# Lumino

Implementa un proyecto web Django que permita gestionar un centro de **formación online**.

La idea es disponer de un software en el que el alumnado se pueda matricular de distintos módulos y en el que el profesorado pueda añadir contenidos a dichos módulos así como calificar las materias.

## 1. Requerimientos

Para que puedas trabajar con normalidad en este ejercicio debes tener instalado en tu máquina:

1. `uv` → Gestor de paquetería y proyectos Python.
2. `just` → Lanzador de comandos como recetas.
3. `redis` → Almacenamiento de datos *clave-valor* en memoria.

## 2. Puesta en marcha

Se proporciona una *receta* `just` para la puesta en marcha del proyecto:

```
just setup
```

¿Qué ha ocurrido?

- Se ha creado un entorno virtual en la carpeta `.venv`
- Se han instalado las dependencias del proyecto.
- Se ha creado un proyecto Django en la carpeta `main`
- Se han aplicado las migraciones iniciales del proyecto.
- Se ha creado un *superusuario* con credenciales: `admin` - `admin`
- Se ha establecido el *timezone* a `Atlantic/Canary` en `settings.py`
- Se han establecido las configuraciones *media* en `settings.py`


CC BY 4.0

# 3. Aplicaciones

Habrá que añadir las siguientes aplicaciones:

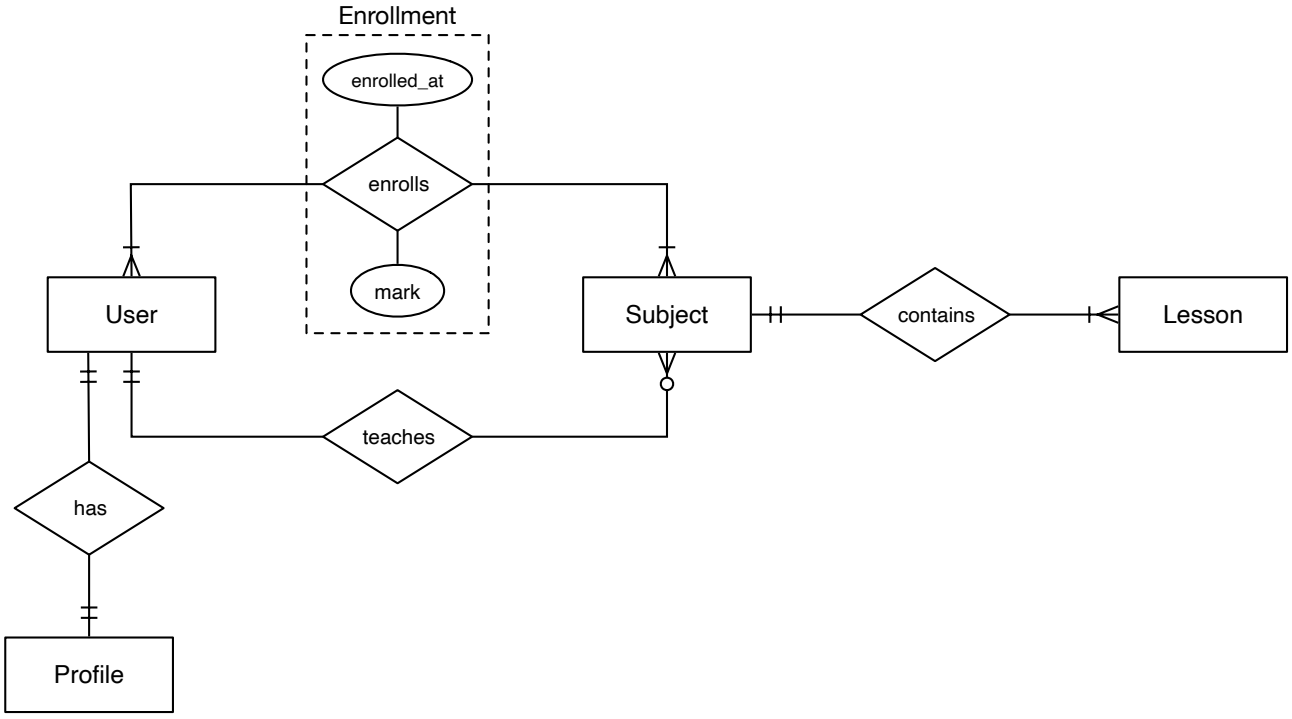
| | |
|---|---|
| `shared` | Artefactos compartidos. |
| `accounts` | Gestión de autenticación. |
| `subjects` | Gestión de módulos (materias) y unidades (temas). |
| `users` | Gestión de usuarios. |

Se proporciona una *receta* `just` para añadir una aplicación:

```
just startapp <app>
```

Esta receta no sólo crea la carpeta de la aplicación sino que añade la línea correspondiente de configuración en la variable `INSTALLED_APPS` del fichero `settings.py`.

# 4. Modelos




## 4.1. `subjects.Subject`

| **Campo** | **Tipo** |
|---|---|
| `code`$^{(*u)}$ | *str* |
| `name`$^{(*)}$ | *str* |
| `teacher`$^{(*)}$ | *fk* → `User` *(PROTECT)* |
| `students`$^{(\varnothing)}$ | *m2m* → `User` (`Enrollment`) |

 CC BY 4.0

### 4.2. `subjects.Lesson`

| Campo | Tipo |
|---|---|
| `subject`(*) | *fk* → `Subject` |
| `title`(*) | *str* |
| `content`(∅) | *str* |

### 4.3. `subjects.Enrollment`

| Campo | Tipo |
|---|---|
| `student`(*) | *fk* → `User` |
| `subject`(*) | *fk* → `Subject` |
| `enrolled_at`(*) | *date* |
| `mark`(∅ $v_{min}$=1 $v_{max}$=10) | *int* |

### 4.4. `users.Profile`

| Campo | Tipo |
|---|---|
| `user`(*) | *o2o* → `User` |
| `role`(*Δ=STUDENT) | *enum* |
| `avatar`(∅Δ=avatars/noavatar.png) | *image* |
| `bio`(∅) | *str* |

### 4.5. Carga de datos

Una vez que hayas **creado los modelos** asegúrate de que tu proyecto pasa los *tests core*. Se proporciona una *receta* `just` para ello:

```
just test tests/test_core.py
```

Ahora **crea y aplica las migraciones** y, si todo ha ido bien, procede a la carga de datos iniciales. Se proporciona una *receta* `just` para ello:

```
just load-data
```

*Puedes consultar los usuarios creados en cualquier momento con la receta*: `just show-users`

## 5. Requisitos de diseño

El proyecto deberá cumplir con los siguientes **requisitos de diseño** a nivel de organización interna del código:

- Utiliza herencia de plantillas, como mínimo, para la plantilla `base.html`.
- Utiliza inclusión de plantillas, como mínimo, para la cabecera `header.html`.

 CC BY 4.0

- Utiliza estáticos, como mínimo, para definir una hoja de estilos CSS.
- Utiliza formularios de modelo para todos los casos propuestos, salvo que se recomiende otra aproximación en algunas vistas.
- Utiliza conversores de ruta personalizados, como mínimo, para los distintos modelos.
- Utiliza "middleware" de mensajes, como mínimo, para los mensajes de confirmación.
- Utiliza URL canónica para los modelos que procedan.
- Utiliza Django Bootstrap para la interfaz del proyecto.
- Utiliza Sorl Thumbnail para la generación de miniaturas de imágenes.
- No modifiques el idioma por defecto:
  Debe ser `LANGUAGE_CODE = 'en_us'` en `main/settings.py`

### 5.1. Señales

Se debe implementar una señal para crear un *perfil de usuario vacío* cuando se guarda (por primera vez) un usuario.

Utiliza (entre otros) el parámetro `raw` chequeando que esté a `False` cuando vayas a crear el perfil.

### 5.2. Comandos de gestión

Se debe crear un *comando de gestión personalizado* para mostrar las **notas medias** de todos los módulos dados de alta en la plataforma. El comando debe alojarse en la aplicación `subjects` y debe llamarse `get_subject_stats.py`

El formato de salida (*con 2 cifras decimales*) debe ser el siguiente:

```
PRO: 4.32
DSW: 8.12
```

La nota media de cada módulo sale de calcular la media de todas las notas asignadas al alumnado matriculado en dicho módulo, excluyendo aquel alumnado que aún no tiene nota puesta. Si no hay ninguna nota puesta para el alumnado de un módulo, la nota media será **0.00**.

### 5.3. Procesadores de contexto

Se debe crear un *procesador de contexto personalizado* que inyecte en todas las plantillas el objecto `subjects` con los **módulos pertenecientes al usuario logeado**.

## 6. URLs

Para acceder a todas las URLs a excepción de `/login/` y `/signup/`, el usuario debe estar logeado.


CC BY 4.0

## 6.1. `main.urls`

`/` ⇒ `shared.views.index()`

- Si el usuario **no está logeado** habrá que mostrar una *homepage* de bienvenida (al menos con el mensaje Join Lumino. The cool education platform!) y con enlaces al *inicio de sesión* y *registro*.
- Si el usuario **está logeado** debe redirigir a `/subjects/`

`/setlang/es/` ⇒ `django.views.i18n.set_language()`

- Cambiar el idioma a `es` (*español*).
- También hay que contemplar que se pase a `en` (*inglés*).
- Al menos en `/subjects/` deben existir estos títulos:
  - **My subjects** (en idioma inglés).
  - **Mis módulos** (en idioma español).

## 6.2. `accounts.urls`

`/login/` ⇒ `accounts.views.user_login()`

- Inicio de sesión.
- Se debe solicitar nombre de usuario y contraseña.
- Si el usuario y/o contraseña son incorrectos, se debe mostrar un mensaje de error: Incorrect username or password.
- Si el usuario y contraseña son correctos, se debe iniciar la correspondiente sesión y redirigir al índice del proyecto.
- Hay que tener en cuenta la redirección en el caso de que tengamos un parámetro `next` desde GET.
- Debe existir un enlace para registrarse ("signup").

`/logout/` ⇒ `accounts.views.user_logout()`

- Cierre de sesión.
- Una vez cerrada la sesión, se debe redirigir al índice del proyecto.


CC BY 4.0

`/signup/` ⇒ `accounts.views.user_signup()`

- Registro de usuario (**sólo alumnado**).
- Campos a desplegar en el formulario:
  - Nombre de usuario$^{(*u)}$.
  - Contraseña$^{(*)}$.
  - Nombre$^{(*)}$.
  - Apellidos$^{(*)}$.
  - Correo electrónico$^{(*u)}$.
- Si el registro fue exitoso, se debe redirigir al índice del proyecto mostrando el mensaje: Welcome to Lumino. Nice to see you!
- Debe existir un enlace para logearse ("login").

## 6.3. `subjects.urls`

`/subjects/` ⇒ `subjects.views.subject_list()`

- Rol **profesorado**:
  - Mostrar el listado de módulos que imparte el/la profe.
  - Cada módulo debe tener un enlace para ir a su detalle.
- Rol **alumnado**:
  - Mostrar el listado de módulos de los que está matriculado el/la alumno/a.
  - Cada módulo debe tener un enlace para ir a su detalle.
  - Debe existir un enlace para matricularse de nuevos módulos.
  - Debe existir un enlace para "desmatricularse" de módulos existentes.
  - Si existe nota en todos los módulos en los que se ha matriculado, mostrar un enlace para solicitar *certifcado de calificaciones*.

 CC BY 4.0

`/subjects/DSW/` ⇒ `subjects.views.subject_detail()`

- Rol **profesorado**:
  - Mostrar las lecciones del módulo.
  - Para cada lección añadir enlace para ver lección, editar lección y borrar lección.
  - Enlace para añadir una nueva lección.
  - Enlace para ver las notas del alumnado del módulo.
  - Prohibido para profesorado que no imparta el módulo.
- Rol **alumnado**:
  - Mostrar las lecciones del módulo.
  - Cada lección debe tener un enlace para ir a su detalle.
  - Si el módulo ya tiene nota, mostrarla con → Your mark for this subject: 9
  - Prohibido para alumnado que no esté matriculado en el módulo.

`/subjects/DSW/lessons/add/` ⇒ `subjects.views.add_lesson()`

- Rol **profesorado**:
  - Mostrar/procesar el formulario para añadir una lección al módulo `DSW`.
  - Redirigir al listado de lecciones.
  - Si todo ha ido bien, mostrar el mensaje: Lesson was successfully added.
  - Prohibido para profesorado que no imparta el módulo `DSW`.
- Rol **alumnado**:
  - Prohibido.

`/subjects/DSW/lessons/17/` ⇒ `subjects.views.lesson_detail()`

- Rol **profesorado**:
  - Mostrar el contenido (como *markdown*) de la lección con `pk=17`.
  - Enlace para editar la lección.
  - Enlace para borrar la lección.
  - Prohibido para profesorado que no imparta el módulo.
- Rol **alumnado**:
  - Mostrar el contenido (como *markdown*) de la lección con `pk=17`.
  - Prohibido para alumnado que no esté matriculado en el módulo.

 CC BY 4.0

`/subjects/DSW/lessons/17/edit/` ⇒ `subjects.views.edit_lesson()`

- Rol **profesorado**:
  - Mostrar/procesar el formulario para editar la lección `pk=17` del módulo `DSW`.
  - Tras guardar, permanecer en la misma página (para seguir editando).
  - Si todo ha ido bien, mostrar el mensaje: Changes were successfully saved.
  - Prohibido para profesorado que no imparta el módulo `DSW`.
- Rol **alumnado**:
  - Prohibido.

`/subjects/DSW/lessons/17/delete/` ⇒ `subjects.views.delete_lesson()`

- Rol **profesorado**:
  - Borrar la lección `pk=17` del módulo `DSW`.
  - Redirigir al listado de lecciones.
  - Si todo ha ido bien, mostrar el mensaje: Lesson was successfully deleted.
  - Prohibido para profesorado que no imparta el módulo `DSW`.
- Rol **alumnado**:
  - Prohibido.

`/subjects/DSW/marks/` ⇒ `subjects.views.mark_list()`

- Rol **profesorado**:
  - Mostrar las calificaciones de todo el alumnado del módulo `DSW`.
  - Cada alumno/a debe estar en formato *Nombre Apellidos* junto con su nota.
  - Para cada alumno/a debe existir un enlace a su perfil.
  - Prohibido para profesorado que no imparta el módulo `DSW`.
- Rol **alumnado**:
  - Prohibido.

 CC BY 4.0

`/subjects/DSW/marks/edit/` ⇒ `subjects.views.edit_marks()`

- Rol **profesorado**:
  - Editar las calificaciones de todo el alumnado del módulo `DSW`.
  - Tras guardar, permanecer en la misma página (para seguir editando).
  - Si todo ha ido bien, mostrar el mensaje: Marks were successfully saved.
  - Prohibido para profesorado que no imparta el módulo `DSW`.
  - Se recomienda utilizar un Model formset de Django:

```python
# subjects/forms.py

class EditMarkForm(forms.ModelForm):
    class Meta:
        model = Enrollment
        fields = ['mark']
```

```python
# subjects/views.py

def edit_marks(request, subject_code: str):
    # ...
    MarkFormSet = modelformset_factory(Enrollment, EditMarkForm, extra=0)
    queryset = subject.enrollments.all()
    if request.method == 'POST':
        if (formset := MarkFormSet(queryset=queryset, data=request.POST)).is_valid():
            # TODO
    else:
        formset = MarkFormSet(queryset=queryset)
    return render(
        request,
        'subjects/marks/edit_marks.html',
        {'subject': subject, 'formset': formset},
    )
```

- Rol **alumnado**:
  - Prohibido.


CC BY 4.0

`/subjects/enroll/` ⇒ `subjects.views.enroll_subjects()`

- Rol **profesorado**:
  - Prohibido.
- Rol **alumnado**:
  - Matricularse en uno o varios módulos.
  - Es obligatorio mostrar (al menos) el **código** de cada módulo.
  - Sólo deben aparecer en el formulario aquellos módulos de los que aún no se ha matriculado.
  - Si todo ha ido bien, mostrar el mensaje: Successfully enrolled in the chosen subjects.
  - Redirigir al listado de módulos.
  - Se recomienda utilizar un formulario de clase con un campo de tipo `ModelMultipleChoiceField` para las `subjects`, inicialmente con `queryset=None`
  - En el constructor del formulario `__init__()` deberías pasar (desde la vista) el estudiante en cuestión y obtener la *queryset* adecuada para los módulos que deben mostrarse (aquellos en los que no está matriculado el estudiante).
  - En el constructor utiliza `self.fields` (diccionario) para acceder al campo `subjects` donde tendrás que asignar el listado de módulos a su atributo `queryset`.
  - Si prefieres utilizar *checkbox* en el formulario debes modificar el "widget" a `forms.CheckboxSelectMultiple`

 CC BY 4.0

`/subjects/unenroll/` ⇒ `subjects.views.unenroll_subjects()`

- Rol **profesorado**:
  - Prohibido.
- Rol **alumnado**:
  - Desmatricularse en uno o varios módulos.
  - Es obligatorio mostrar (al menos) el **código** de cada módulo.
  - Sólo deben aparecer en el formulario aquellos módulos de los que ya se ha matriculado.
  - Si todo ha ido bien, mostrar el mensaje: Successfully unenrolled from the chosen subjects.
  - Redirigir al listado de módulos.
  - Se recomienda utilizar un formulario de clase con un campo de tipo `ModelMultipleChoiceField` para las `subjects`, inicialmente con `queryset=None`
  - En el constructor del formulario `__init__()` deberías pasar (desde la vista) el estudiante en cuestión y obtener la *queryset* adecuada para los módulos que deben mostrarse (aquellos en los que está matriculado el estudiante).
  - En el constructor utiliza `self.fields` (diccionario) para acceder al campo `subjects` donde tendrás que asignar el listado de módulos a su atributo `queryset`.
  - Si prefieres utilizar *checkbox* en el formulario debes modificar el "widget" a `forms.CheckboxSelectMultiple`

 CC BY 4.0

`/subjects/certificate/` ⇒ `subjects.views.request_certificate()`

- Rol **profesorado**:
  - Prohibido.
- Rol **alumnado**:
  - Solicitar certificado de calificaciones.
  - Se debe redirigir a una página con un mensaje de confirmación: You will get the grade certificate quite soon at guido@example.com
  - El correo debe contener un adjunto con el certificado en PDF generado dinámicamente desde la propia aplicación.
  - La tarea Django-RQ debe ubicarse en `subjects.tasks.deliver_certificate`
  - Signatura de la función: `def deliver_certificate(base_url, student) { ... }`
  - El certificado debe generarse en: `/media/certificates/guido_grade_certificate.pdf`
  - Se debe hacer uso de la clase `EmailMessage` y de su método `send()` para enviar el correo.
  - Prohibido cuando alguno de los módulos aún no tenga calificación.
  - La llamada desde esta vista a la tarea desacoplada debe ser: `deliver_certificate.delay(request.build_absolute_uri(), request.user)`

## 6.4. `users.urls`

`/users/guido/` ⇒ `users.views.user_detail()`

- Visualizar el perfil del usuario con nombre de usuario `guido`.
- Elementos a mostrar:
  - Identificación del usuario en formato *Nombre Apellidos*.
  - *Teacher* (si es profesorado) o *Student* si es alumnado.
  - Fotografía de "avatar" (**obligatorio** usar aquí las etiquetas de `sorl-thumbnail`).
  - Correo electrónico.
  - Biografía completa.
- Debe existir un botón para *editar el perfil*.
- Debe existir un botón para *abandonar la plataforma* (**sólo para alumnado**).
- Si el usuario no existe habrá que devolver una respuesta 404.

 CC BY 4.0

`/user/edit/` ⇒ `users.views.edit_profile()`

- Editar el perfil del usuario logeado.
- Los campos serían: `bio` y `avatar`.
- Redirigir a la visualización del perfil.
- Si todo ha ido bien, mostrar el mensaje: User profile has been successfully saved.

`/user/leave/` ⇒ `users.views.leave()`

- Rol **profesorado**:
  - Prohibido.
- Rol **alumnado**:
  - Abandonar la plataforma.
  - Redirigir a `/`
  - Si todo ha ido bien, mostrar el mensaje: Good bye! Hope to see you soon.

# 7. Administración

Los siguientes modelos deben estar accesibles desde la **interfaz administrativa** de Django:

- `subjects.Subject`
- `subjects.Lesson`
- `subjects.Enrollment`
- `users.Profile`

 CC BY 4.0