Trees Everywhere Documentation

Version 1.0.0

João Vitor Antoniassi Segantin

February 11, 2023

Contents

List of Code Listings

```
function X = BitXorMatrix(A,B)
%function to compute the sum without charge of two vectors

        %convert elements into usigned integers
        A = uint8(A);
        B = uint8(B);

        m1 = length(A);
        m2 = length(B);
        X = uint8(zeros(m1, m2));
        for n1=1:m1
                for n2=1:m2
                        X(n1, n2) = bitxor(A(n1), B(n2));
                end
        end
```

Listing 1: Example from external file

1 Introduction

This document is a software documentation created using LaTeX and Overleaf. It provides an overview of what was made, where and why.

Some examples in this document includes mathematical equations, code listings, charts and graphs to better visualize code complexities and decisions made.

2 Project Architecture

Code can be printed using the minted or listings packages, or several other tools. Listing **??** shows an example of typesetting code from an external file.

Alternatively, code can be written directly in your .tex files, as in Listing **??**. It's also possible to typeset syntax-highlighted code inline with a number of code listing packages. Check the documentation for the package you're using for more details. For example, section 3.3 of the Minted package documentation[1] gives some examples and guidelines, as does our help article on code highlighting with minted[2]. Forums such as TeX StackExchange[3] and

[1] `http://texdoc.net/pkg/minted`
[2] `https://www.overleaf.com/learn/latex/Code_Highlighting_with_minted`
[3] `https://tex.stackexchange.com/`

```python
print("Hello World")
```

Listing 2: Example Python code

LaTeX Community[4] are also a great source of tips.

3 Some further examples to get started

3.1 Directory structure

Many Django projects are built using a common set of patterns for its directory structure. Some programmers prefer to use several *django apps* to better organize their code, while others prefer to use a single directory for all applications. Both approaches are valid, and they offer advantages and drawbacks.

3.1.1 Multiple django apps

Django apps are a way to organize your code into reusable components, they can have its own set of models, views, urls, admins, etc. This is a big advantage on large projects, where you can have several apps, each one with its own purpose, and you can reuse them as they were designed for.

The hardest thing about using multiple apps is to keep track of all the dependencies between them, and to make sure that you don't have circular dependencies. For instance, if you have an app called *users*, and another app called *trees*, if you wish to keep track of the trees that each user has planted, the easiest way is to import the *users* app into the *trees* app, which can potentially create a circular dependency. I think that the best way to avoid this is to keep your apps as independent as possible, avoiding foreign keys between them.

In the case we saw is hard to say they are really *reusable components*, since they are not really independent of each other. It is possible to solve this problem by creating messaging queues to handle the communication between them and making the models *denormalized*, but this would be unnecessarily complex and susceptible to errors.

The organization of the Django app can influence the way you write your code, and how you design the database tables, procedures, views, etc. This makes models to be the most critical part of an app, as they are the ones

[4] `https://latex.org/forum/`

that are used to organize data, offer facilities in order to manage the data with custom methods and the list goes on.

Database is the most important part of any application, and its design have a huge impact on the performance and maintenance. It is easier migrate all the applications to another language or framework, than to migrate a database structure. It's easy to cause data loss when migrating a database, and it's hard to keep track of all the changes that were made during the migration process.

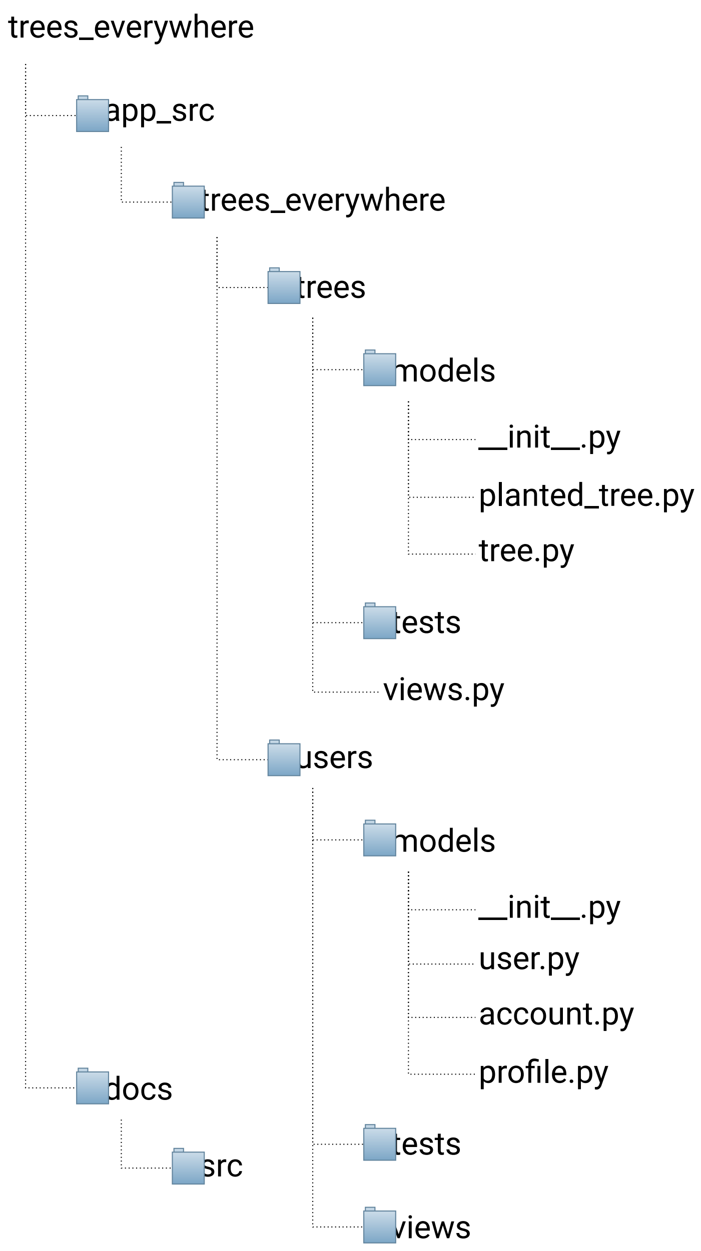
3.1.2 Single django app

The single directory approach is the simplest one, and it is the one used in this project. Of course this approach has its own drawbacks and advantages.

If not properly organized, the code can become a mess, and it can be hard to find what you are looking for. This is why it is important to have a good directory structure, and to keep it organized.

In order to avoid the code to become a mess, I decided to create the models, views, urls. . . in splitted directories inside the application. This makes easier to work in the application thinking in namespaces, and it makes easier to find what you are looking for.

The directory structure used in this project is the following:



3.2 Password Security

Passwords cannot be stored in plain text, because if someone gets access to the database, they can easily get access to sensitive information. This can be avoided by using an algorithm that transforms the password into a hash, and then store the hash in the database.

The algorithm used by django by default is called *pbkdf2_sha256*, and it is

Figure 1: This frog was uploaded via the file-tree menu.

Item	Quantity
Widgets	42
Gadgets	13

Table 1: An example table.

quite hard to break, recommended by NIST (since 2010 https://nvlpubs.nist.gov/nistpubs/Legacy/
132.pdf) but it can be better.

The algorithm used in this project is called *Argon2*, the winner of the Password Hashing Competition (https://password-hashing.net/) of 2015, and supported by Django through the installation of third-party libraries. You can see more information about this hasher in: https://en.wikipedia.org/wiki/Argon2.

3.3 How to include Figures

First you have to upload the image file from your computer using the upload link in the file-tree menu. Then use the includegraphics command to include it in your document. Use the figure environment and the caption command to add a number and a caption to your figure. See the code for Figure **??** in this section for an example.

Note that your figure will automatically be placed in the most appropriate place for it, given the surrounding text and taking into account other figures or tables that may be close by. You can find out more about adding images to your documents in this help article on including images on Overleaf.

3.4 How to add Tables

Use the table and tabular environments for basic tables — see Table **??**, for example. For more information, please see this help article on tables.

3.5 How to add Comments and Track Changes

Comments can be added to your project by highlighting some text and clicking "Add comment" in the top right of the editor pane. To view existing comments, click on the Review menu in the toolbar above. To reply to a comment, click on the Reply button in the lower right corner of the comment. You can close the Review pane by clicking its name on the toolbar when you're done reviewing for the time being.

Track changes are available on all our premium plans[5], and can be toggled on or off using the option at the top of the Review pane. Track changes allow you to keep track of every change made to the document, along with the person making the change.

3.6 How to add Lists

You can make lists with automatic numbering ...

1. Like this,
2. and like this.

... or bullet points ...

- Like this,
- and like this.

3.7 How to write Mathematics

LaTeX is great at typesetting mathematics. Let $X_1, X_2, \ldots, X_n$ be a sequence of independent and identically distributed random variables with $\mathrm{E}[X_i] = \mu$ and $\mathrm{Var}[X_i] = \sigma^2 < \infty$, and let

$$S_n = \frac{X_1 + X_2 + \cdots + X_n}{n} = \frac{1}{n}\sum_{i}^{n} X_i$$

denote their mean. Then as n approaches infinity, the random variables $\sqrt{n}(S_n - \mu)$ converge in distribution to a normal $\mathcal{N}(0, \sigma^2)$.

[5] https://www.overleaf.com/user/subscription/plans

3.8 How to customize the template

You may wish to customize the template for your own style, or to meet the specific needs of your documentation. If you're already familiar with LaTeX, you can go ahead and add the packages you're familiar with to the document preamble. If you run into any problems and can't find the answers in the package documentation or in the Overleaf help library[6], the forums such as TeX StackExchange[7] and LaTeX Community[8] are a great source of answers.

Some details on how to customize a .cls file (which sets the layout and overall format of the various elements of the template) can be found at Writing your own class[9], and LaTeX2e for class and package writers[10].

[6]https://www.overleaf.com/learn
[7]https://tex.stackexchange.com/
[8]https://latex.org/forum/
[9]https://www.overleaf.com/learn/latex/Writing_your_own_class
[10]http://texdoc.net/pkg/clsguide