Universidad del Centro

Educativo Latinoamericano

Estándares de Programación

Carrera:

Ingeniería en Sistemas de Información.

Docente:

Lic. Luciano Ripani.

Integrantes:

- Facundo Bachin.
- Pablo Amauli.
- Lucio Cesolari.

Historial de Versiones del documento

Fecha	Versión	Descripción	Autor
04/11/2020	1.0	Versión Inicial del Documento	Amauli Pablo, Cesolari Lucio, Bachin Facundo
11/11/20	2.0	Agregado de Historial de Versiones, Documentos relacionados y modificación en Introducción y Alcance de Documento.	Amauli Pablo, Cesolari Lucio, Bachin Facundo

Índice

1. Introducción

1.1. Propósito Del Documento

El propósito de este documento es determinar los estándares de programación de los diferentes niveles de programación y bases de datos utilizados en el proyecto “Red de Voluntariado”.

1.2. Alcance Del Documento

El alcance de este documento es el entorno de Typescript, Django y Modelos de bases de datos provisto por Django, el cual se construye como parte del proyecto final de los alumnos Bachin Facundo, Cesolari Lucio y Amauli Pablo, como parte de la carrera de Ingeniería en Sistemas de Información de la Universidad del Centro Educativo Latinoamericano.

1.3. Documentos Relacionados

Documento	Nombre / Ubicación del archivo	Fuente
Plan de Proyecto	**Nombre:** Plan de Proyecto V2.4.pdf **Ubicación:** https://drive.google.com/drive/folders/1DqYXQUH8cvFp7zoE55yxg6s1fotFGer6	Amauli Pablo, Cesolari Lucio, Bachin Facundo
Vista de Arquitectura	**Nombre:** Vista de Arquitectura v5.0.pdf **Ubicación:** https://drive.google.com/drive/folders/1ywsgFkVNK9PxZdEdWdFzrRBCJRf_Xc70	Amauli Pablo, Cesolari Lucio, Bachin Facundo

2. Desarrollo de los estándares

2.1. Estándares de programación (Typescript y Django)

- Django:
  - Utilizar para dividir todo tipo de etiqueta _, ej.: **eventos_postulados**
  - Utilizar _serializer para definir las clases tipo Serializer, ej.: **usuarios_activos_serializer**
  - Utilizar mayúscula en la primera letra de las clases, ej.: **Usuarios**
  - Dividir las temáticas en el código, ej.: Dividir con un comentario las clases utilizadas para usuarios de otras clases, ej.: **#Usuarios** //

- Typescript:
    - Utilizar $ al final de las variables que son tipo observables, ej.: **usuarios$**
    - Utilizar _ antes de las variables que son privadas, ej.: **private _apiService: ApiService**
    - Utilizar const para las variables que no cambiarán de valor en el transcurso del código en vez de let (ambas definen variables una indica que será constante y la otra solo la define como variable), ej.: **const dialogRef vs let dialogRef**
    - Utilizar el pipe async en vez de suscribirme a los observables para ahorrar pasos de desubscripción y tener una mejora en optimización, ej.: **(usuarios$|async) vs usuarios.subscribe(data=> this.usuarios = data)**
    - Utilizar mayúsculas para dividir los nombres de variables con múltiple representación, ej.: **fechaNacimiento**

2.2. Estándares de modelo (Django)

- Utilizar mayúsculas para dividir los nombres de campos con múltiple representación, ej.: **fechaNacimiento**
- Utilizar plural para el nombre de los modelos, ej.: **Estados**
- Utilizar minúsculas en los nombres de los campos a excepción de la regla número 1 donde se podría usar en los campos con múltiple representación.