INGENIERIA INDUSTRIAL
UNIVERSIDAD DE CHILE

Auxiliar 8

IN3501 Tecnologías de Información para la Comunicación y Gestión

Auxiliares:
Gustavo Álvarez – galvarez901@gmail.com
Javiera Ovalle – javiera.ovallet@gmail.com
Jorge Pinto – jorgepintoriveros@gmail.com

Agenda

- Administrar Bases de Datos en Django
- Extendiendo el proyecto
- Usuarios en Django

Administración de Bases de Datos

Cuando creamos una base de datos difícilmente lo haremos bien a la primera

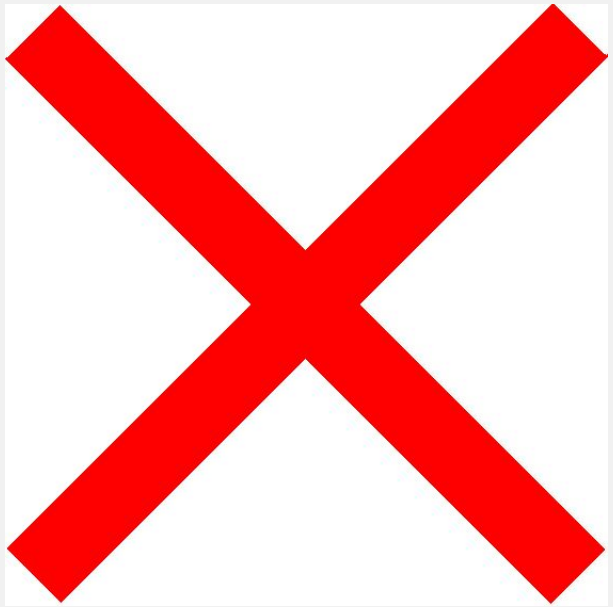
Nos puede faltar una relación o una entidad. Nos podemos equivocamos en el nombre de una tabla.

Administración de Bases de Datos

¿Qué hacemos si nos equivocamos?

Administración de Bases de Datos

Realizamos un nuevo modelo en workbench y luego lo exportamos a MySQL



En rigor se puede hacer,
pero seria trabajar de más

Administración de Bases de Datos

Modificar la base de datos a través de django

Al modificar los modelos de django se puede modificar la base de datos de MySQL

Administración de Bases de Datos

Ejemplo: Modificaremos la tabla usuarios de la base de datos de la auxiliar anterior.

Estructura de la tabla usuarios.

```
mysql> describe usuarios;
+-------------+--------------+------+-----+---------+----------------+
| Field       | Type         | Null | Key | Default | Extra          |
+-------------+--------------+------+-----+---------+----------------+
| id          | int(11)      | NO   | PRI | NULL    | auto_increment |
| Nombre      | varchar(64)  | NO   |     | NULL    |                |
| Apellido    | varchar(64)  | NO   |     | NULL    |                |
| Direccion   | varchar(128) | NO   |     | NULL    |                |
| RUT         | varchar(32)  | NO   |     | NULL    |                |
| Puntaje_PSU | double       | NO   |     | NULL    |                |
| password    | varchar(64)  | NO   |     | NULL    |                |
| username    | varchar(64)  | NO   |     | NULL    |                |
+-------------+--------------+------+-----+---------+----------------+
8 rows in set (0.00 sec)
```

Administración de Bases de Datos

Ejemplo: Modificaremos la tabla usuarios de la base de datos de la auxiliar anterior.

```python
class Usuarios(models.Model):
    id = models.IntegerField(db_column="id",primary_key=True,)
    nombre = models.CharField(db_column='Nombre', max_length=64)  # Field
    apellido = models.CharField(db_column='Apellido', max_length=64)  # F
    direccion = models.CharField(db_column='Direccion', max_length=128)
    rut = models.CharField(db_column='RUT', max_length=32)  # Field name
    puntaje_psu = models.IntegerField(db_column='Puntaje_PSU')  # Field n
    username = models.CharField(max_length=64)
    password = models.CharField(max_length=64)
    email = models.CharField(max_length=50)

    class Meta:
        managed = True
        db_table = 'usuarios'
```

Queremos editar el la clase Usuarios

Quitaremos el nombre de usuario y contraseña

```python
class Usuarios(models.Model):
    id = models.IntegerField(db_column="id",primary_key=True,)
    nombre = models.CharField(db_column='Nombre', max_length=64)  # Fiel
    apellido = models.CharField(db_column='Apellido', max_length=64)  #
    direccion = models.CharField(db_column='Direccion', max_length=128)
    rut = models.CharField(db_column='RUT', max_length=32)  # Field name
    puntaje_psu = models.IntegerField(db_column='Puntaje_PSU')  # Field

    email = models.CharField(max_length=50)

    class Meta:
        managed = True
        db_table = 'usuarios'
```

Administración de Bases de Datos

Ejemplo: Modificaremos la tabla usuarios de la base de datos de la auxiliar anterior.

Si queremos ver los cambios reflejados en MySQL debemos realizar las migraciones de python.
Luego de modificado lo necesario debemos ejecutar lo siguiente en nuestra consola:

```
$ python manage.py makemigrations

$ python manage.py migrate
```

Administración de Bases de Datos

Ejemplo: Modificaremos la tabla usuarios de la base de datos de la auxiliar anterior.

```
^CiMac-de-Ricardo:Auxiliar7 jorgepinto$ python3 manage.py makemigrations
Migrations for 'ejemplo7':
  ejemplo7/migrations/0004_auto_20180709_1259.py
    - Remove field password from usuarios
    - Remove field username from usuarios
iMac-de-Ricardo:Auxiliar7 jorgepinto$ python3 manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, ejemplo7, sessions
Running migrations:
  Applying ejemplo7.0004_auto_20180709_1259... OK
```

Administración de Bases de Datos

Ejemplo: Modificaremos la tabla usuarios de la base de datos de la auxiliar anterior.

```
mysql> describe usuarios;
+-------------+--------------+------+-----+---------+----------------+
| Field       | Type         | Null | Key | Default | Extra          |
+-------------+--------------+------+-----+---------+----------------+
| id          | int(11)      | NO   | PRI | NULL    | auto_increment |
| Nombre      | varchar(64)  | NO   |     | NULL    |                |
| Apellido    | varchar(64)  | NO   |     | NULL    |                |
| Direccion   | varchar(128) | NO   |     | NULL    |                |
| RUT         | varchar(32)  | NO   |     | NULL    |                |
| Puntaje_PSU | double       | NO   |     | NULL    |                |
| password    | varchar(64)  | NO   |     | NULL    |                |
| username    | varchar(64)  | NO   |     | NULL    |                |
+-------------+--------------+------+-----+---------+----------------+
8 rows in set (0.00 sec)
```

```
mysql> describe usuarios;
+-------------+--------------+------+-----+---------+----------------+
| Field       | Type         | Null | Key | Default | Extra          |
+-------------+--------------+------+-----+---------+----------------+
| id          | int(11)      | NO   | PRI | NULL    | auto_increment |
| Nombre      | varchar(64)  | NO   |     | NULL    |                |
| Apellido    | varchar(64)  | NO   |     | NULL    |                |
| Direccion   | varchar(128) | NO   |     | NULL    |                |
| RUT         | varchar(32)  | NO   |     | NULL    |                |
| Puntaje_PSU | double       | NO   |     | NULL    |                |
+-------------+--------------+------+-----+---------+----------------+
6 rows in set (0.00 sec)
```

Extendiendo el proyecto

Queremos agregarle más funcionalidades a nuestro proyecto

En específico, vamos a agregar un sistema de autenticación de usuarios (login, logout, registro)

Extendiendo el proyecto

Crearemos una nueva aplicación, de esta manera el manejo de los usuarios será más MODULAR

```
$ python manage.py startapp usuarios
```

Extendiendo el proyecto

Crearemos una nueva aplicación, de esta manera el manejo de los usuarios será más MODULAR
Esta aplicación se encargará de crear usuarios, del login y logout de estos mismo

```
$ python manage.py startapp usuarios
```

Extendiendo el proyecto

Recordar que al crear una nueva aplicación hay que agregarla al proyecto tanto en settings.py como en urls.py

```
31  # Application definition
32
33  INSTALLED_APPS = [
34      'django.contrib.admin',
35      'django.contrib.auth',
36      'django.contrib.contenttypes',
37      'django.contrib.sessions',
38      'django.contrib.messages',
39      'django.contrib.staticfiles',
40      'ejemplo7',
41      'usuarios',
42  ]
```

```
16  from django.contrib import admin
17  from django.urls import path, include
18
19  urlpatterns = [
20      path('admin/', admin.site.urls),
21      path('ejemplo7/', include('ejemplo7.urls')),
22      path('', include('usuarios.urls')),
23
24  ]
25
```

como se escogió el prefijo vacío (‘’), la url base para la aplicación usuarios será localhost:8000/

Usuarios en Django

Django tiene su propio módulo de Usuarios.
Es importante saber usar las herramientas de los Frameworks, para así evitar trabajo (no es necesario reinventar la rueda)

The primary attributes of the default user are:

- **username**
- **password**
- **email**
- **first_name**
- **last_name**

[En este enlace esta toda la documentacion de los usuarios]

```
mysql> show tables
    -> ;
+----------------------------+
| Tables_in_in2501           |
+----------------------------+
| auth_group                 |
| auth_group_permissions     |
| auth_permission            |
| auth_user                  |
| auth_user_groups           |
| auth_user_user_permissions |
| cursos                     |
| departamento               |
| django_admin_log           |
| django_content_type        |
| django_migrations          |
| django_session             |
| rol_usuario_curso          |
| semestres                  |
| tipo_usuario               |
| usuario_tipo_usuario       |
| usuarios                   |
+----------------------------+
```

Django carga los modulos de usuarios en nuestra base de datos

```
mysql> describe auth_user;
+--------------+--------------+------+-----+---------+----------------+
| Field        | Type         | Null | Key | Default | Extra          |
+--------------+--------------+------+-----+---------+----------------+
| id           | int(11)      | NO   | PRI | NULL    | auto_increment |
| password     | varchar(128) | NO   |     | NULL    |                |
| last_login   | datetime(6)  | YES  |     | NULL    |                |
| is_superuser | tinyint(1)   | NO   |     | NULL    |                |
| username     | varchar(150) | NO   | UNI | NULL    |                |
| first_name   | varchar(30)  | NO   |     | NULL    |                |
| last_name    | varchar(150) | NO   |     | NULL    |                |
| email        | varchar(254) | NO   |     | NULL    |                |
| is_staff     | tinyint(1)   | NO   |     | NULL    |                |
| is_active    | tinyint(1)   | NO   |     | NULL    |                |
| date_joined  | datetime(6)  | NO   |     | NULL    |                |
+--------------+--------------+------+-----+---------+----------------+
```

Usuarios en Django

Un usuario en Django es un objeto User, la manera mas facil de usarlo es creando una relación 1:1 con alguna entidad de nuestra base de datos, en este caso escogeremos Usuarios:

```
8  from django.db import models
9  from django.contrib.auth.models import User
10
11
```

```
$ python manage.py makemigrations

$ python manage.py migrate
```

```
80  class Usuarios(models.Model):
81      id = models.IntegerField(db_column="id",primary_key=True,)
82      nombre = models.CharField(db_column='Nombre', max_length=64)  # Field name ma
83      apellido = models.CharField(db_column='Apellido', max_length=64)  # Field nam
84      direccion = models.CharField(db_column='Direccion', max_length=128)  # Field
85      rut = models.CharField(db_column='RUT', max_length=32)  # Field name made low
86      puntaje_psu = models.IntegerField(db_column='Puntaje_PSU')  # Field name made
87      email = models.CharField(max_length=50)
88
89      user = models.OneToOneField(User, on_delete=models.CASCADE, default = True)
90
91      class Meta:
92          managed = True
93          db_table = 'usuarios'
94
```

Ahora para crear un usuario, primero debemos crear un user:
new_user = User.objects.create_user(username='username',
password='password')
new_user.save()

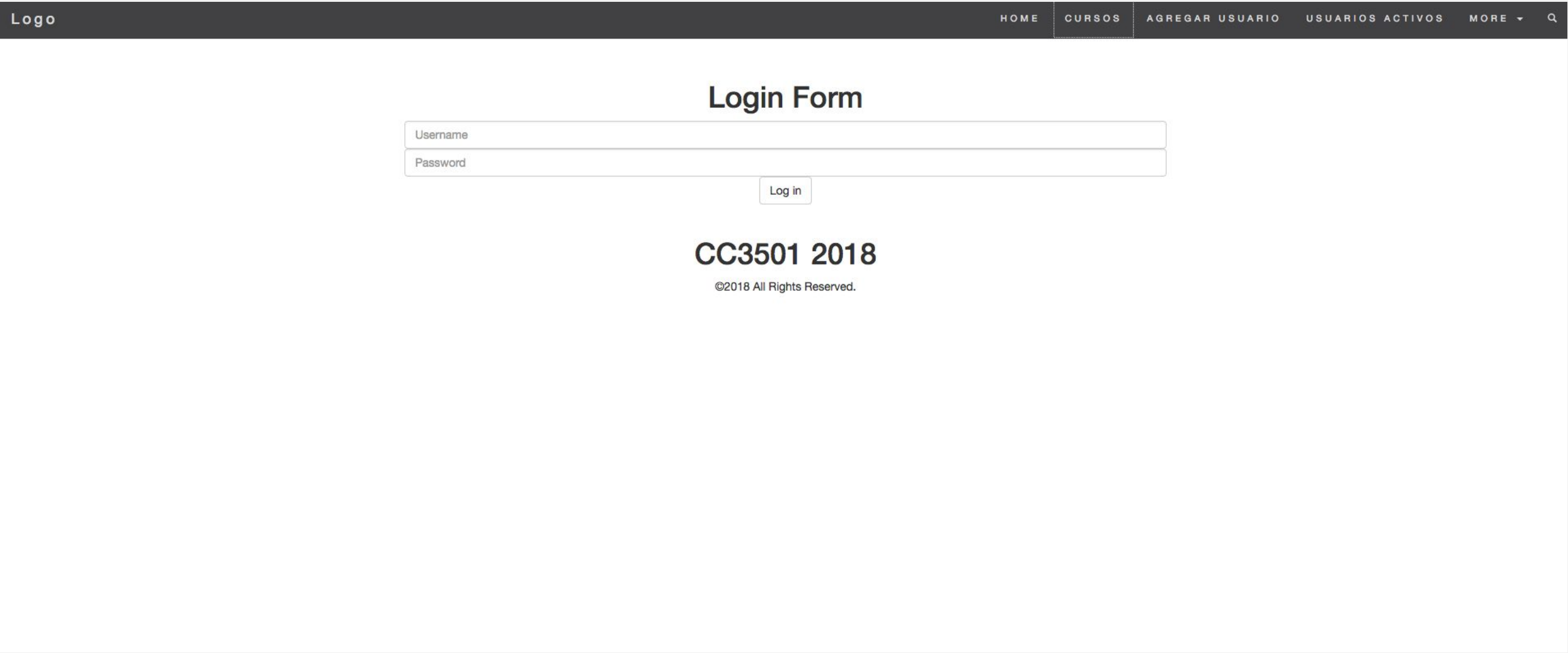
usuario = Usuario(user = new_user, nombre = '...', apellido,...)
usuario.save()

** tiene el valor de default=True ya que en la base de datos ya existen usuarios sin User, estos permite que el valor sea null para esos usuarios

Usuarios en Django: Login (formulario)

```
<div class="container text-center">
  {% if user.is_authenticated %}
    Hi {{ user.username }}!
    <br>
    <p><a href="/logout/">logout</a></p>

  {% else %}
    <div>
      <div class="login_wrapper">
        <div class="animate form login_form">
          <section class="login_content">
            <form action ="/login/" method="POST">
              {% csrf_token %}
              <h1>Login Form</h1>
              <div>
                <input type="text" class="form-control" placeholder="Username" required="" name="username" autofocus="" required=""/>
              </div>
              <div>
                <input type="password" class="form-control" placeholder="Password" required="" name="password" required=""/>
              </div>
              <div>
                <button class="btn btn-default submit" type="submit">Log in</button>
              </div>

              <div class="clearfix"></div>

              <div class="separator">

                <div class="clearfix"></div>
                <br />

                <div>
                  <h1><i class="fa fa-paw"></i> CC3501 2018</h1>
                  <p>©2018 All Rights Reserved.</p>
                </div>
              </div>
            </form>
          </section>
        </div>
      </div>
    </div>

  {% endif %}
```

Usuarios en Django: Login (formulario)



Usuarios en Django: Login (logica)

```
urlpatterns = [

    path('', views.index, name='index'),
    path('login/', views.loginView, name='login'),
    path('logout/', views.logoutView, name='logout'),
]
```

```
def loginView(request):
    username = request.POST.get('username')
    password = request.POST.get('password')

    user = authenticate(username=username, password=password)
    if user is not None:
        login(request, user)

    return render(request, 'usuarios/index-login.html')
```

Usuarios en Django: Logout (url)

```
73      <div class="collapse navbar-collapse" id="myNavbar">
74        <ul class="nav navbar-nav navbar-right">
75          <li><a href='#'>HOME</a></li>
76          <li><a href='ejemplo7/cursos/'>CURSOS</a></li>
77          <li><a href='ejemplo7/adduser/'>AGREGAR USUARIO</a></li>
78          <li><a href='ejemplo7/users/'>USUARIOS ACTIVOS</a></li>
79          {% if user.is_authenticated %}
80          <li><a href="/logout/"'>Logout</a></li>
81          {% else %}
82          <li class="dropdown">
83            <a class="dropdown-toggle" data-toggle="dropdown" href="#">MORE
84              <span class="caret"></span>
85            </a>
```

Usuarios en Django: Logout (logica)

```
6  urlpatterns = [
7
8      path('', views.index, name='index'),
9      path('login/', views.loginView, name='login'),
10     path('logout/', views.logoutView, name='logout'),
11 ]
```

```
19 def logoutView(request):
20     logout(request)
21     return redirect('/')
```

Usuarios en Django

La información del usuario Django se encarga de guardarla en las cookies de nuestro computador.

Se puede acceder a la información del user en cualquier momento en las vistas mediante:
request.user

Se puede acceder al usuario (usuario creado por nosotros, no el de django) usando el usuario que está guardado en el request mediante:
user = request.user
usuario = Usuario.objects.get(user=user)

El uso de usuarios de django facilita mucho las cosas en cuanto a los permisos y también es extensible para tener muchos tipos de usuarios.

INGENIERIA INDUSTRIAL
UNIVERSIDAD DE CHILE

Auxiliar 8

IN3501 Tecnologías de Información para la Comunicación y Gestión

Auxiliares:
Gustavo Álvarez – galvarez901@gmail.com
Javiera Ovalle – javiera.ovallet@gmail.com
Jorge Pinto – jorgepintoriveros@gmail.com