

# Lab4.

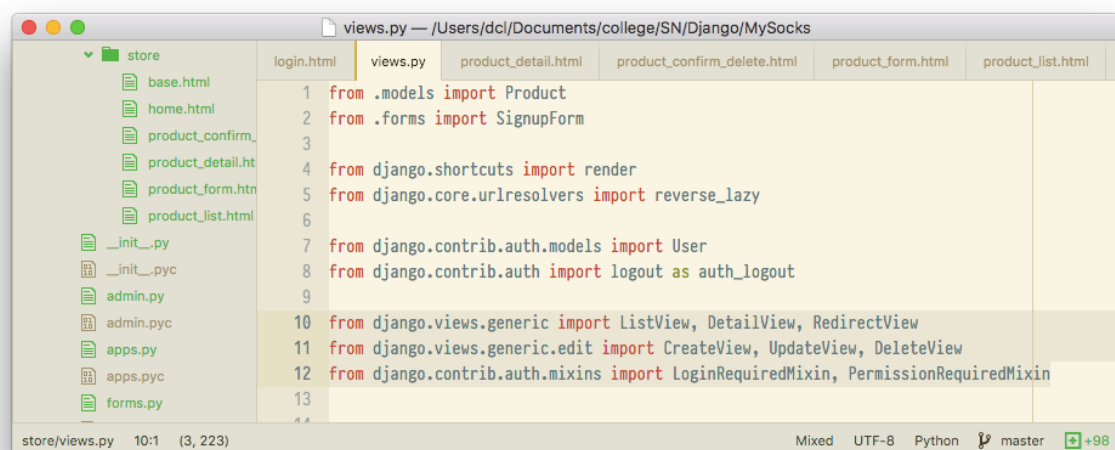
- **En el laboratorio anterior he:**

- Construido la vista de añadir productos y la vista de listado de productos.
- Además de modificar el "index" de forma que dé la bienvenida al usuario.

- **En este laboratorio he:**

- Reestructurado el proyecto para usar *Class Based Views*
- Añadido un control de usuarios; *Registro, Login, Logout*.
- Añadido vistas al producto; *Alta, Detallada, Modificación, Borrado, Paginado el listado*.
- Añadido un botón de Paypal.

## Uso de *Class Based Views* y *Mixin*.



```
1 from .models import Product
2 from .forms import SignUpForm
3
4 from django.shortcuts import render
5 from django.core.urlresolvers import reverse_lazy
6
7 from django.contrib.auth.models import User
8 from django.contrib.auth import logout as auth_logout
9
10 from django.views.generic import ListView, DetailView, RedirectView
11 from django.views.generic.edit import CreateView, UpdateView, DeleteView
12 from django.contrib.auth.mixins import LoginRequiredMixin, PermissionRequiredMixin
```

## Añadido un control de usuarios.

### *Registro.*

### Template.

The screenshot shows a code editor window titled "user\_form.html — /Users/dcl/Documents/college/SN/Django/MySocks". The left sidebar displays a file tree for the "MySocks" project, with the "auth" folder under "templates" selected. The main editor area shows the content of "user\_form.html":

```
1 {% extends 'store/base.html' %}
2 {% block content %}
3 <h2>{{title}}</h2><hr>
4 <form method="post" action="."> {% csrf_token %}
5     <p><label for="id_username">Username:</label> {{ form.username }}</p>
6     <p><label for="id_email">E-Mail:</label> {{ form.email }}</p>
7     <p><label for="id_password">Password:</label> {{ form.password1 }}</p>
8     <p><label for="id_password">P.Confirm:</label> {{ form.password2 }}</p><hr>
9     <input class="btn btn-success btn-block" type="submit" value="Register" />
10 </form>
11 {% endblock %}
12
```

The status bar at the bottom indicates the file path "store/templates/auth/user\_form.html", line 1:1, and encoding "UTF-8".

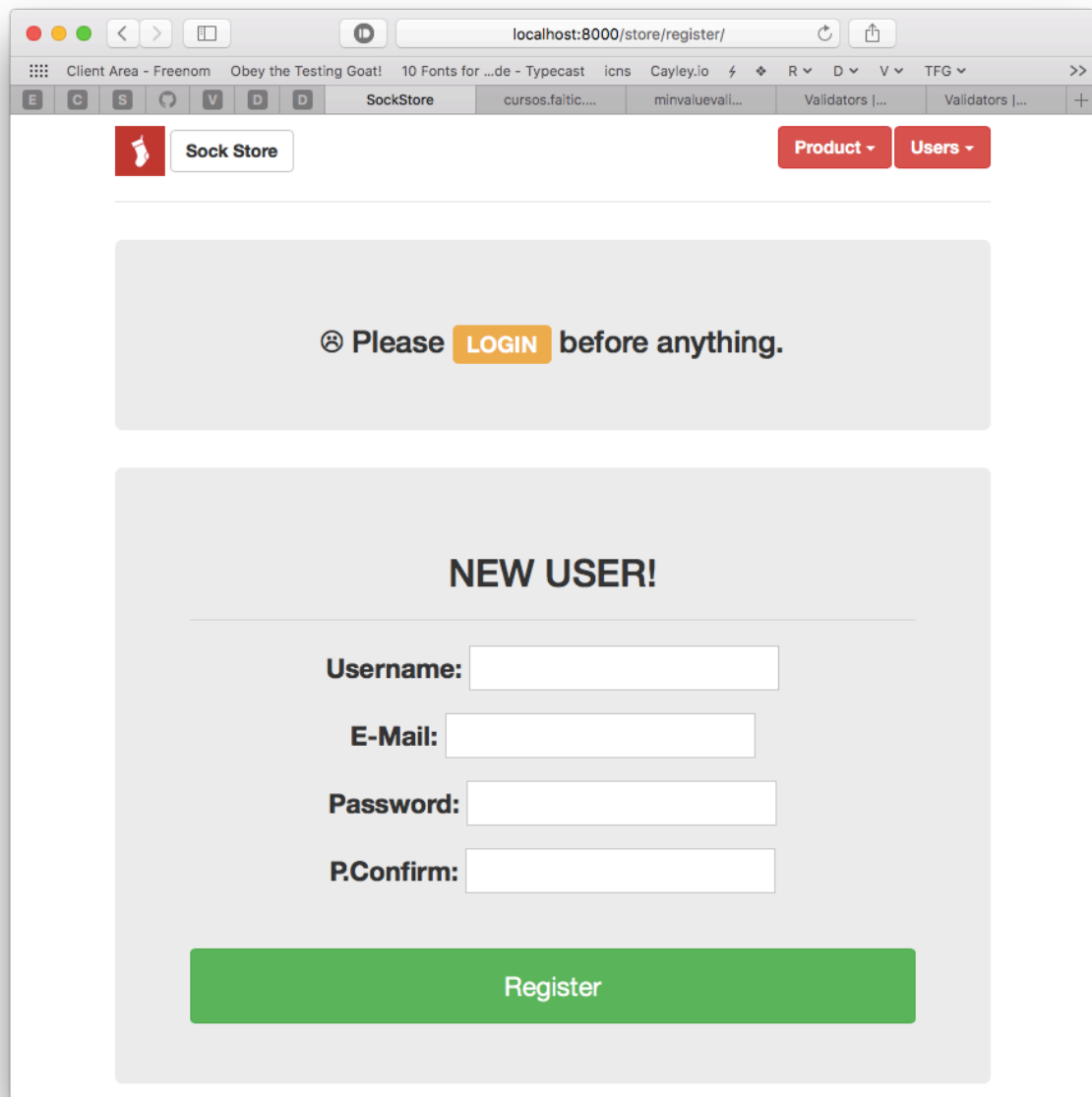
View.

The screenshot shows a code editor window titled "views.py — /Users/dcl/Documents/college/SN/Django/MySocks". The left sidebar displays the same file tree as the previous image, with the "store" folder under "templates" selected. The main editor area shows the content of "views.py":

```
15 # CLASS-BASED VIEWS.
16
17 # ... User
18 class UserCreate(CreateView):
19     model = User
20     form_class = SignupForm
21     success_url = reverse_lazy('store:product_list')
22
23     def get_context_data(self, **args):
24         context = super(UserCreate,
25                         self).get_context_data(**args)
26         context.update( { 'title': 'NEW USER!' } )
27         return context
28
```

The status bar at the bottom indicates the file path "store/views.py", line 1:1, and encoding "UTF-8".

Render.



***Login.***

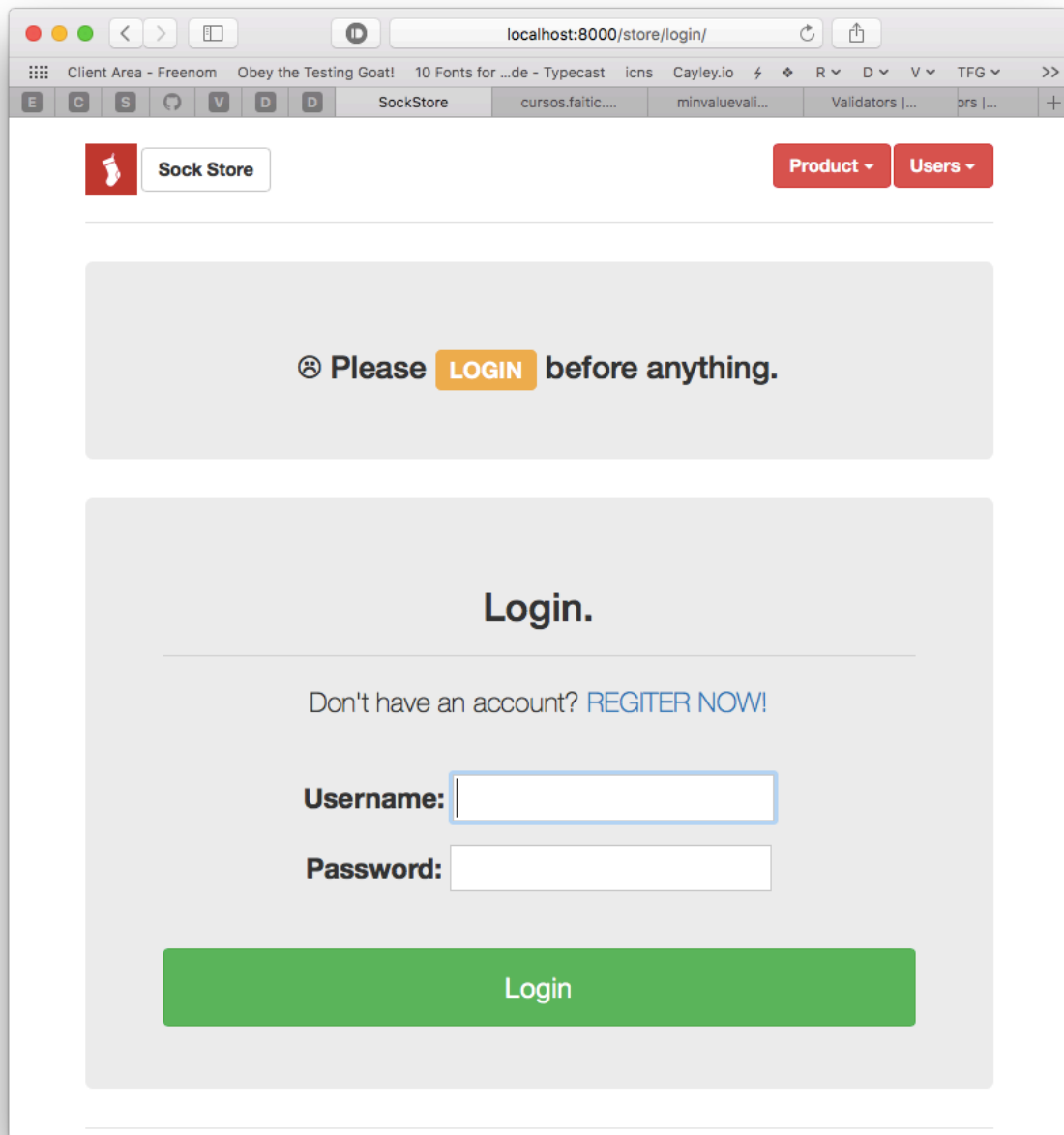
**Template.**

```
login.html
1 {% extends 'store/base.html' %}
2
3 {% block content %}
4
5 {% if user.is_authenticated %}
6 <h3>Sorry
7 <span class="label label-danger"> {{user.username}}</span>
8 you can't view this page.
9 </h3>
10
11 {% else %}
12 <h2>Login.</h2><hr>
13 {% if form.has_errors %} <p>ERROR: Please try again.</p> {% endif %} <!-- FALLA -->
14 <form method="post" action="."> {% csrf_token %}
15 <p>Don't have an account? <a href="{% url 'store:user_add' %}">REGISTER NOW!</a> </p><hr>
16 <p><label for="id_username">Username:</label> {{ form.username }}</p>
17 <p><label for="id_password">Password:</label> {{ form.password }}</p><hr>
18 <input type="hidden" name="next" value="{% if next %} '{{ next }}' {% else %} '/store' {% endif %}" />
19 <input class="btn btn-success btn-block" type="submit" value="Login" />
20 </form>
21
22 {% endif %}
23
24 {% endblock %}
```

View.

```
urls.py
1 from . import views
2 from django.conf.urls import include, url
3 from django.contrib.auth.views import login
4
5 app_name = 'store'
6
7 urlpatterns = [
8     url(r'^$', views.home, name='home'),
9     url(r'^login/$', login, name='login'),
10 ]
```

Render.



***Logout.***

**Template.**

The screenshot shows a code editor window titled "user\_form.html — /Users/dcl/Documents/college/SN/Django/MySocks". The left sidebar displays a file tree for the "MySocks" project, with the "auth" folder selected. The main editor area shows the content of "user\_form.html", which is a Django template. The template extends "store/base.html" and contains a form with fields for username, email, password, and password confirmation. A "Register" button is at the bottom of the form. The status bar at the bottom indicates the file is at line 1, column 1, and is part of a Django project.

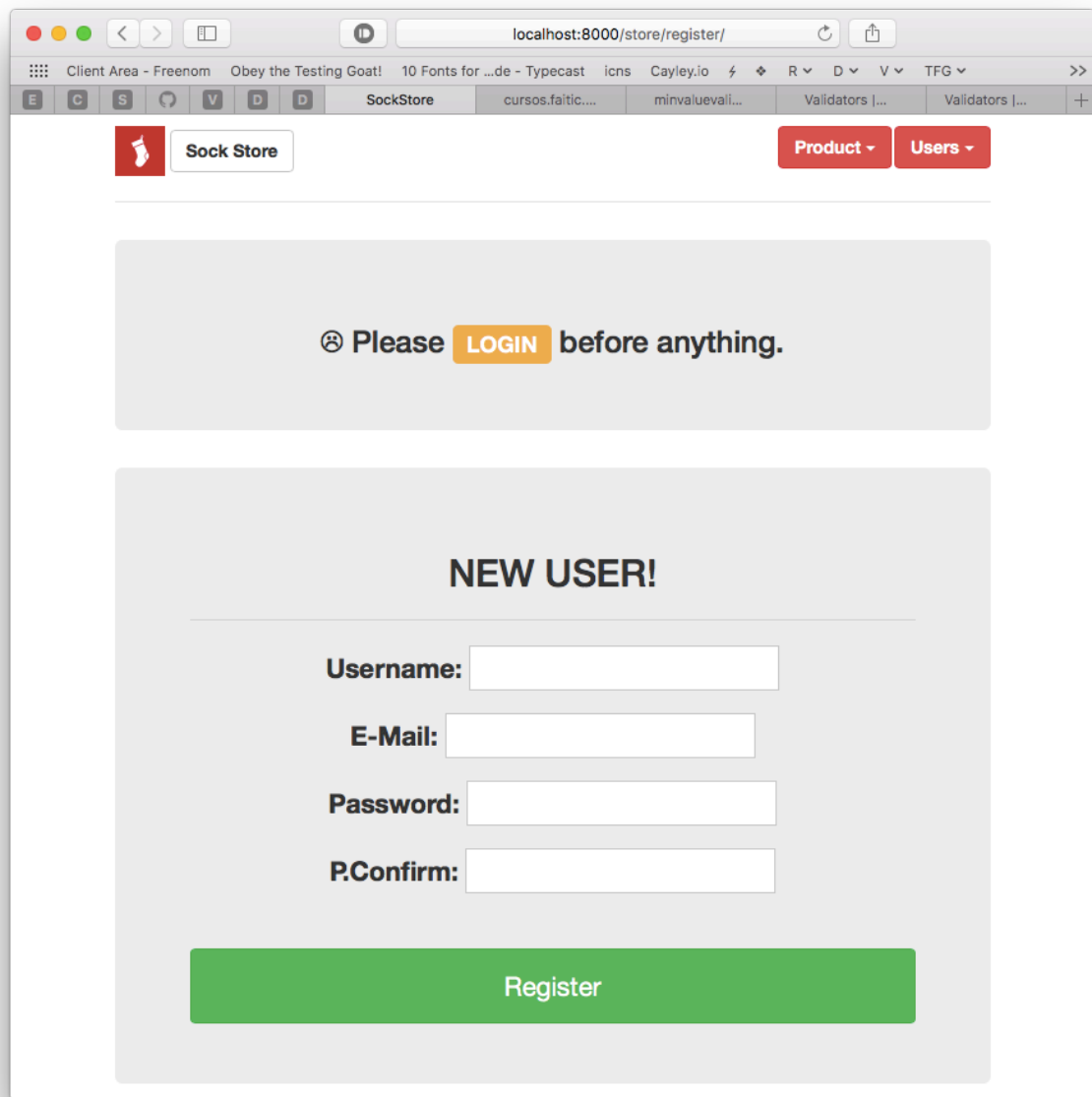
```
1 {% extends 'store/base.html' %}
2 {% block content %}
3 <h2>{{title}}</h2><hr>
4 <form method="post" action="."> {% csrf_token %}
5     <p><label for="id_username">Username:</label> {{ form.username }}</p>
6     <p><label for="id_email">E-Mail:</label> {{ form.email }}</p>
7     <p><label for="id_password">Password:</label> {{ form.password1 }}</p>
8     <p><label for="id_password">P.Confirm:</label> {{ form.password2 }}</p><hr>
9     <input class="btn btn-success btn-block" type="submit" value="Register" />
10 </form>
11 {% endblock %}
12
```

View.

The screenshot shows a code editor window titled "views.py — /Users/dcl/Documents/college/SN/Django/MySocks". The left sidebar displays the same file tree as the previous image, with the "store" folder selected. The main editor area shows the content of "views.py", which defines a "UserCreate" class that inherits from "CreateView". The class sets the model to "User", the form class to "SignupForm", and the success URL to "reverse\_lazy('store:product\_list')". The "get\_context\_data" method is overridden to update the context with a title of "NEW USER!". The status bar at the bottom indicates the file is at line 1, column 1, and is part of a Django project.

```
15 # CLASS-BASED VIEWS.
16
17 # ... User
18 class UserCreate(CreateView):
19     model = User
20     form_class = SignupForm
21     success_url = reverse_lazy('store:product_list')
22
23     def get_context_data(self, **args):
24         context = super(UserCreate,
25                         self).get_context_data(**args)
26         context.update( { 'title': 'NEW USER!' } )
27         return context
28
```

Render.



## Añadido vistas al producto

***Detallada.***

**Template.**

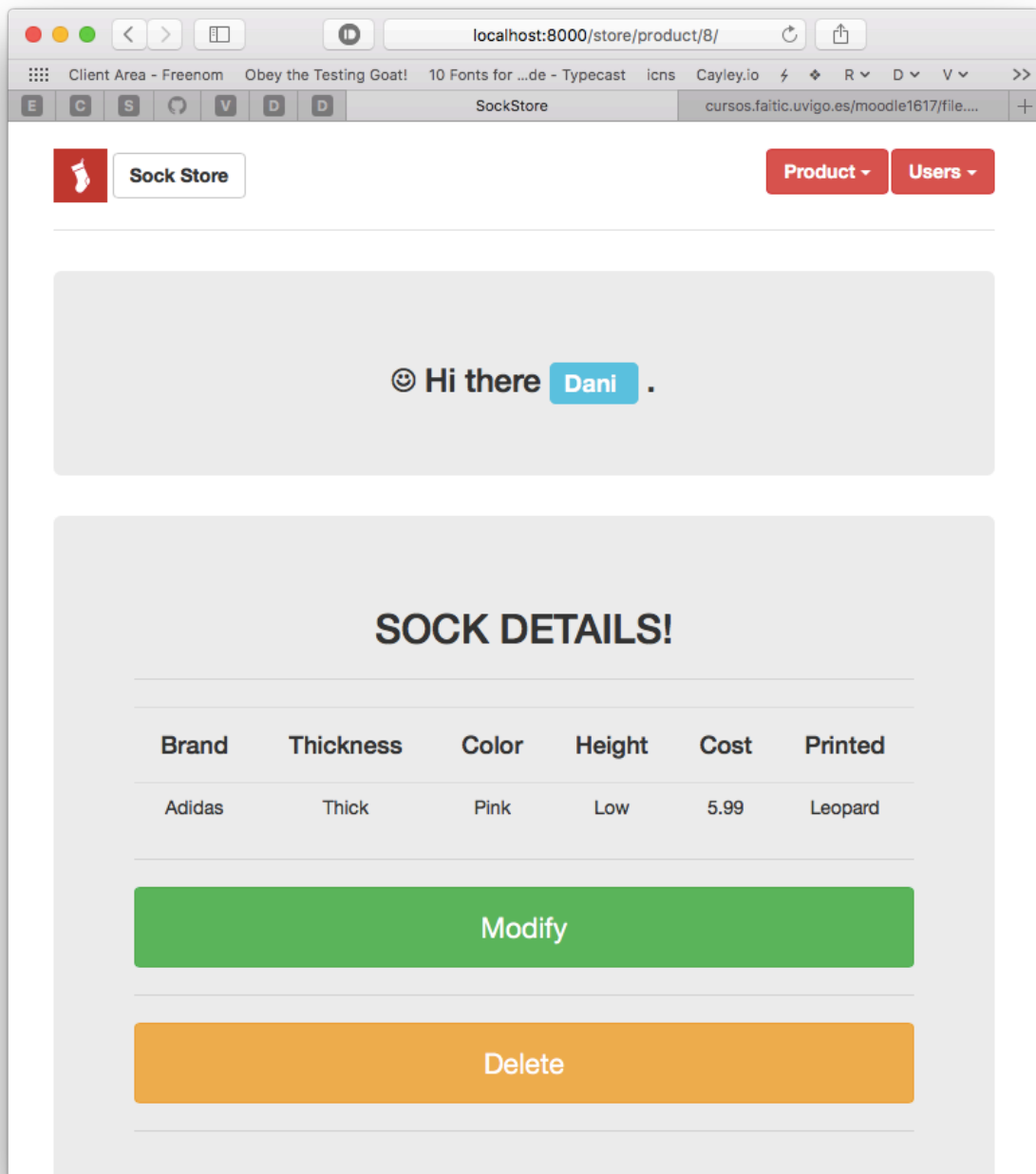
```
1 {% extends 'store/base.html' %}
2 {% block content %}
3 <h2>{{title}}</h2><hr>
4 <table class="table">
5   <tr>
6     <td> <h4> Brand </h4> </td>
7     <td> <h4> Thickness </h4> </td>
8     <td> <h4> Color </h4> </td>
9     <td> <h4> Height </h4> </td>
10    <td> <h4> Cost </h4> </td>
11    <td> <h4> Printed </h4> </td>
12  </tr>
13  <tr>
14    <td> {{ product.brand }} </td>
15    <td> {{ product.thickness }} </td>
16    <td> {{ product.color }} </td>
17    <td> {{ product.height }} </td>
18    <td> {{ product.cost }} </td>
19    <td> {{ product.printed }} </td>
20  </tr>
21 </table>
22 {% if user.is_superuser %}
23 <hr><a class="btn btn-success btn-block" href="{% url 'store:product_update' product.id %}"> Modify </a>
24 <hr><a class="btn btn-warning btn-block" href="{% url 'store:product_delete' product.id %}"> Delete </a><hr>
25 {% endif %}
26 {% endblock %}
```

View.

```
39
40 class ProductDetail(LoginRequiredMixin, DetailView):
41     model = Product
42     login_url = '/store/login/'
43
44     def get_context_data(self, **args):
45         context = super(ProductDetail,
46                         self).get_context_data(**args)
47         context.update( { 'title':'SOCK DETAILS!' } )
48         return context
```

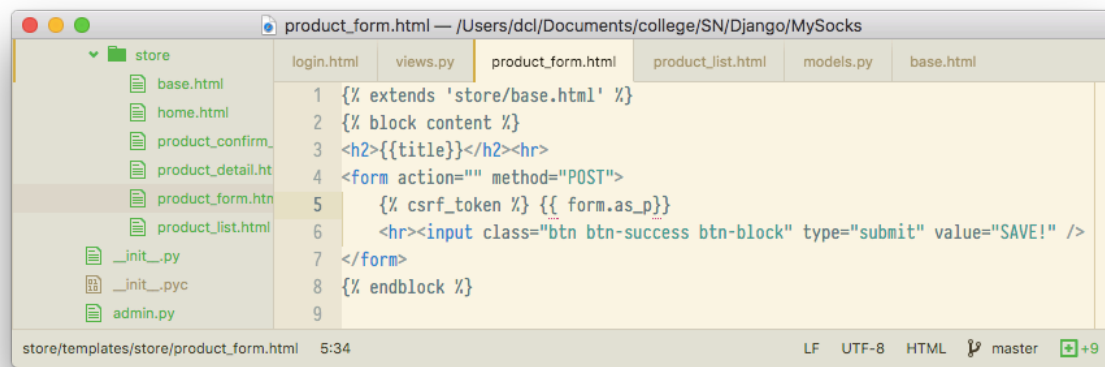
Render.





***Alta.***

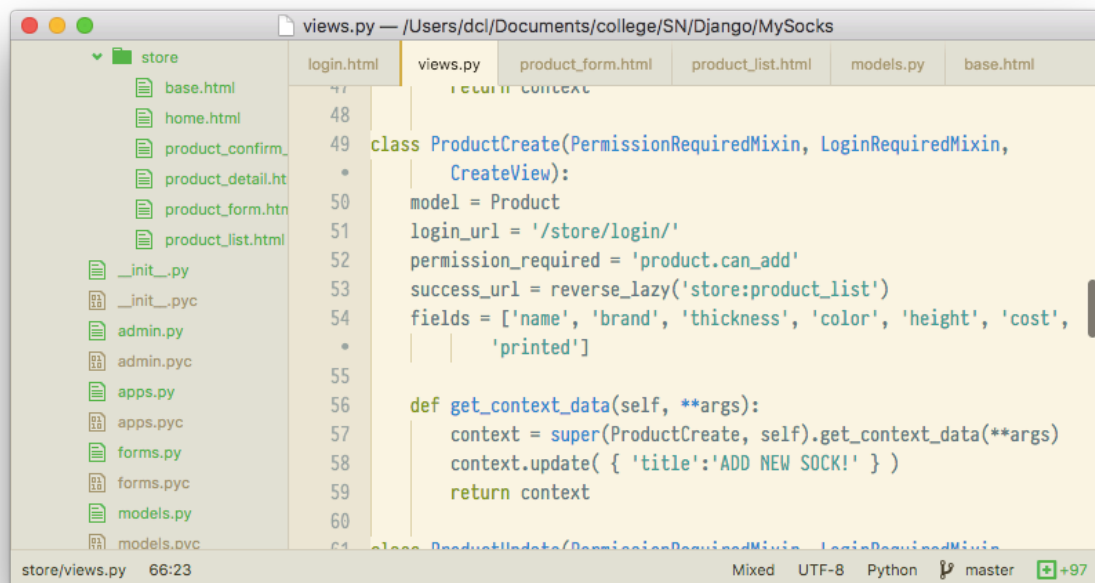
**Template.**



The screenshot shows a code editor window titled "product\_form.html — /Users/dcl/Documents/college/SN/Django/MySocks". The left sidebar displays a file tree for a project named "store", with files including base.html, home.html, product\_confirm\_\*.html, product\_detail.html, product\_form.html (selected), product\_list.html, \_\_init\_\_.py, \_\_init\_\_.pyc, and admin.py. The main editor area shows the content of product\_form.html, which extends the store/base.html template. It includes a title block and a form with a "SAVE!" button. The status bar at the bottom indicates the file path "store/templates/store/product\_form.html", the time "5:34", and encoding "UTF-8".

```
1 {% extends 'store/base.html' %}
2 {% block content %}
3 <h2>{{title}}</h2><hr>
4 <form action="" method="POST">
5     {% csrf_token %} {{ form.as_p }}
6     <hr><input class="btn btn-success btn-block" type="submit" value="SAVE!" />
7 </form>
8 {% endblock %}
9
```

## View.



The screenshot shows a code editor window titled "views.py — /Users/dcl/Documents/college/SN/Django/MySocks". The left sidebar shows the same file tree as the previous image, with views.py selected. The main editor area shows the Python code for the ProductCreate class, which inherits from PermissionRequiredMixin, LoginRequiredMixin, and CreateView. It defines the model as Product, the login URL as '/store/login/', the permission as 'product.can\_add', and the success URL as the reverse\_lazy('store:product\_list'). It also defines the fields for the form. The get\_context\_data method is overridden to update the context with a title 'ADD NEW SOCK!'. The status bar at the bottom indicates the file path "store/views.py", the time "66:23", and encoding "UTF-8".

```
47 return context
48
49 class ProductCreate(PermissionRequiredMixin, LoginRequiredMixin,
50                     CreateView):
51     model = Product
52     login_url = '/store/login/'
53     permission_required = 'product.can_add'
54     success_url = reverse_lazy('store:product_list')
55     fields = ['name', 'brand', 'thickness', 'color', 'height', 'cost',
56             'printed']
57
58     def get_context_data(self, **args):
59         context = super(ProductCreate, self).get_context_data(**args)
60         context.update( { 'title': 'ADD NEW SOCK!' } )
61         return context
62
63 class ProductUpdate(PermissionRequiredMixin, LoginRequiredMixin,
```

## Render.

localhost:8000/store/product/add/

Client Area - Freenom Obey the Testing Goat! 10 Fonts for ...de - Typecast icns Cayley.io

SockStore cursos.faitic.uvigo.es/moo...

**Sock Store** Product Users

Hi there **Dani**.

### ADD NEW SOCK!

Name:

Brand:

Thickness:

Color:

Height:

Cost:

Printed:

**SAVE!**

***Modificación.***

**Template.**

The screenshot shows a code editor window titled "product\_form.html — /Users/dcl/Documents/college/SN/Django/MySocks". The left sidebar displays a file tree for the "store" directory, including files like base.html, home.html, product\_confirm.html, product\_detail.html, product\_form.html (selected), product\_list.html, \_\_init\_\_.py, \_\_init\_\_.pyc, and admin.py. The main editor area shows the content of product\_form.html, which extends the store/base.html template. It contains a block of content with an h2 tag for the title, followed by a form with a POST method and a CSRF token. The form includes a submit button labeled "SAVE!". The status bar at the bottom indicates the file is "store/templates/store/product\_form.html", the cursor is at line 5, column 34, and the encoding is UTF-8.

```
1 {% extends 'store/base.html' %}
2 {% block content %}
3 <h2>{{title}}</h2><hr>
4 <form action="" method="POST">
5     {% csrf_token %} {{ form.as_p }}
6     <hr><input class="btn btn-success btn-block" type="submit" value="SAVE!" />
7 </form>
8 {% endblock %}
9
```

View.

The screenshot shows a code editor window titled "views.py — /Users/dcl/Documents/college/SN/Django/MySocks". The left sidebar displays a file tree for the "store" directory, including files like base.html, home.html, product\_confirm.html, product\_detail.html, product\_form.html, product\_list.html, \_\_init\_\_.py, \_\_init\_\_.pyc, admin.py, admin.pyc, apps.py, apps.pyc, forms.py, forms.pyc, and models.py. The main editor area shows the content of views.py, which defines a ProductUpdate class. This class inherits from PermissionRequiredMixin, LoginRequiredMixin, and UpdateView. It sets the model to Product, the login\_url to '/store/login/', the permission\_required to 'product.can\_change', and the success\_url to reverse\_lazy('store:product\_list'). The fields attribute lists 'name', 'brand', 'thickness', 'color', 'height', 'cost', and 'printed'. The get\_context\_data method is overridden to update the context with a title of 'MODIFY SOCK!'. The status bar at the bottom indicates the file is "store/views.py", the cursor is at line 66, column 23, and the encoding is UTF-8.

```
61 class ProductUpdate(PermissionRequiredMixin, LoginRequiredMixin,
62                     UpdateView):
63     model = Product
64     login_url = '/store/login/'
65     permission_required = 'product.can_change'
66     success_url = reverse_lazy('store:product_list')
67     fields = ['name', 'brand', 'thickness', 'color', 'height',
68             'cost', 'printed']
69
70     def get_context_data(self, **args):
71         context = super(ProductUpdate,
72                         self).get_context_data(**args)
73         context.update( { 'title':'MODIFY SOCK!' } )
74         return context
```

Render.

The screenshot shows a web browser window with the address bar displaying `localhost:8000/store/product/1/update`. The browser's tab bar shows several tabs, including 'Client Area - Freenom', 'Obey the Testing Goat!', '10 Fonts for ...de - Typecast', 'icns', 'Cayley.io', and 'cursos.faitic.uvigo.es/mood...'. The active tab is 'SockStore'.

The application interface features a top navigation bar with a 'Sock Store' logo on the left and two red buttons labeled 'Product' and 'Users' on the right. Below the navigation bar, a light gray box contains a greeting: 'Hi there Dani'.

The main content area is a light gray box titled 'MODIFY SOCK!'. It contains a form with the following fields:

- Name:** HelloKitty
- Brand:** Nike
- Thickness:** Thick (dropdown menu)
- Color:** Pink
- Height:** Mid (dropdown menu)
- Cost:** 9,99 (text input with a small up/down arrow icon)
- Printed:** Cats

At the bottom of the form is a large green button labeled 'SAVE!'.

***Borrado.***

**Template.**

The screenshot shows a code editor window titled "product\_confirm\_delete.html" with the file path "/Users/dcl/Documents/college/SN/Django/MySocks". The editor displays the Django template for the product confirmation delete view. The template extends "store/base.html" and contains a form with a "DELETE" button and a "CANCEL" link. The file explorer on the left shows the project structure, including the "store" directory with templates and the "admin" directory with Python files. The status bar at the bottom indicates the file is "store/templates/store/product\_confirm\_delete.html" with 7 lines of code, using LF line endings, UTF-8 encoding, and HTML syntax highlighting.

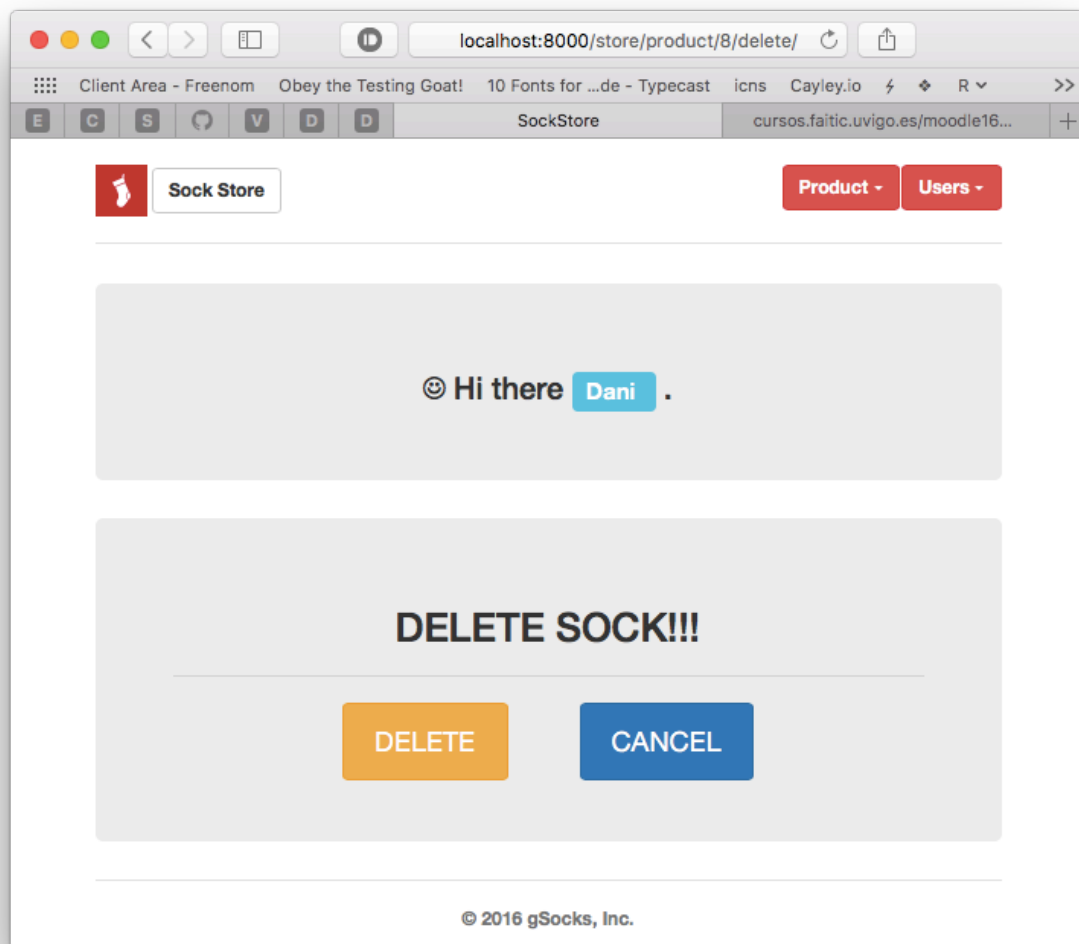
```
1 {% extends 'store/base.html' %}
2 {% block content %}
3 <h2>{{title}}</h2><hr>
4 <form action="" method="POST">
5     {% csrf_token %}
6     <input class="btn btn-warning" type="submit" value="DELETE" />
7     <a class="btn btn-primary" href="{% url 'store:product_list' %}"> CANCEL </a>
8 </form>
9 {% endblock content %}
10
```

View.

The screenshot shows a code editor window titled "views.py" with the file path "/Users/dcl/Documents/college/SN/Django/MySocks". The editor displays the Django view for the product delete action. The view class "ProductDelete" inherits from "PermissionRequiredMixin", "LoginRequiredMixin", and "DeleteView". It sets the model to "Product", the login URL to "/store/login/", the permission required to "product.can\_delete", and the success URL to "reverse\_lazy('store:product\_list')". The "get\_context\_data" method is overridden to update the context with a title "DELETE SOCK!!!". The file explorer on the left shows the project structure, including the "store" directory with templates and the "admin" directory with Python files. The status bar at the bottom indicates the file is "store/views.py" with 66 lines of code, using Mixed line endings, UTF-8 encoding, and Python syntax highlighting.

```
73
74 class ProductDelete(PermissionRequiredMixin,
75                     LoginRequiredMixin, DeleteView):
76     model = Product
77     login_url = '/store/login/'
78     permission_required = 'product.can_delete'
79     success_url = reverse_lazy('store:product_list')
80
81     def get_context_data(self, **args):
82         context = super(ProductDelete,
83                         self).get_context_data(**args)
84         context.update( { 'title':'DELETE SOCK!!!' } )
85         return context
```

Render.



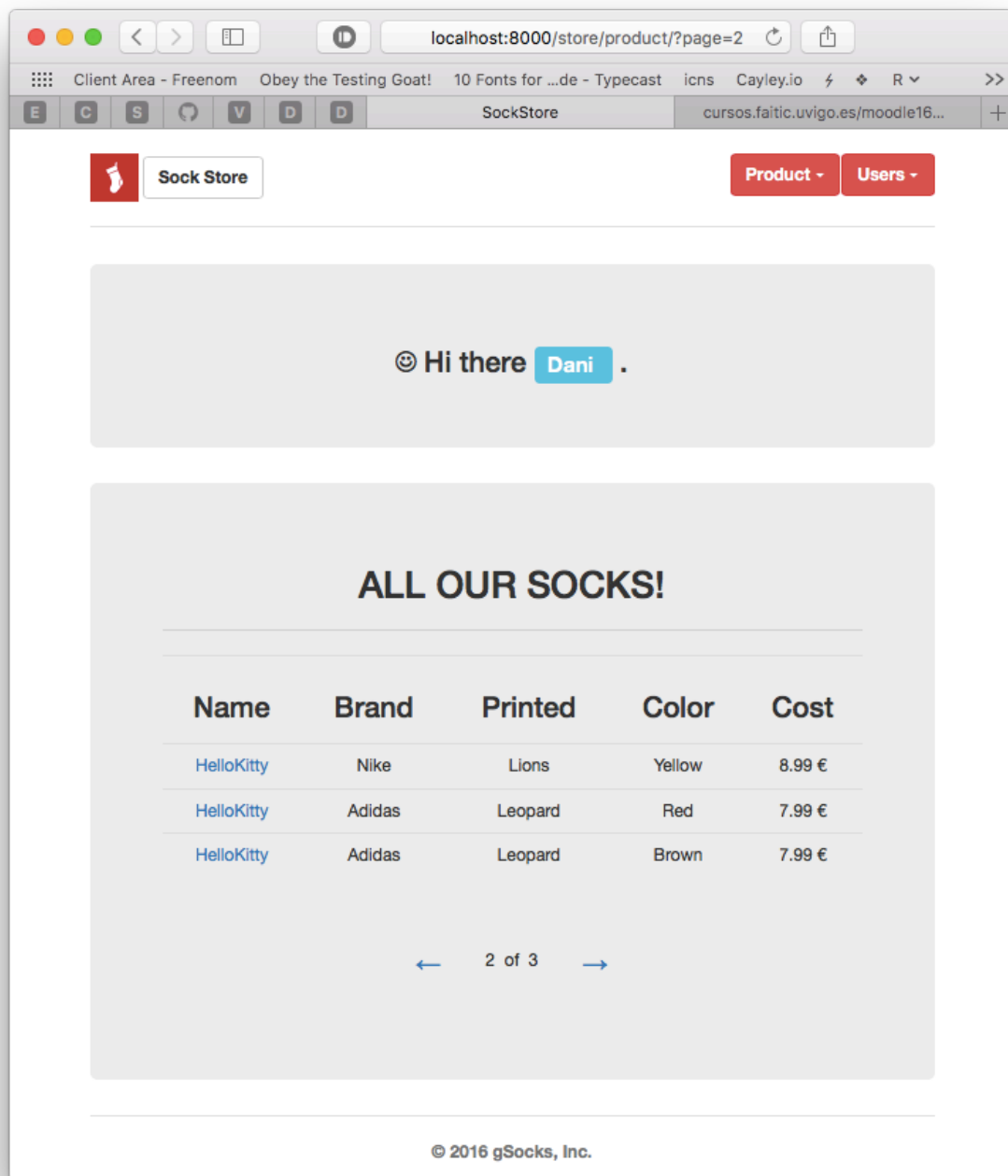
***Paginación.***

**Template.**



## Render.





**Añadido un botón de Paypal.**

**Template.**

The image shows a code editor window titled 'base.html' with the file path '/Users/dcl/Documents/college/SN/Django/MySocks'. The editor displays the Django template 'base.html' with line numbers 115 to 138. The template includes a page content section, a footer section, and a form for donating. The form has a select menu for donation amounts, a currency dropdown, and a submit button. The footer section contains a copyright notice for 2016 gSocks, Inc. and a PayPal donation link.

```
115 <!-- Page Content -->
116 <div class="jumbotron">{% block content %}{% endblock %}</div>
117
118 <!-- Page Footer -->
119 <footer class="footer">
120
121   <p style="text-align: center;"><b>&copy; 2016 gSocks, Inc.</b></p><hr>
122   <form style="text-align: center;" target="paypal" action="https://www.paypal.com/cgi-bin/webscr" method="post">
123     <input type="hidden" name="cmd" value="_s-xclick">
124     <input type="hidden" name="hosted_button_id" value="AFJ94EYJHFX7W">
125     <input type="hidden" name="on0" value="Would you like donate?">Would you like donate?<br>
126     <select name="os0">
127       <option value="A coffee.">A coffee. €2,00 EUR</option>
128       <option value="A travel.">A travel. €100,00 EUR</option>
129       <option value="All my money.">All my money. €500,00 EUR</option>
130     </select>
131     <br><br><input type="hidden" name="currency_code" value="EUR">
132     <input type="image" src="https://www.paypalobjects.com/en_US/i/btn/btn_cart_SM.gif" border="0" name="submit">
133     
134   </form>
135 </footer>
136
137 </div> <!-- Container end -->
138 </body> <!-- Body end -->
139 </html> <!-- Html end -->
```

Render.

