Solutions

Suppression développeur

> `developer/views.py`

```
def delete(request, developer_id):
 get_object_or_404(Developer, pk=developer_id).delete()

 return HttpResponseRedirect(reverse('developer:index'))
```

> `developer/urls.py`

```
 path('<int:developer_id>/delete', views.delete, name='delete'),
```

Il faut aussi modifier le modèle afin que la suppression d'un développeur n'implique pas la suppression de la tâche.

> `developer/models.py`

```
assignee = models.ForeignKey(
     Developer,
     related_name="tasks",
     on_delete=models.SET_NULL,
     null=True,
     verbose_name="assignee")
```

⚠️ Attention, il est nécessaire de réaliser une migration !

App Task - réponse

1. Création de l'app task `python manage.py startapp task`.
2. On ajoute cette application aux applications installée (dans `settings.py`).
3. On copie le modèle dans l'application task
4. On ajoute l'import de `developer` : `from developer.models import Developer`.
5. On réinitialise la DB `python manage.py migrate developer zero`
6. On supprime l'historique de migration (fichier migrations)
7. On lance la commande `python manage.py makemigrations`
8. On réalise la migration `python manage.py migrate`

Ajout de la liste des tâches - réponse

1. Commençons par écrire un nouveau gabarit `task/index.html` au sein du dossier `templates` que l'on ajoute à l'application

```
{% extends "_base.html" %}

{% block title %}MProject - tâches{% endblock title%}

{% block content %}
{% if tasks %}
<ul class="list-group">
```

```
    {% for task in tasks %}
    <li class="list-group-item">
        {{ task.title }}

        {% if task.assignee %}
        <span>(assignée à {{ task.assignee.first_name }})</span>
        {% else %}
        <span>(non assignée)</span>
        {% endif %}
    </li>
    {% endfor %}
</ul>
{% else %}
<div class="container m-4">
    <alert class="alert alert-warning">No Tasks</alert>
</div>
{% endif %}
{% endblock content %}
```

2. Ajoutons une nouvelle vue dans `task.views.py`

```
from django.shortcuts import render
from django.views import generic

from .models import Task

class IndexView(generic.ListView):
    model = Task
    template_name = "task/index.html"
    context_object_name = 'tasks'
```

3. Ajoutons une nouvelle route vers cette vue.
    - Dans l'application `task`, ajoutons le dossier `urls.py`

    ```
    from django.urls import path

    from .views import IndexView

    app_name = 'task'
    urlpatterns = [
        path('', IndexView.as_view(), name='index'),
    ]
    ```

    - Faisons un lien vers ce urlpatterns dans celui du projet (`mproject/urls.py`)

```
path('task/', include('task.urls')),
```

4. Ajoutons un lien vers cette vue dans le template de base (`BASE_DIR/templates/_base.html`)

```
<li id="nav-task" class="nav-item">
    <a class="nav-link" href="{% url 'task:index' %}">Tasks</a>
</li>
```

Activer le bon lien - solution

`tempaltes/developer/_base.html`

```
{% extends "_base.html" %}

{% block title %}GProject - Gestion des tâches{% endblock title %}

{% block menu-script %}
 $("#nav-home").removeClass('active')
 $("#nav-dev").addClass('active')
 $("#nav-task").removeClass('active')
{% endblock menu-script %}
```

`tempaltes/developer/index.html`

```
{% extends "_base.html" %}        👈old
{% extends "developer/_base.html" %}    👈new
```

Suppression d'une tâche - solution

On ajoute une vue qui permet de supprimer une tâches.

`task/views.py`

```
#...
def delete(request, task_id):
task = get_object_or_404(Task, pk=task_id)
task.delete()

return HttpResponseRedirect(reverse('task:index'))
```

On ajoute un chemin vers cette vue.

`task/views.py`

```
urlpatterns = [
 path('', IndexView.as_view(), name='index'),
 path('<int:task_id>/delete', views.delete, name='delete'), 👈 new
]
```

Création d'une tâche - solution

On ajoute un formulaire permettant la création d'une tâche

`task/forms.py`

```
from django import forms
from developer.models import Developer

class TaskForm(forms.Form):
 title = forms.CharField(label='Title', max_length=100)
 description = forms.CharField(label='Description', widget=forms.Textarea)
 assignee = forms.ModelChoiceField(queryset=Developer.objects.all(),
required=False)
```

Ou mieux, avec un `ModelForm` 😉

> `task/forms.py`

```
from django import forms
from .models import Task

class TaskForm(forms.ModelForm):
 class Meta:
     model = Task
     fields = ['title', 'description', 'assignee']
```

On ajoute le formulaire dans la vue d'index ; et tant qu'on est dans les vues, on ajoute une vue qui va nous permettre de valider et traiter le formulaire.

> `task/views.py`

```
from .forms import TaskForm 👈 new


class IndexView(generic.ListView):
 model = Task
 template_name = "task/index.html"
 context_object_name = 'tasks'

 def get_context_data(self, **kwargs):                              👈 new
     context = super(IndexView, self).get_context_data(**kwargs)    👈 new
     context['form'] = TaskForm                                     👈 new
     return context                                                 👈 new
#...
def create(request):
form = TaskForm(request.POST)

if form.is_valid():
    Task.objects.create(
        title=form.cleaned_data['title'],
        description=form.cleaned_data['description'],
        assignee=form.cleaned_data['assignee'])

return HttpResponseRedirect(reverse('task:index'))
```

On modifie le gabarit, ou plutôt, on créé un gabari que le va importer 😉

> `templates/task/_create_task_modal.html

```
{% load crispy_forms_tags %}

<button type="button" class="btn btn-primary" data-toggle="modal" > data-
target="#add-dev-modal">Ajouter une tâche</button>

<div class="modal fade " id="add-dev-modal">
 <div class="modal-dialog modal-dialog-centered">
     <div class="modal-content">
         <div class="modal-header">
             <h4 class="modal-title">Nouvelle tâche</h4>
             <button type="button" class="close" data-dismiss="modal"><i
class="fa fa-close"></i></button>
```

```
        </div>
        <div class="modal-body">
            <form action="{% url 'task:create' %}" method="post">
                {% csrf_token %}
                {{ form|crispy }}
                <div>
                    <button class="btn btn-primary"
type="submit">Créer</button>
                </div>
            </form>
        </div>
    </div>
 </div>
</div>
```

On inclu ce gabari dans le gabari d'index des tâches.

> `templates/task/index.html

```
<div class="container m-4">
 <alert class="alert alert-warning">No Tasks</alert>
</div>
{% endif %}
{% include 'task/_create_task_modal.html' %}
{% endblock content %}
```

On ajoute un chemin vers la vue de création de tâche.

> `task/urls.py`

```
app_name = 'task'
urlpatterns = [
 path('', IndexView.as_view(), name='index'),
 path('<int:task_id>/delete', views.delete, name='delete'),
 path('create', views.create, name="create"),
]
```

On fait une petite prière au dieu du copy/past 🙏

Et on lance le serveur pour tester.

Dans le détail d'un développeur - solution

On inclut le formulaire

> `developeur/detail.html`

```
#...
        </div>
    {% endif %}
    {% include 'task/_create_task_modal.html' %} 👈new
 </div>
{% endblock content %}
```

Dans la vue détail, on ajoute le formulaire dans le contexte.

> `developeur/views.py`

```
from task.forms import TaskForm
  👈new
#...
class DevDetailVue(DetailView):
model = Developer
template_name = 'developer/detail.html'

def get_context_data(self, **kwargs):
 👈new
    context = super(DetailView, self).get_context_data(**kwargs)
👈new
    form = TaskForm(
👈new
        initial={
 👈new
            'assignee': get_object_or_404(Developer, pk=self.kwargs['pk'])
👈new
            })
👈new
    form.fields['assignee'].disabled = True
 👈new
    context['form'] = form
👈new
    return context
👈new
```

On réactive le champs désactivé lors de l'envoie du formulaire. Sinon la donnée n'est pas envoyée.

`task/_create_task_modal.html`

```
<div class="modal-body">
 <form
     id="create-form"                                                    👈new
     onsubmit="$('#create-form *').removeAttr('disabled');"              👈new
     action="{% url 'task:create' %}" method="post">
     {% csrf_token %}
     {{ form|crispy }}
     <button class="btn btn-primary" type="submit">Créer</button>
 </form>
</div>
```

Annexes

Activer le bon lien app_name

Une autre solution pour activer le bon lien consiste à utiliser la variable `request.resolver_match.app_name`` afin de récupérer l'application courante.

- `app_name` représente l'espace nom de l'application pour laquel il y a eu correspondance avec l'URL.

`BASE_DIR/_base.html

```
<nav class="navbar navbar-expand-sm bg-primary navbar-dark border-top
border-white">
    <ul class="navbar-nav">
        <li id="nav-home" class="nav-item {% if
request.resolver_match.app_name == "" %}active{% endif %}">
            <a class="nav-link" href="{% url 'home' %}"><i class="fa fa-
home"></i></a>
        </li>
        <li id="nav-dev" class="nav-item {% if
request.resolver_match.app_name == "developer" %} active {% endif %}">
            <a class="nav-link" href="{% url 'developer:index'
%}">Developers</a>
        </li>
        <li id="nav-task" class="nav-item {% if
request.resolver_match.app_name == "task" %} active {% endif %}">
            <a class="nav-link" href="{% url 'task:index' %}">Tasks</a>
        </li>
    </ul>
</nav>
```

LiveServerTestCase et selenium

Voir ce lien