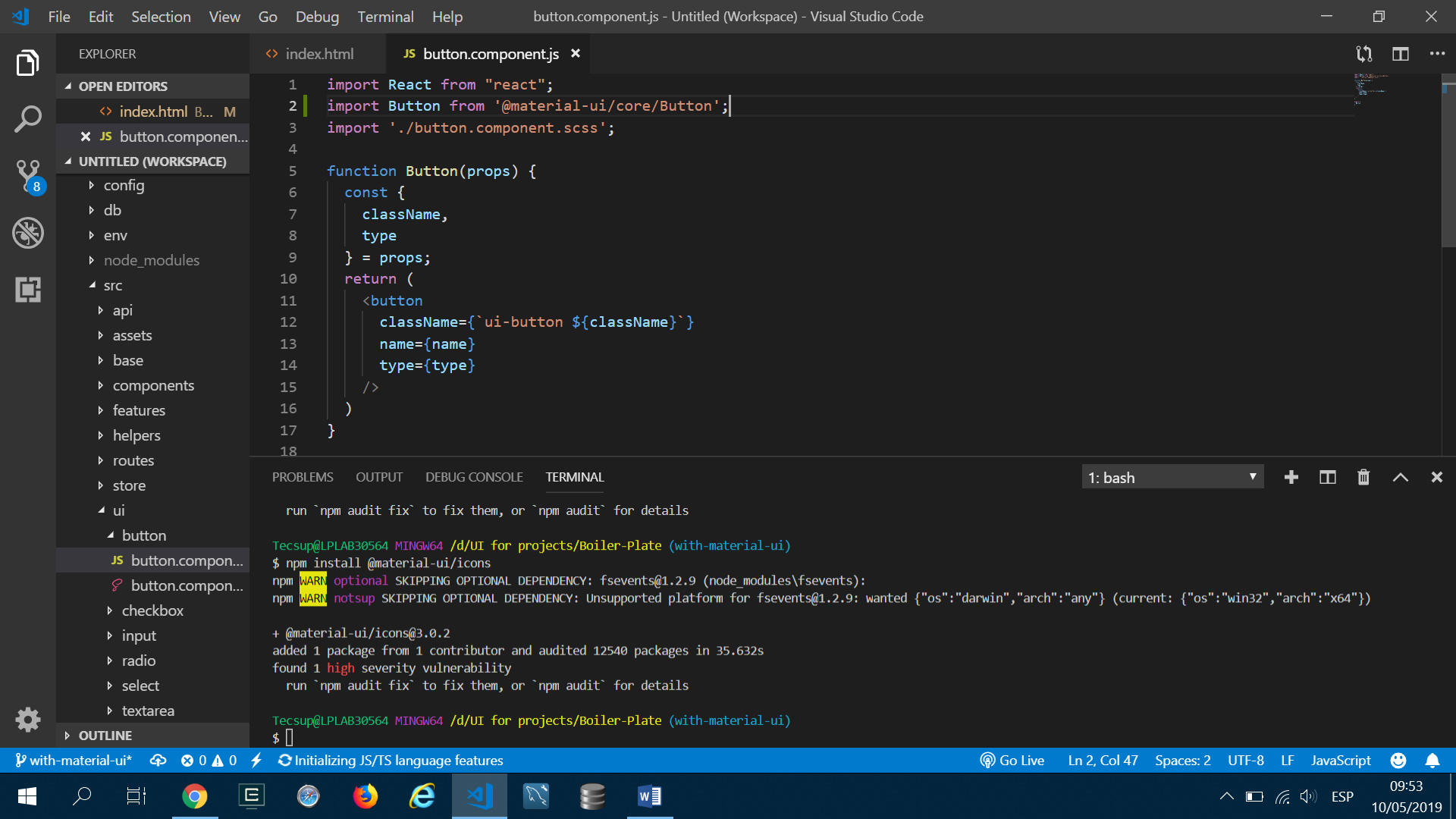
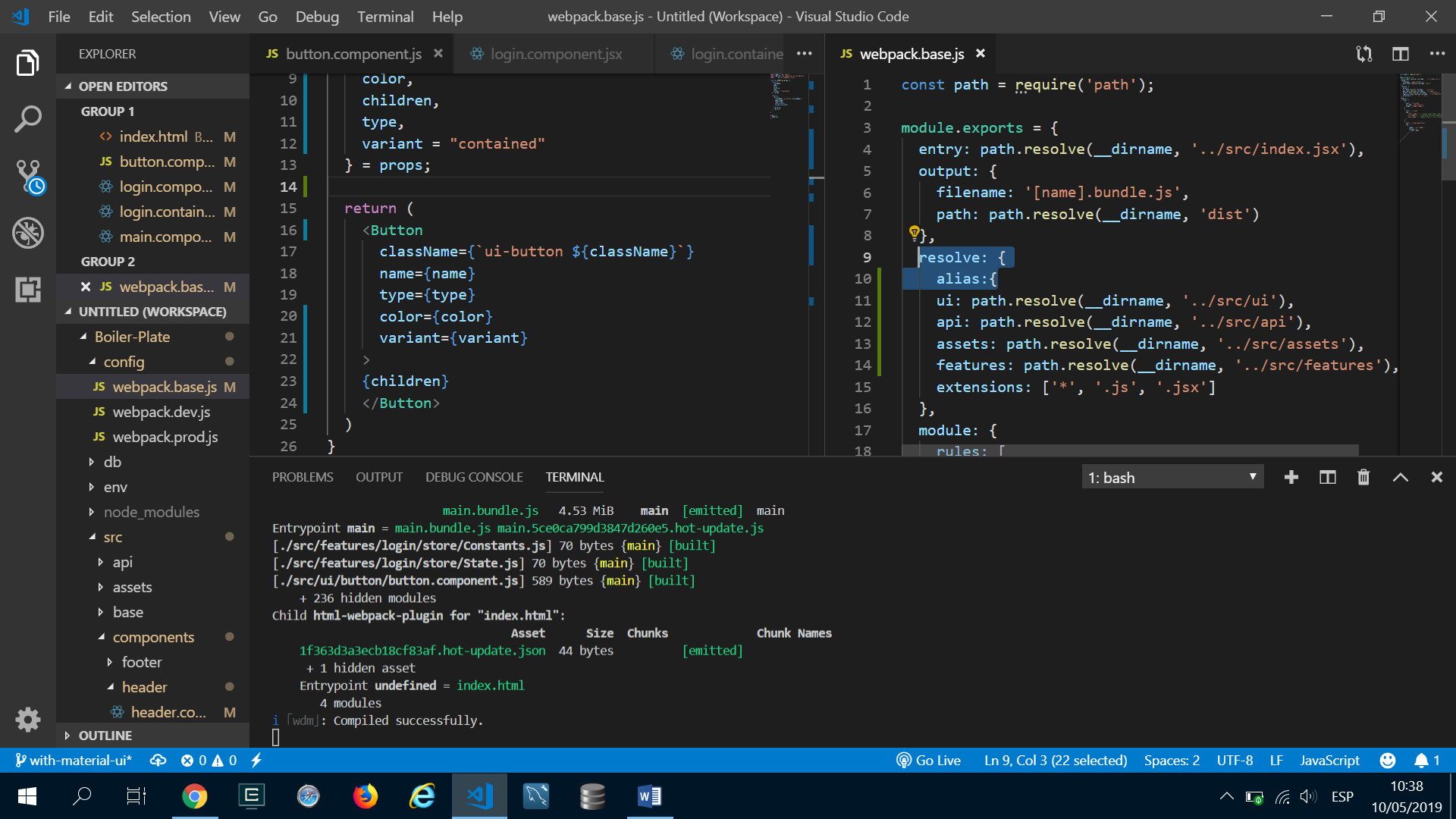
Install materia-ui:

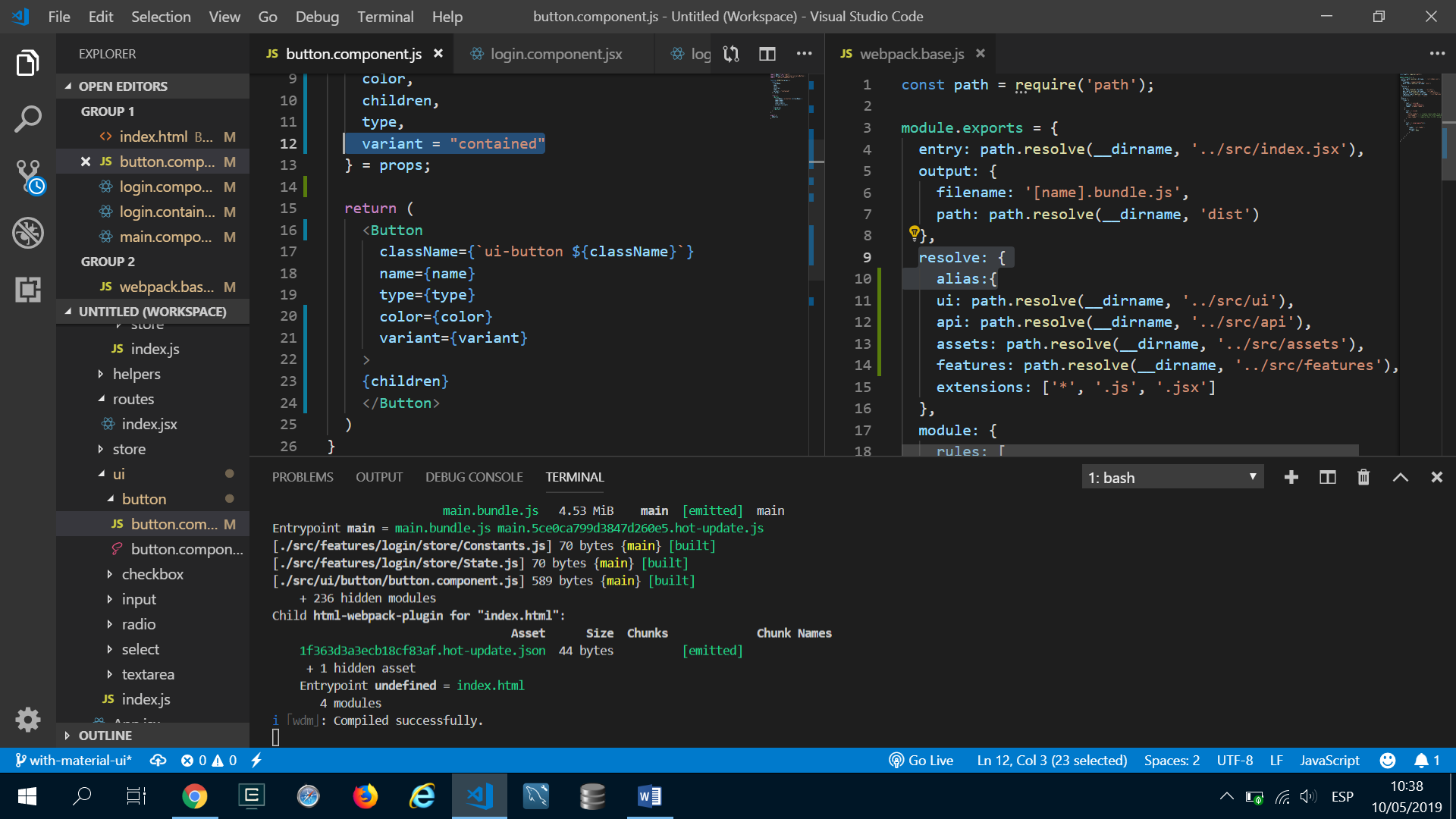
<https://material-ui.com/getting-started/installation/>



Podemos usar un alias y configurarlo en webpack para evitar usar “../../../” (<https://webpack.js.org/configuration/resolve/>) El problema es que no podemos linkear a los archivos luego.



Si incluyo variant se puede editar el diseño del botón, si no incluyo variant como propiedades no se mostrará ningún diseño costumizado.



Alt para mover un row hacia arriba y hacia abajo

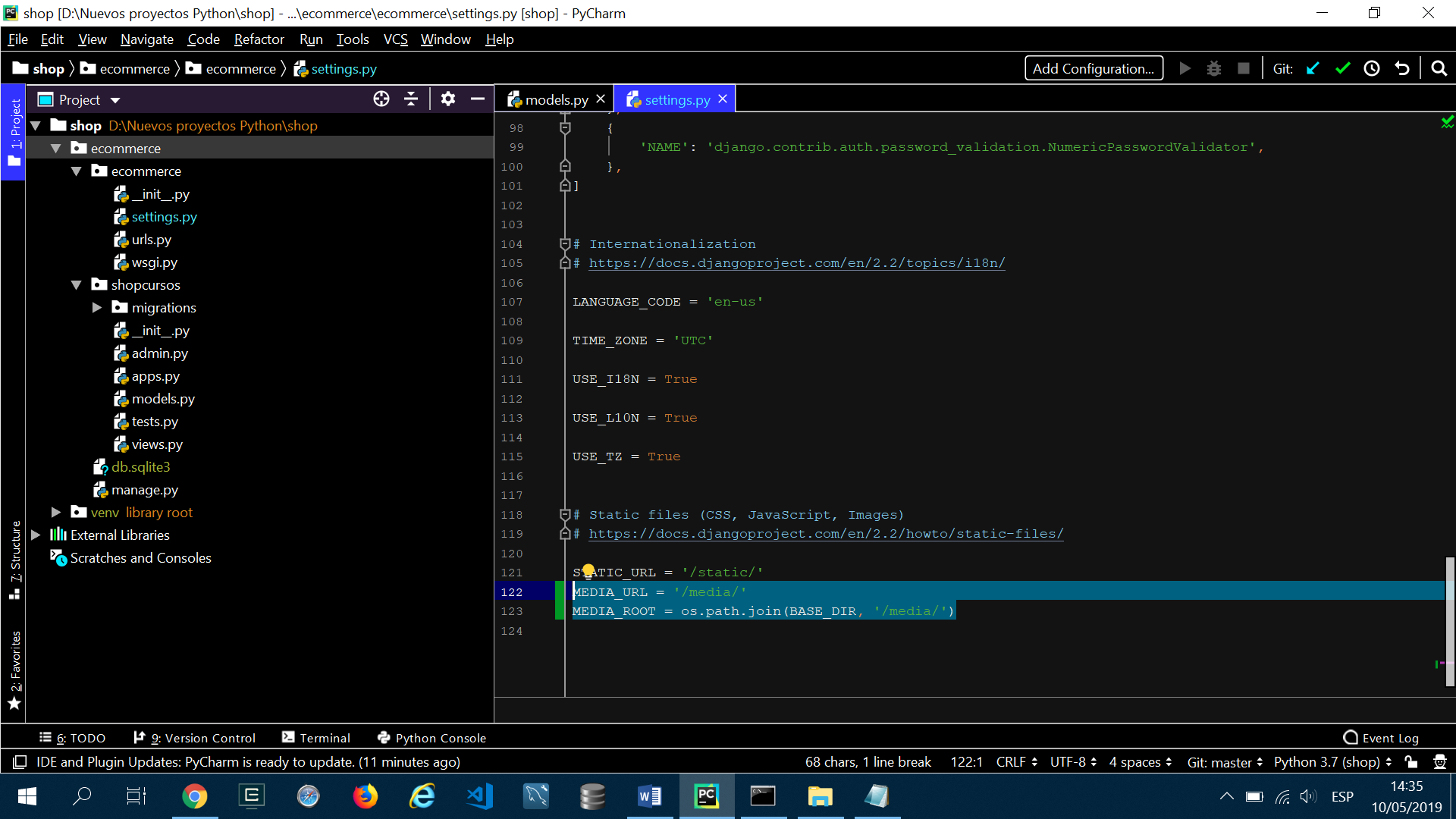
FIRST:

import { connect } from 'react-redux';

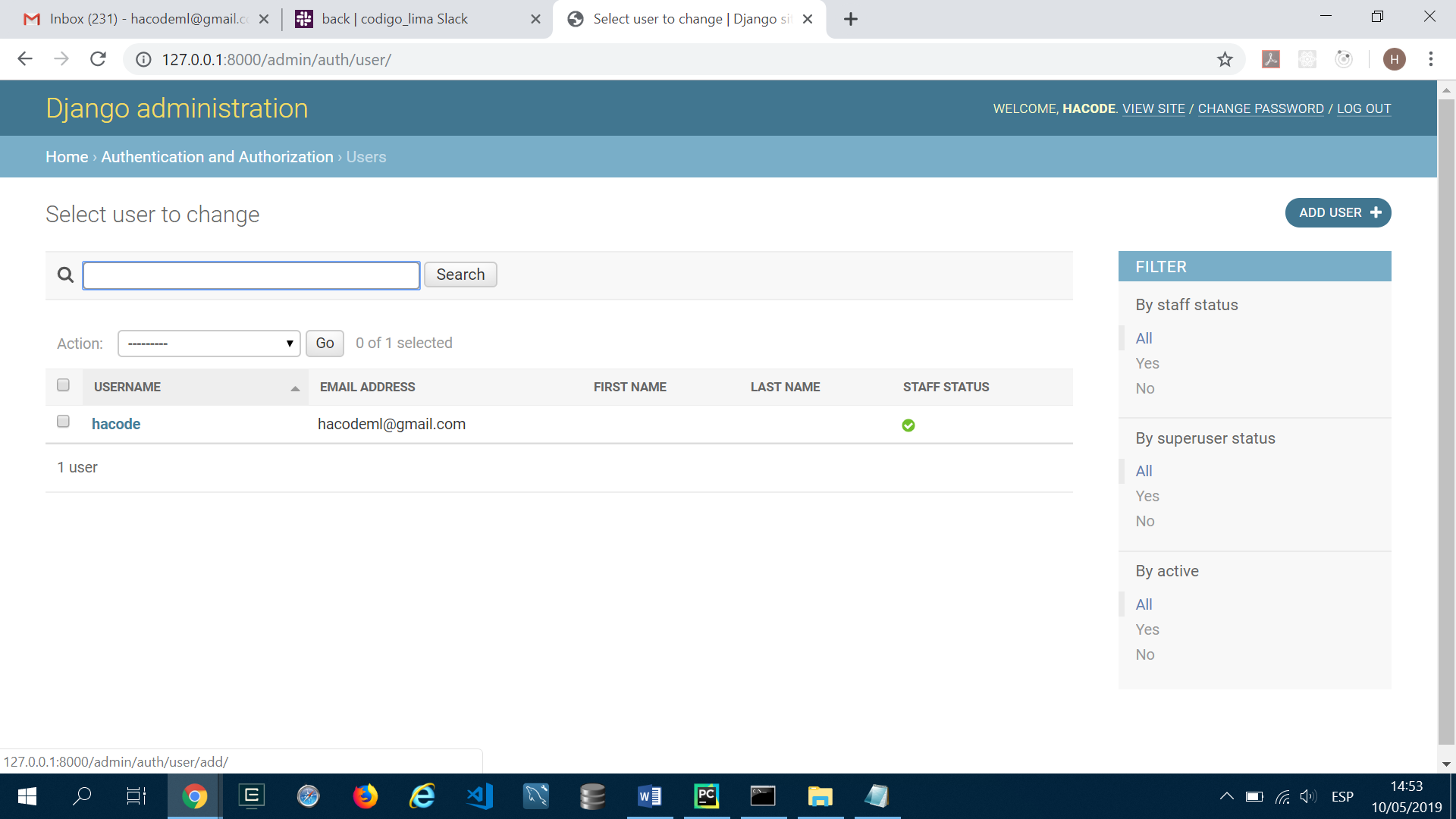
BACKEND:

ADD to Settings in main project, to upload images to /media/ folder.

ERROR: I wrote /media/ and what it did was to create the media folder outside the main folder (nuevos\_proyectos\_python), I must write only “media/”, that way the folder is created inside my project Folder.



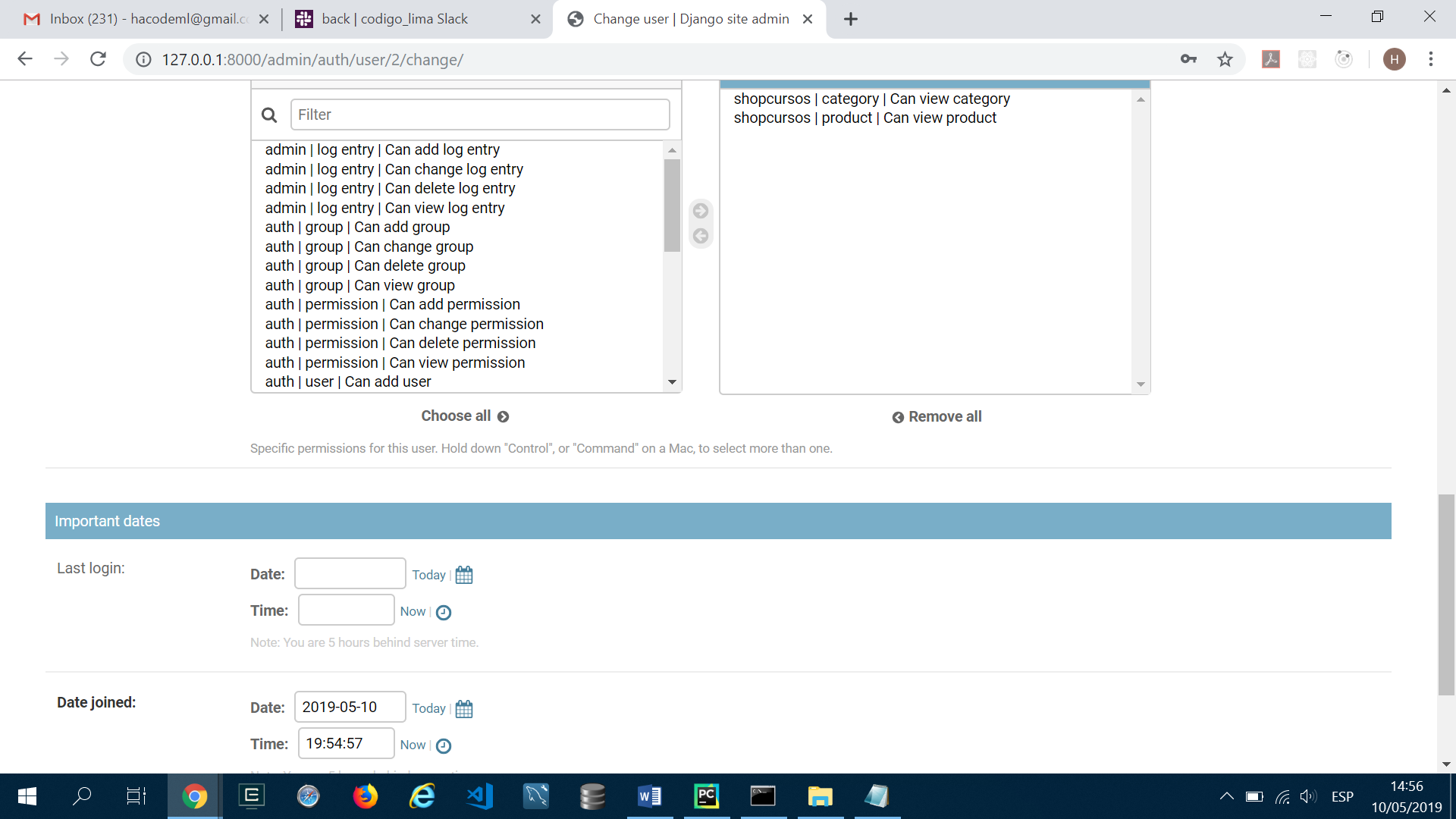
Creating USERS:

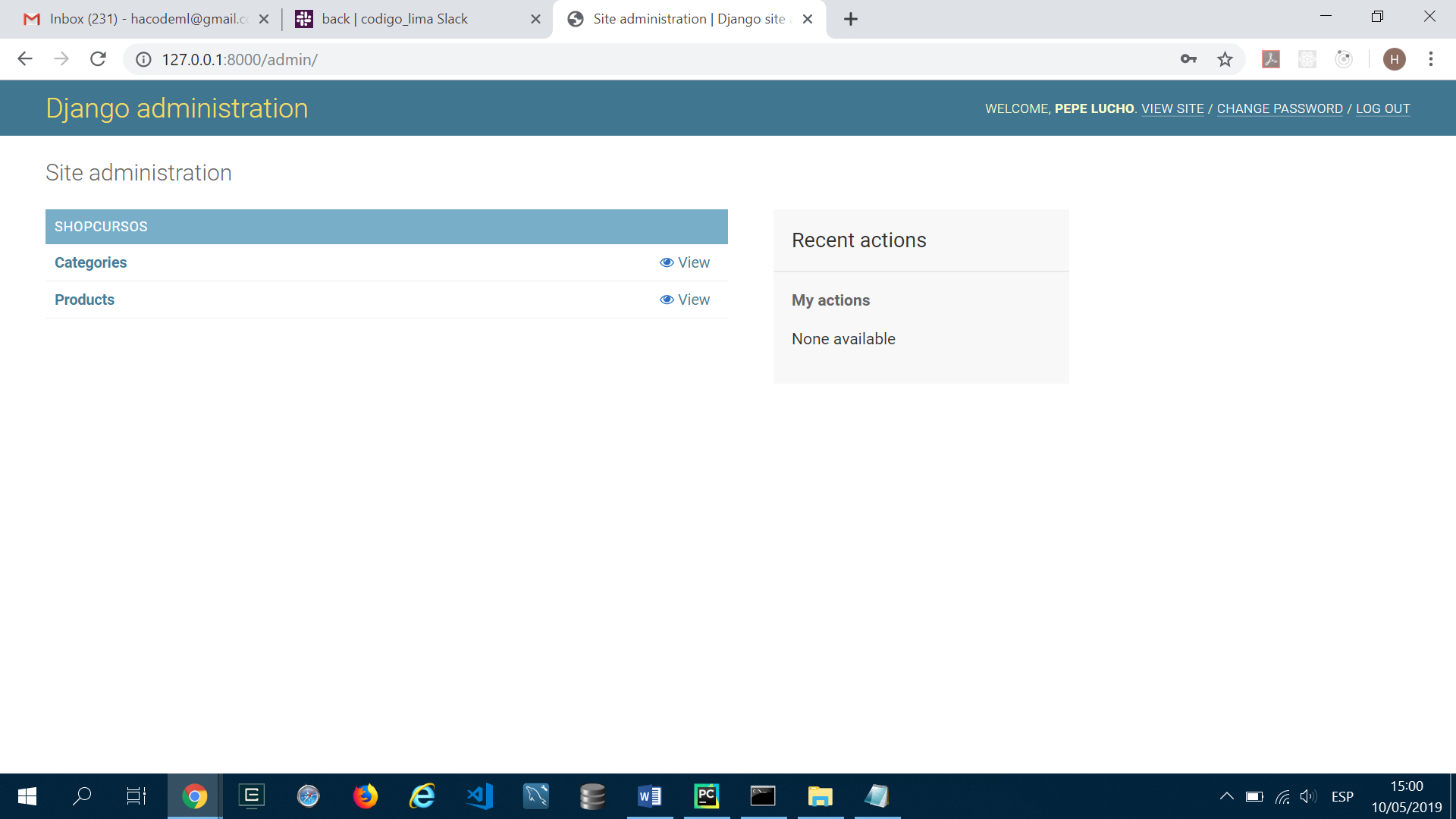


Pepelucho

money””;;

PERMISOS DE USUARIOS:





CURSOS EN UNA CATEGORÍA ESPECÍFICA

def get\_absolute\_url(self):  
 return reverse('shopcursos:product\_list\_by\_category'**,** args=[self.slug])

función que nos llevará a otra vista que filtre por el slug de la categoría que usemos.

DETALLE CURSO

def get\_absolute\_url(self):  
 return reverse('shopcursos:product\_detail'**,** args=[self.id**,** self.slug])

nos mostrará el curso sobre el q hagamos curso, para ello adicionamos como el id del curso y su slug.

categories = Category.objects.all()  
products = Product.objects.filter(available=True)

primero muestra todas las categorias y todos los cursos, hasta que no se le pase el slug como parámetro.

Si el curso no se encontró:

get\_object\_or\_404

context: es como el state de react, definimos el objeto que podremos renderizar en el template.

En template jalamos estos valores usando el comando:

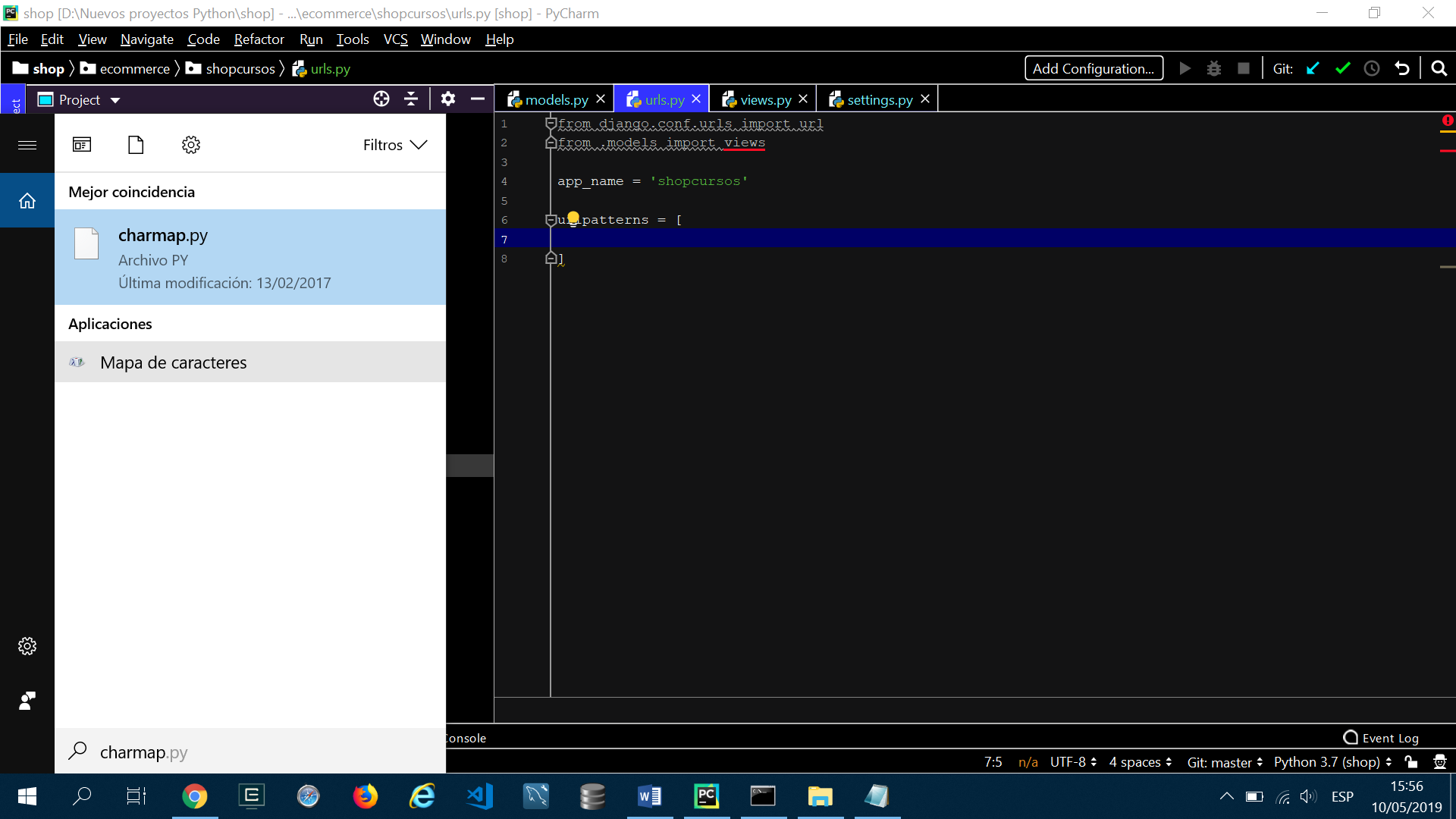
{% objects.all %}

Hasta ahora ya hemos creado nuestras vistas o funciones que conectan los modelos y que renderizan y envían el context o diccionario con toda la información que será mostrara en una página específica.

Ahora tenemos que configurar las urls.py dentro de la APP.

Importamos todas las vistas:

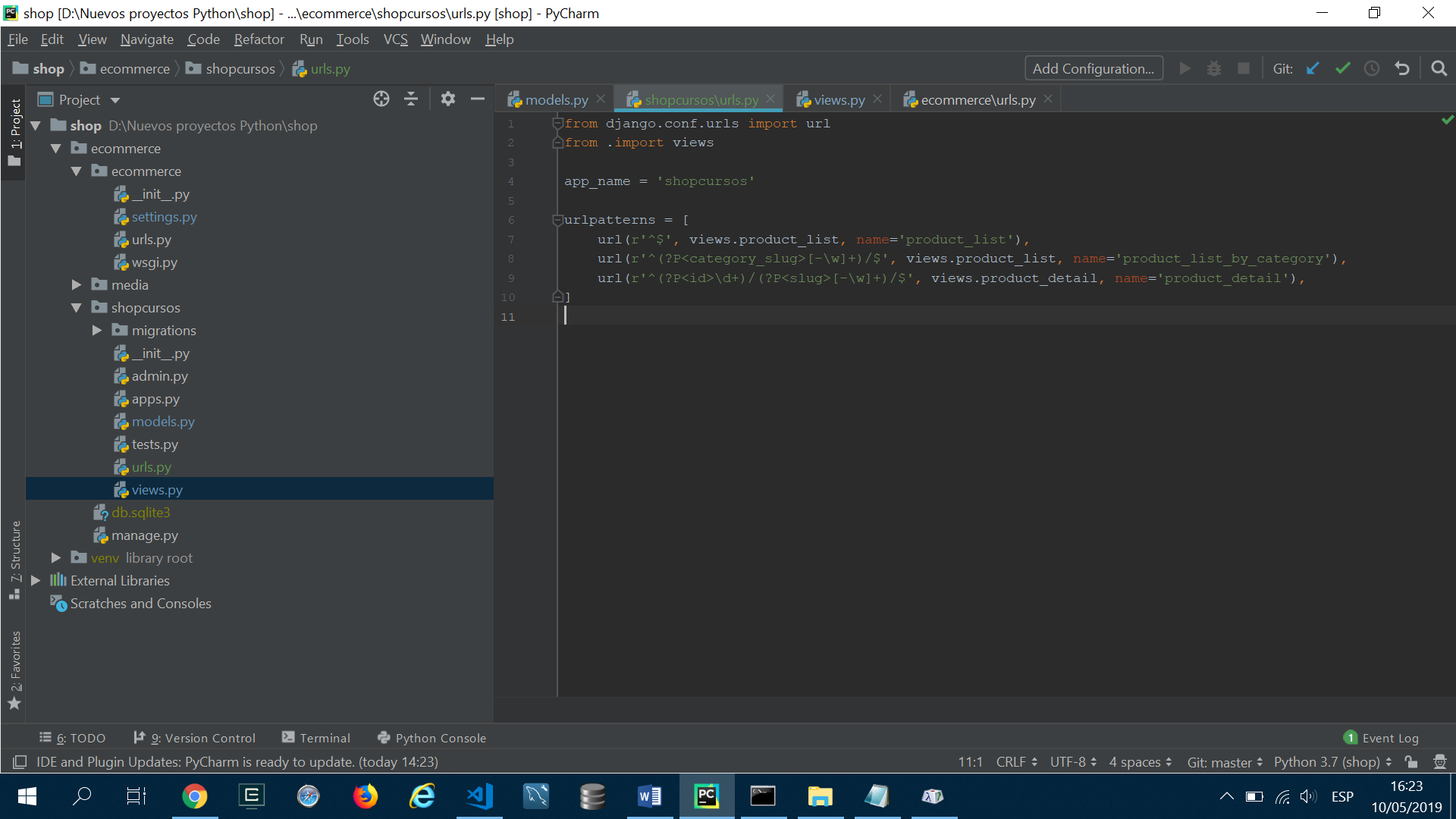
from .import views



DJANGO: regex

^ indica inicio de texto

$ indica fin de texto



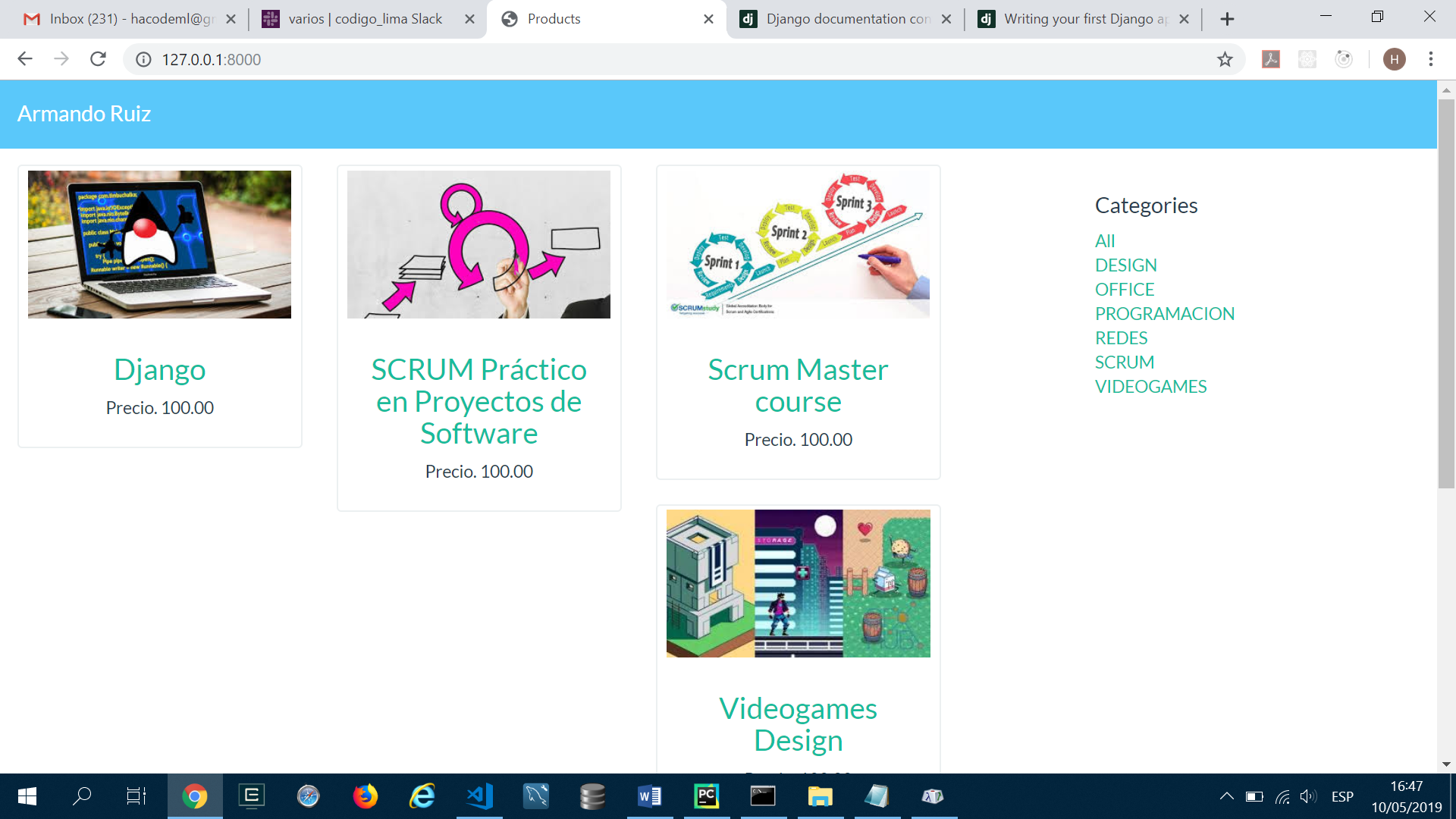
Agregar el namespace de la app a las urls del proyecto.

**Averiguar added in urls.py of Project.**

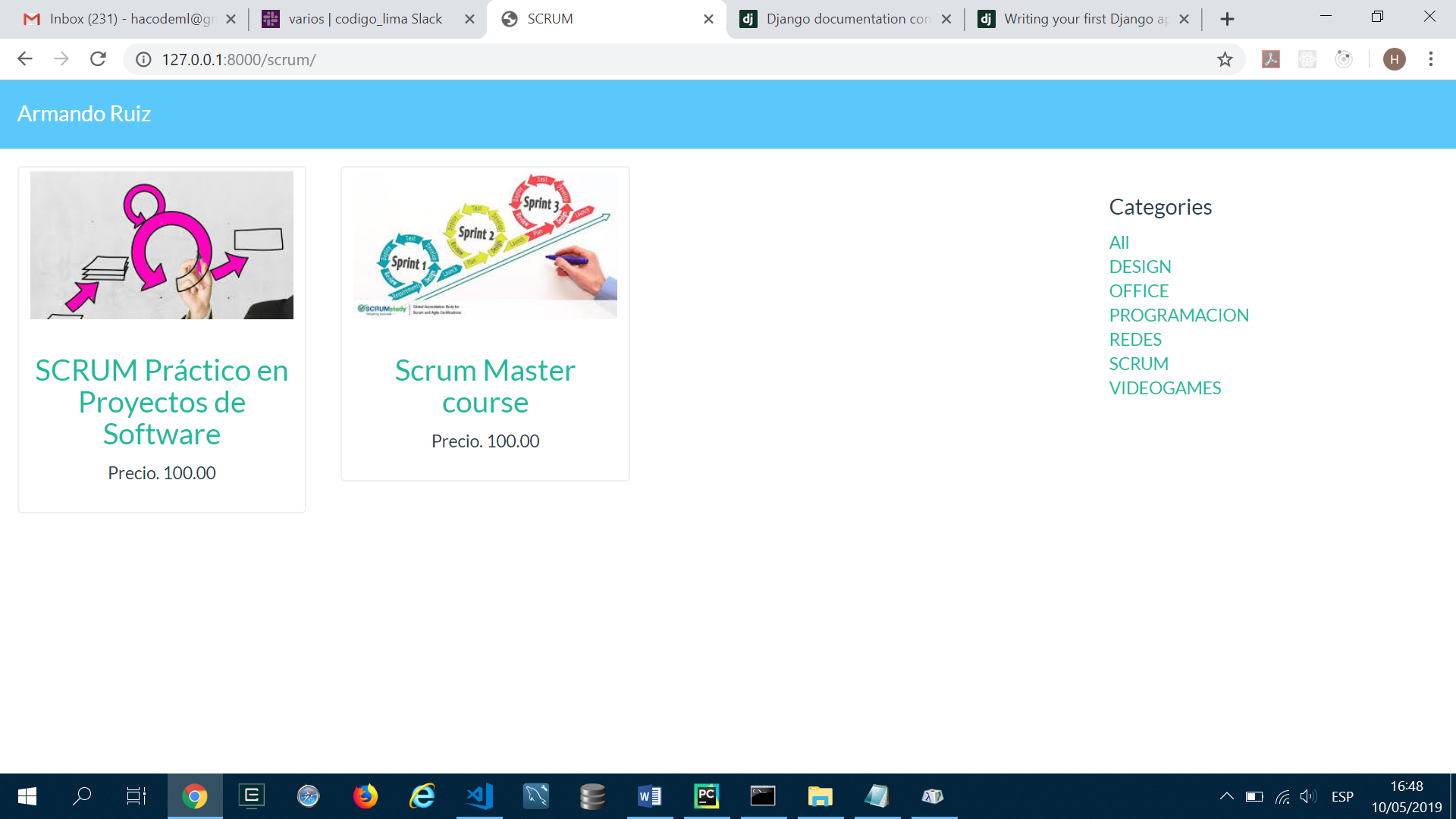
if settings.DEBUG:  
 urlpatterns += static(settings.MEDIA\_URL, document\_root=settings.MEDIA\_ROOT)

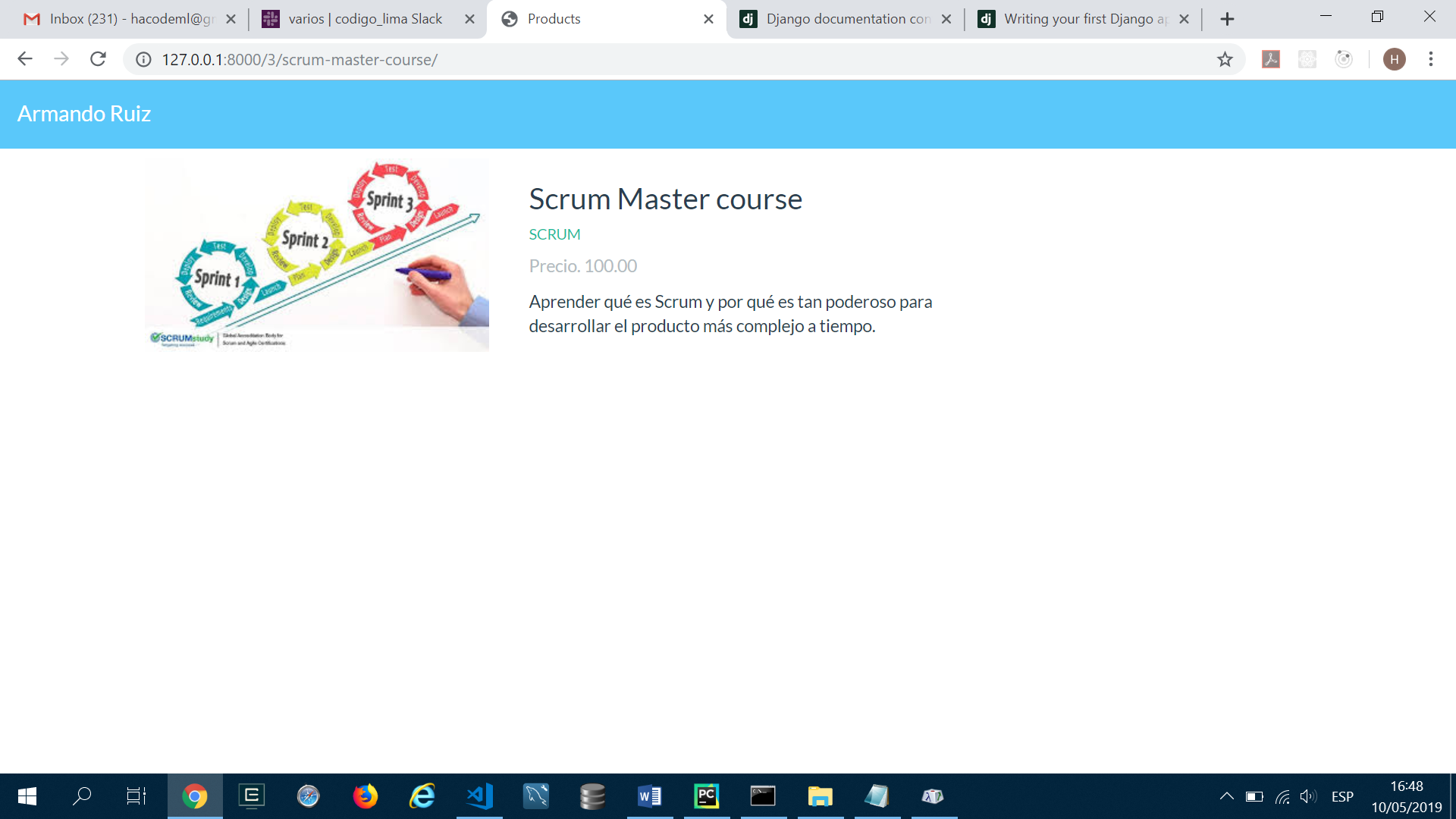
When running our server:

ALL COURSES



BY CATEGORY





Nos manda defrente a la pagina de cursos ya que está configurado en urls.py del proyecto como:

urlpatterns = [  
 path('admin/', admin.site.urls),  
 path('', include('shopcursos.urls')),  
]

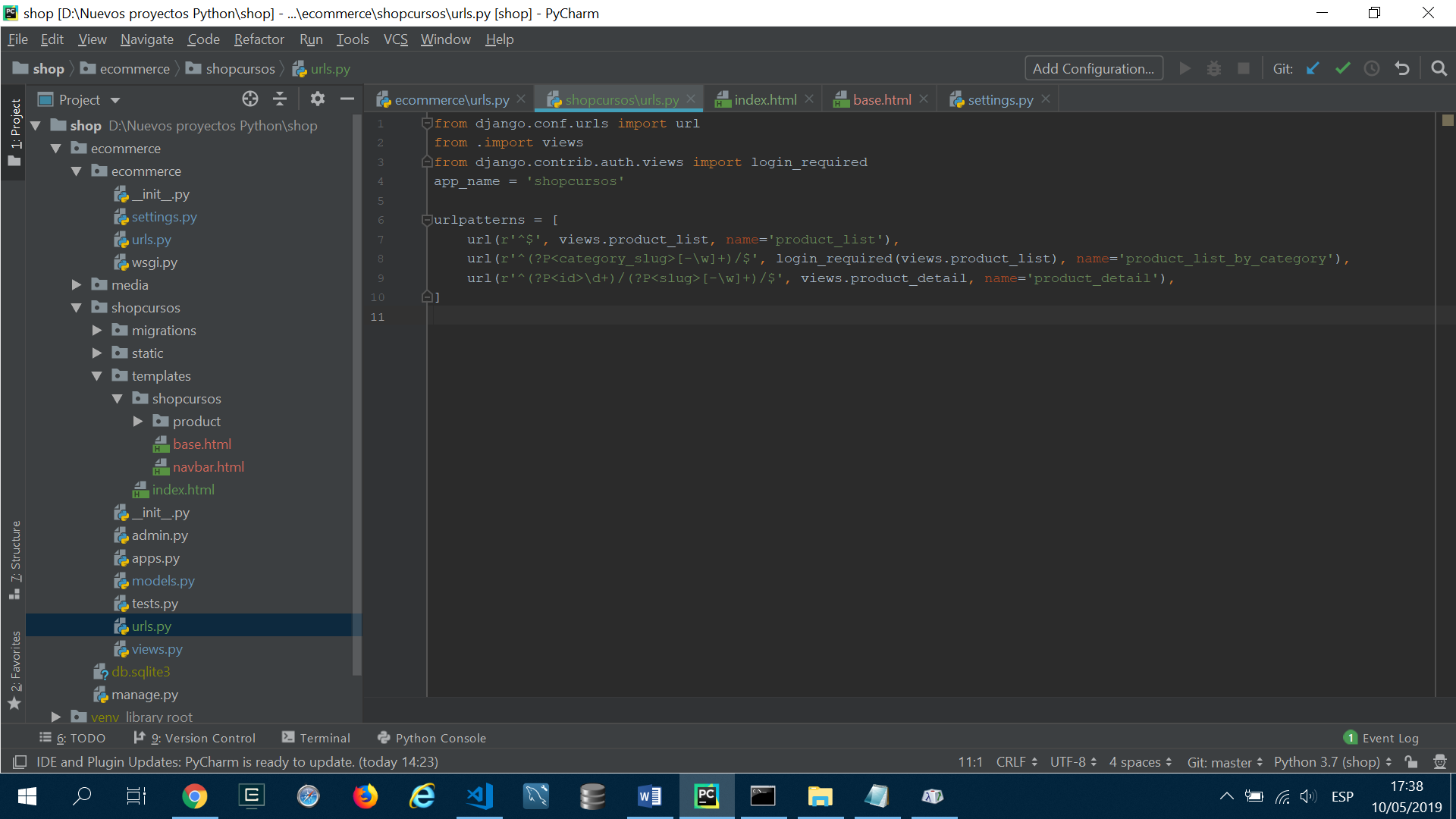
Es decir entrará al Segundo path luego a shopcursos y dentro de shopcursos:

urlpatterns = [  
 url(r'^$', views.product\_list, name='product\_list'),  
 url(r'^(?P<category\_slug>[-\w]+)/$', views.product\_list, name='product\_list\_by\_category'),  
 url(r'^(?P<id>\d+)/(?P<slug>[-\w]+)/$', views.product\_detail, name='product\_detail'),  
]

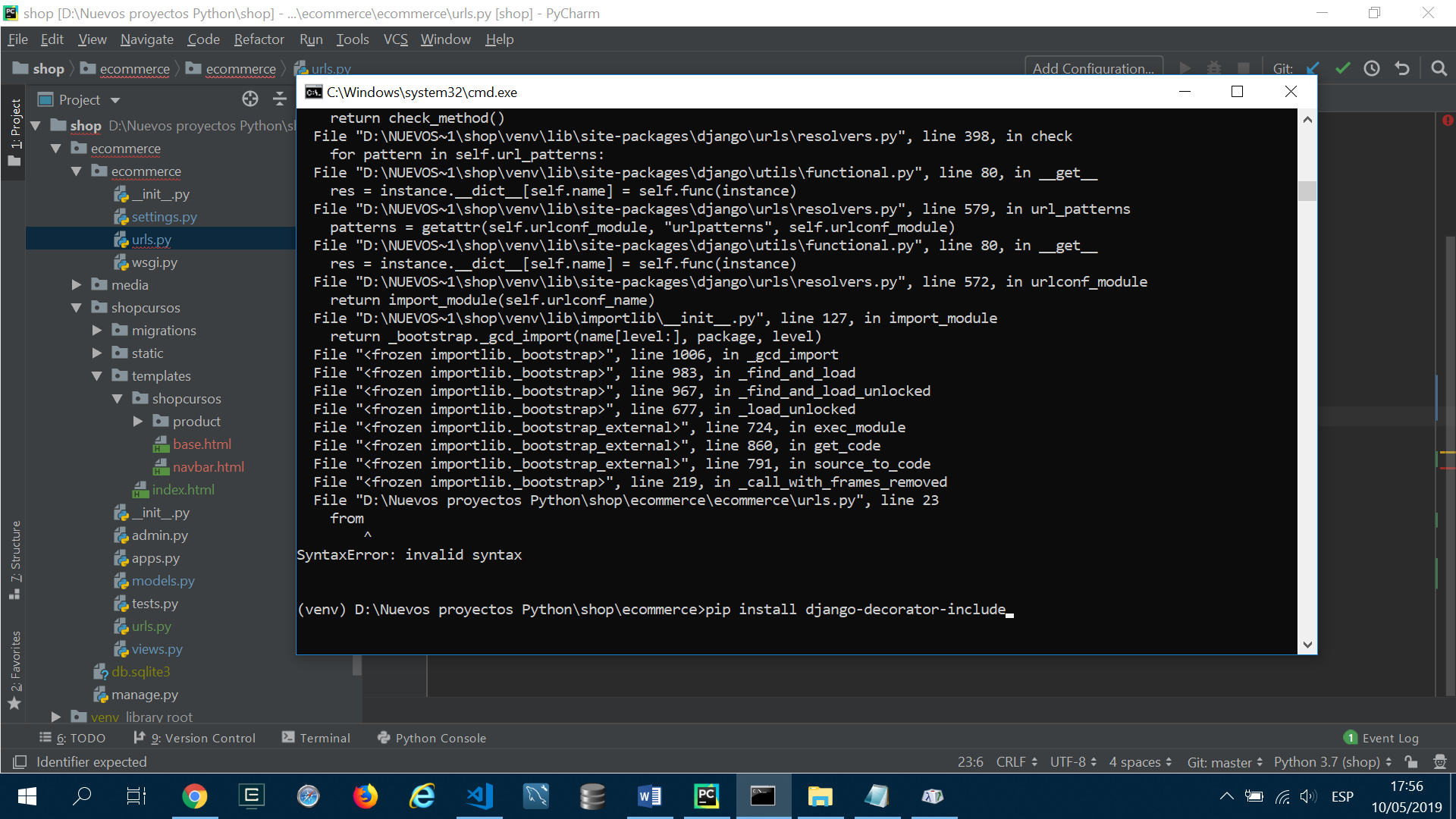
entrará al primer path que está vacío

LOGIN

RESTRINGIR ACCESO A URLS:



Decorators module:



@login\_required(login\_url='http://127.0.0.1:8000/login/')

from django.contrib.auth.decorators import login\_required