Lumino

Implementa un proyecto web Django para **gestión académica**.

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. Puesta en marcha

Lleva a cabo los siguientes comandos para la puesta en marcha del proyecto:

```
just create-venv
source .venv/bin/activate
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`

2. Aplicaciones

Habrá que añadir las siguientes aplicaciones:

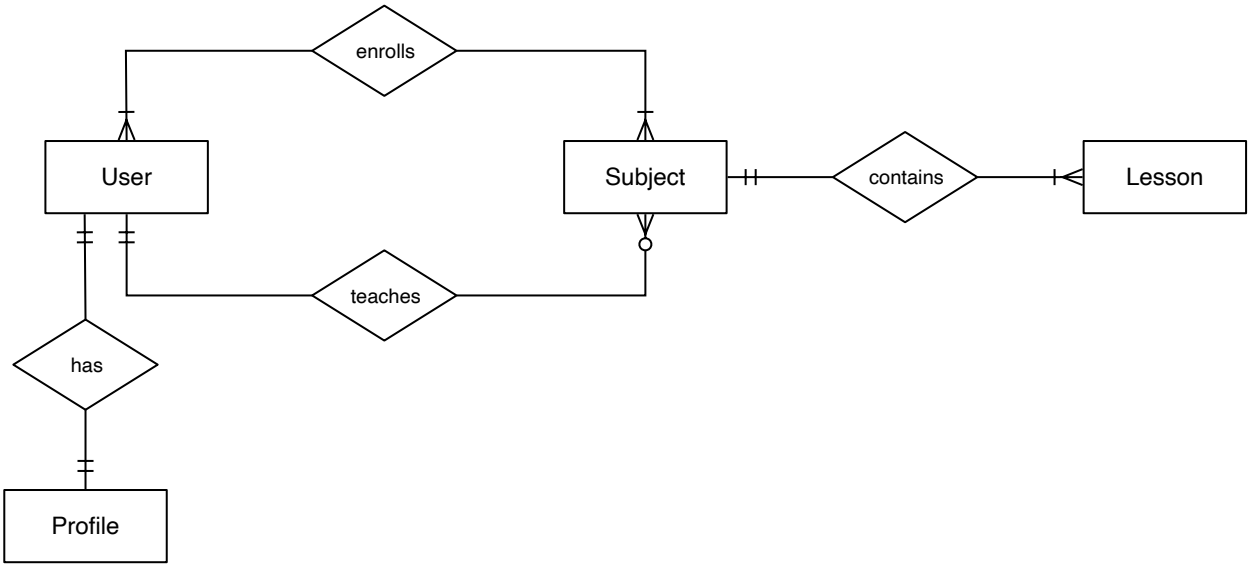
`accounts`	Gestión de autenticación.
`shared`	Artefactos compartidos.
`users`	Gestión de usuarios.
`subjects`	Gestión de módulos.

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`.

3. Modelos



3.1. `subjects.Subject`

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

3.2. `subjects.Lesson`

Campo	Tipo
`subject`$^{(*)}$	*fk* → `Subject`
`title`$^{(*)}$	*str*
`content`$^{(\varnothing)}$	*str*

3.3. `subjects.Enrollment`

Campo	Tipo
`student`$^{(*)}$	*fk* → `User`
`subject`$^{(*)}$	*fk* → `Subject`
`enrolled_at`$^{(*)}$	*date*
`mark`$^{(\varnothing)}$	*int*

3.4. `users.Profile`

Campo	Tipo
`user`$^{(*)}$	*o2o* → `User`
`role`$^{(*\Delta)}$	*enum*
`avatar`$^{(\varnothing\Delta)}$	*image*
`bio`$^{(\varnothing)}$	*str*

Valores por defecto (Δ):

- `role` → `STUDENT`
- `avatar` → `avatars/noavatar.png`

3.5. `User`

No hay que implementar este modelo. Se usará el modelo `User` que ofrece Django.

4. URLs

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

4.1. `main.urls`

`/` ⇒ `shared.views.index()`

- Si el usuario no está logeado habrá que mostrar una *homepage* de bienvenida con opciones de *login* y *registro*.
- Si el usuario está logeado debe redirigir a `/subjects/`

4.2. `accounts.urls`

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

- Inicio de sesión.
- Se debe solicitar nombre de usuario$^{(*)}$ y contraseña$^{(*)}$.
- Se debe incluir un enlace al *registro de usuario*.

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

- Cierre de sesión.
- Mediante petición `GET`.

`/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$^{(*u)}$.
- Se debe crear un *perfil de usuario vacío*.
- Se debe incluir un enlace al *registro de usuario*.
- Si todo ha ido bien, mostrar el mensaje: *Welcome to Lumino. Nice to see you!.*

4.3. `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*.

`/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 editar 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 en esta página.
  - 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.

`/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.

`/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`.
- Rol **alumnado**:
  - Prohibido.

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

- Rol **profesorado**:
  - Prohibido.
- Rol **alumnado**:
  - Matricularse en uno o varios módulos.
  - 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.*

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

- Rol **profesorado**:
  - Prohibido.
- Rol **alumnado**:
  - Desmatricularse en uno o varios módulos.
  - 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.*

`/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.

4.4. `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 exsitir un botón para *abandonar la plataforma* (**sólo para alumnado**).

`/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.*

5. Administración

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

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