.php and it is used to display the website front end page, this templates takes precedence over home.php
- single post: single post is used to render posts and it renders the following templates by priority:
  - single-post-type-slug.php
  - single-post-type.php
  - single.php
  - singular.php
  - index.php
- single page: to render static pages wordpress follows these templates by priority:
  - page-slug.php
  - page-id.php
  - page.php
  - singular.php
  - index.php
- category: the category pages get all posts assign to the category and get the following templates by priority:
  - category-slug.php
  - category-id.php
  - category.php
  - archive.php
  - index.php

One of the most basic concepts of Wordpress to display content is called "The Loop", the loop goes through all posts to render in the templates. As an example if the user is on a category.php which is called when Posts are being filtered by category, the following code could be used to list the Posts by turning the title of each one into links for the single post template:

```
<?php
// The Loop
while ( have_posts() ) : the_post(); ?>
<li>
  <a href="<?php the_permalink() ?>" >
```

```
    <?php the_title(); ?>
  </a>
</li>

<?php endwhile; ?>
</ul>
```

In a single page the loop would be similar to this code to render the entire Post:

```
<?php

// The Loop
while ( have_posts() ) : the_post(); ?>
  <h1> <?php the_title(); ?> </h1>
  <div> <?php the_content(); ?> </div>

<?php endwhile; ?>
```

IV. Implementations

A. Django

1) Implemented functionalities: In the Django solution the following requirements are implemented: News, Publications and Directories, Qualification Offers and Graphs. The requirement to create and edit new webpages is not implemented since this requirement is the use case for the Wagtail integration.

News: The News system allows the website to publish News about the telecommunication field, the system allows the administrator to submit news with title, date, thumbnail and body through the admin interface,

The News are then presented in the News section in the website grouped by year. The website administrator can access the backoffice to list the News where they can be edited, deleted or created.

To edit News the administrator has access to an editor. The editor in used is TinyMCE that provides a WYSIWYG (what you see is what you get) editor that converts the user input to HTML, this HTML input is then stored on the database.

Publications: The old website had a Publication feature that allowed the administrator to upload a file and associate it with a Directory to publish the file in a specific part of the website. According with the Directory that the Publication belonged it appeared in a specific page. The website has 6 type of Directories:

- Publicações do Setor
- Anuário das Comunicações
- Plano de Atividades
- Relatório de Atividades
- Relatórios e Contas

As an example the Publication items that belong to the Directory "Plano de Atividades" will appear in a list within the page "/atividades". Then each "Plano de Atividades" item list is a link to a PDF that represents the Activity Plans report for a given year.

Qualification Offers: Arctel organizes qualification offers related with the telecommunication field and uses the website to publish and advertise them. Each qualification offer has a title, description, a start date and an end date. The website administrator has a backoffice section in which is possible to create, edit and delete qualification offers.

The list of qualification offers will then be listed automatically to public if the current date is between the start_date and end_date, with a button for registration. Upon a submitted registration a QualificationRegistration is stored in a database with the data of the submitted form that has the information about the enrolled person. The administrator can then see the list of registration in backoffice.

Graphs: The old website had a section called observatory where it was presented graphs showing some telecommunication metrics over the years for different countries, the old website did not had the data integrated in it, the data was external and the graphs where imported as html snippets.

In this implementation the data is integrated in the website database.The graphs can show the Django capabilities to work with data, from data modelling to creating api to access the data.

All graphs have a similar data structure with a value (number), country and year, so the DataPoint model was created with those fields to store the data.

The graph is made using ApexChartsJs [2] library, and it is populated by a javascript fetch call to an endpoint created in Django to serve the data in json format.

2) Implemented Data Models: The implemented data models followed a Domain Driven Design approach (more commonly referred as DDD), this concept is a software design approach that focus modelling a software system to match the domain.

As an example, if the website has News there should be a class called News with the attributes of a new like title, date, text, etc; or if the website has qualification offers section there should be a class called QualificationOffer that represents the qualification offers and some class called QualificationSubmission that represents the submission to enroll in the qualification offer.

To create and interact with the database the project is using the Django ORM, so for each table/entity there is a corresponding python class that extends "django.db.models.Model", after defining each data model class the framework will then create the tables and do migrations if any field is then added, deleted or modified.

For the project the following models can be defined:

- New: it has a title, text, description, date and a image field for the thumbnail 1
- Directory: it has the purpose to group Publications into directories 2
- Publication: each publication stores files, and based on which Directory it belongs it will be presented in different pages 3
- QualificationOffer: it describes a qualification offer, it contains a title, description, text, thumbnail image 4
- QualificationSubmission: it stores the submission for each qualification offer that comes from a form submission 5
- DataPoint: it stores data to be used by graphs 7
- Country: it stores country names and codes to be used by DataItem model 6

Listing 1. Directory class data model
```
class New(models.Model):
  title = models.CharField()
  text = models.TextField()
  description = models.TextField()
  date = models.DateField(default=now)
  image = models.TextField()
```

Listing 2. Directory class data model
```
class Directory(models.Model):
  identifier = models.CharField()
  name = models.CharField()
```

Listing 3. Publication class data model
```
class Publication(models.Model):
  title = models.CharField()
  file = models.TextField()
  image = models.TextField()
  directory = models.ForeignKey()
  order = models.IntegerField()
```

Listing 4. QualificationOffer class data model
```
class QualificationOffer(models.Model):
  title = models.CharField()
  text = models.TextField()
  description = models.TextField()
  image = models.TextField()
  start_date = models.DateField(default=now)
  end_date = models.DateField()
```

Listing 5. QualificationSubmission class data model
```
class QualificationRegistration(models.Model):
  name = models.TextField()
  email = models.TextField()
  text = models.TextField()
  date = models.DateField(default=now)
```

Listing 6. Country class data model
```
class Country(models.Model):
  label = models.CharField()
  code = models.CharField()

  def __str__(self):
      return self.label
```

Listing 7. Data point class data model
```
class DataPoint(models.Model):
  country = models.ForeignKey(Country)
  year = models.IntegerField()
  value = models.IntegerField()
  graph = models.CharField()

class Meta:
  unique_together = (
      'country',
      'year',
      'graph')
```

3) URLs and Controllers: The developed Django solution follows a REST resource naming convention often used to create URLs, for this convection we use the concept of "resource", normally each resource corresponds to an entity (ex: database objects) and is identified by an unique id (usually the database primary key). The backend provides a set or URL to interact with the resources using the basic HTTP methods (GET, POST, DELETE, PUT) with the following conventions:

- "/resource" - GET: gets the collection of a resource, usually this endpoint get multiple instance of a resource with pagination functionalities.
- "/resource" - POST: submit a HTTP POST method with data to create an instance of type resource
- "/resource/<id>" - GET: get the resource with the identifier "id"
- "/resource/<id>" - DELETE: delete the resource with the identifier "id"
- "/resource/<id>" - PUT: modify the resource with the identifier "id"

In this project we can define 5 resources: News, Publications, Qualification Offers, Qualification Submissions, and Data Points. As an example the URL endpoints for the backoffice News to list, create, update and delete NEWS are the following:

- "/backoffice/news" - GET: get all NEWS
- "/backoffice/news" - POST: submit a HTTP POST with data to create a New
- "/backoffice/news/<id>" - GET: get the resource with the identifier "id"
- "/backoffice/news/<id>" - DELETE: delete the resource with the identifier "id"
- "/backoffice/news/<id>" - PUT: modify the resource with the identifier "id"

As the forms do not provide PUT or DELETE methods the project uses a common workaround that implies having a hidden field in the form 8 usually with name "_method" that has the value DELETE or PUT that will then be catch by the dispatcher and turn into the corresponding method.

Listing 8. Form hidden field to turn POST request in DELETE or PUT
```
<form action="/" method="POST">
  <input type="hidden" name="_method"
    value="DELETE">
  <input type="submit" value="delete">
</form>
```

The REST resource naming convection is used in all resources on the project.

V. Comparing the solutions

Each solution has pros and cons, by analyzing the data models, flexibility, extensibility, the ease of use for the end user and the ecosystem that surrounds the technology and framework.

A. Data Models

The data modeling in Django is far superior to Wordpress. In django the data models are defined as python classes through

Django ORM providing automatic database management creating and migrating tables. In the project the following tables were created without counting the tables created by Django itself and Wagtail:

- New
- Directory
- Publication
- QualificationOffer
- QualificationSubmission
- DataPoint
- Country

This flexibility to interact with the database makes the data maintainable and easy to work it. Even in cases of integrations or data migrations is useful do have well structure data, as an example more than 9000 news from the old website were migrated to the Django project.

In wordpress by default all content is stored in the posts table. It is possible to create tables by the plugin system, although each plugin can manage the database in different ways and so different plugins do not interact with data from each other, opposed to Django were the ORM brings a standard way to manage the database and it is possible to interact with data from different applications, as an example the Django project uses the data models from the Wagtail library.

The advantage of wordpress in this subject is that manages the database and creates the default data schema without any need of coding, while in Django every data model needs to be coded.

B. Flexibility and Extensibility

Django gives great flexibility by providing tools to create complex web apps, the programmer can easily do complex database schemas, define how each url should like and assign different templates for each controller. The flexibility brings the possibility to extend the project with all available Python libraries for different purposes like Machine Learning, data/images/files/pdfs/execel manipulation, web scraping, etc, since each controller is a python function/method it can easily contain external libraries. Django also allows to use multiple types of databases like sqlite, mysql or postgres, multiple storage solutions like ftp, aws s3 and local storage and allows the use of different cache solutions like redis or memcache that can be choose taking into account the project requirements and scalability.

Wordpress is flexible in term of a blogging platform, it is possible to create pages and posts that can be grouped by category and tags, and offers great front end customization with the Gutenberg editor. Wordpress does not support multiple databases besides mysql which can limit the project scope if it needs another database. Wordpress is extensible with the plugin system although is not meant to add endpoints and work with the database as Django is. In fact Wordpress as a basic api to access the database but this api involves writing raw sql queries which can become messy overtime, however each developer can decide to incorporate php libraries in the plugins to integrate better tooling like for example Doctrine ORM to interact with the database or Symfony HTTPFoundation to handle requests.

C. Easy of use for the end user

Wordpress is far superior in terms of offering a friendly environment to manage the website for non tech users, this is one of the focus of wordpress, it is being developed with the goal of giving the possibility of the user to create and manage a website without the need of coding.

Django also provides an adnin interface to manage the data models, however it is not user friendly enough for non tech users, what usually happens is that the developer does a custom admin interface for the user that is easier to use for non tech people. The Wagtail provides a better interface to manage pages and content however it is still not so good as wordpress in this aspect.

D. Pros and Cons of each tech stack

Below is a summary how well the multiple solutions implement the system requirements for the website.

1) Django - Requirements Analysis:

- News System: The News system integrates well, a data model can be defined with the fields (title, date, body). It needs more work than Wordpress because the data model, templates and editor need to be coded
- Publication and Directories: Integrates well with Django, Publication and Directories have each one their data model and it is easy to create the relationship of one to many between the entities.
- Qualification Center: Good use case, the qualification center has well defined fields (title, description, start date and end date), with one model is possible to control the web page. The web page renders the qualification offers automatically publishing or hiding them by comparing the current date with the enrollment period (between start and end date). The data model QualificationOffer also controls the template that renders the detail page for one qualification offer that has the description and the page with the form for the enrollment. So one data model controls two templates, avoiding the need to create two web pages individually
- Telecommunication Metric Graphs: Really good use case for Django, the framework excel at creating data models, which is an important factor when working with data analysis and graphs. The endpoints the graphs use to get the data are also trivial to create in Django

2) Wagtail - Requirements Analysis:

- News System: Handles well the News system by providing a rich editor and CMS system to manage images and files with little code
- Publication and Directories: No the best use case, although the web pages containing publications could be done creating a page with a list of links pointing to pdf's (emulating Publications)
- Qualification Center: Not a good use case, this system is handled better by Django

- Telecommunication Metric Graphs: Use case handled by Django, does not make sense to integrate this system in Wagtail

3) Wordpress - Requirements Analysis:

- News System: Good use case, no code needed, the News are Posts assign to a category (ex: new_category) and a Page can be created to list the News already with pagination, filtering the Posts by the category assign to News Posts (ex: new_category) . All this just using the Gutenberg editor
- Publication and Directories: Similar to Wagtail, a Page can be created to publish the Publications as link to files. Wordpress media is not great to organize files, there is not a native way to group files in albums and folders, so eventually it would be hard to manage dozens of publications, unless a plugin is used to give Wordpress the ability to create folder in the media subsection
- Qualification Center: Qualification Offers can be created as Posts assigned to category "qualification_offers" then a Page would render a list of posts filter by that category. The submission form can be handled by installing a plugin to create forms and the Qualification Offer Post could be removed automatically with the plugin Post Expirator than can add an expire date so a Post can be unpublished automatically after the enrollment end date
- Telecommunication Metric Graphs: Not trivial to do in Wordpress, this system could be implemented by developing a plugin to create the database table, the endpoint to serve the data and the admin subsection in the dashboard to insert data. Wordpress is not the best to work with data analysis, Django is much better with data since it is easy to create and manage data models.

The technology by itself is not the only reason to pick a tech stack, all the ecosystem around the technology has a big impact when deciding what to use. Below there is a summary showing the pros and cons about multiple topics comparing Django, Wagtail and Wordpress.

4) Django - Ecosystem analysis:

- Cost: It definitely costs more to develop in Django than in Wordpress (unless the project is complex, in this case developing in Wordpress can become more time consuming), Django is used mostly by tech companies to develop complex web applications or used by companies that rely heavily on a web application.
- Ease of use for non tech people: It depends totally on the development of a dashboard admin interface, it can take a good amount of time to code a nice interface for the website management. For tech people Django has natively a admin dashboard to manage the websites and data models that is very good but only usable by programmers.
- Documentation and Resources: Django is one of the most popular web frameworks and is used by many companies (ex: instagram). The documentation is good and there is a lot of information and resources about the framework.
- Hosting: Django is easy to host, however it takes some linux management knowledge to put it online using Nginx and Gunicorn, it is also needed to manage a database, certificates and if the project uses cache it is also necessary to manage a redis or memcache server. Although there are some companies that automate this process creating deployment services that abstract all of this management (ex: Heroku)
- Popularity: Very popular among programmers, it is relatively easy to find programmers to create and maintain Django projects.

5) Wagtail - Ecosystem analysis:

- Cost: Similar to Django, if the Django app needs a CMS system is still less time consuming to integrate Wagtail, even if the developers need to learn it first, than implementing a CMS system in Django
- Ease of use for non tech people: Wagtail has a nice interface to manage the website by providing a good editor and great CMS capabilities.
- Documentation and Resources: The documentation is not so good and it is not also so popular as Django or Wordpress, finding answer to a particular problem can be hard, although it is gaining traction and is used and sponsor by big companies like Google [**?**]
- Hosting: Same as Django since Wagtail is itself a Django application
- Popularity: The less popular of the 3, although Wagtail is itself a Django application so it is somehow similar, but it is still required to study the framework reading the docs which can be time consuming.

6) Wordpress - Ecosystem analysis:

- Cost: Very safe choice in terms of costs for small projects, for some cases there is not even the need to code.
- Ease of use for non tech people: The best of the 3, Wordpress has been developed to be easy to be used by no tech people, by having a admin dashboard and a great editor (Gutenberg)
- Documentation and Resources: The most popular technology to make websites, 42% of the websites online use Wordpress. Documentation is good and it is extremely easy to find resources
- Hosting: Extremely easy to deploy, Wordpress uses the popular LAMP stack (Linux, Apache, MySql and PHP), it is easy to setup and there are multiple resources online explaining how to deploy and manage the stack. There are also multiple cheap hosting services for Wordpress
- Popularity: Extremely popular, very easy to find developers and users that can work with Wordpress

VI. Conclusion

From the work developed in this thesis we can check the pros and cons of each solution. The biggest advantage of Wordpress is the fact that is possible to develop a website without the need to code. Although for complex websites it needs integrations with the wordpress api which can be hard to maintain since the Wordpress api did not suffer many substantial refactors over the years, one reason for this is because Wordpress compromises to be compatible with old PHP version (from version PHP5.6), and so it does not

use the most modern features of php, as an example the Options API https://developer.wordpress.org/plugins/settings/options-api/ and Settings API https://developer.wordpress.org/plugins/settings/settings-api/ are all function based, which it is not bad by itself but in some cases object oriented programming approach can help making the code more reusable and decoupled with inheritance and interfaces.

Wordpress is suitable for websites that do not need to do complex data management or create api's, like blog platforms, business websites, portfolios or personal webpages.

Django excels at creating flexible websites that can scale to handle huge traffic loads, it is suitable for websites that need well structure data models, api's and complex logic. It gives the freedom to choose from multiple database and cache solutions, however it needs much more coding compared to Wordpress, not being a viable solution for companies that do not have focus on web technologies and just need a simple website.

The Django solution by default does not give an easy way to manage the website for non tech users, however as the framework is flexible this can be solved by integrating a full feature CMS like Wagtail that provides a way for non tech people to manage the website with a friendly interface.

For the Arctel use case, choosing Wordpress is an understandable choice since the company is not a software company with an huge complex website. However in the last years the organization has been having initiatives for more complex use cases for the website like for example the graphs that show metrics about the telecommunication field, with wordpress the graphs were done by inserting external html snippets taken form a external web application. The adoption of wordpress can be limiting in what the website can do in the future and be a roadblock for initiatives that require complex website features. A solution with Django + Wagtail can be the most sane tech stack for companies that need an easy interface to manage the website and the need to do more complex features.

REFERENCES

[1] 42.9% of the websites use Wordpress - https://w3techs.com/technologies/details/cm-wordpress
[2] Apex charts website- https://w3techs.com/technologies/details/cm-wordpress
[3] Observador uses Wordpress - https://wp-portugal.com/2014/11/26/observador-caso-estudo-wordpress-todo-mundo/
[4] Wordpress website - https://wordpress.org/
[5] TinyMCE editor website - https://www.tiny.cloud/
[6] Jinja2 templates website https://jinja.palletsprojects.com/en/3.1.x/
[7] Django Website - https://www.djangoproject.com/
[8] Django in Instagram - https://instagram-engineering.com/types-for-python-http-apis-an-instagram-story-d3c3a207fdb7
[9] Django in Disqus - https://instagram-engineering.com/types-for-python-http-apis-an-instagram-story-d3c3a207fdb7
[10] Mozilla uses Django - https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/Introduction
[11] NASA's Jet Propulsion Laboratory uses Wagtail CMS - https://torchbox.com/blog/nasa-jpl-launches-on-wagtail/
[12] British NHS migration to Wagtail - https://digital.nhs.uk/blog/transformation-blog/2018/nhs.uk-content-migration-update
[13] Compose website - https://getcomposer.org/
[14] TorchBox website - https://jinja.palletsprojects.com/en/3.1.x/