

Une webapp
100% Python
avec **NiceGUI**,
c'est possible ?

Gaël DURAND

Faire du web avec Python : Backend

django

Pyramid



Flask

web development,
one drop at a time

Bottle

SANIC
FRAMEWORK

Faire du web avec Python : Frontend

1ère approche : Interpréteur Python en JavaScript =

brython

<https://brython.info/index.html>

—

<div>0</div>			C
7	8	9	/
4	5	6	*
1	2	3	-
0	.	=	+

```
<!doctype html>
<html>

<head>
  <meta charset="utf-8">
  <script type="text/javascript" src="/src/brython.js">
  </script>
  <style>
    *{
      font-family: sans-serif;
      font-weight: normal;
      font-size: 1.1em;
    }
    td{
      background-color: #ccc;
      padding: 10px 30px 10px 30px;
      border-radius: 0.2em;
      text-align: center;
      cursor: default;
    }
    #result{
      border-color: #000;
      border-width: 1px;
      border-style: solid;
      padding: 10px 30px 10px 30px;
      text-align: right;
    }
  </style>
</head>
```

```

<body>

<script type="text/python">
from browser import document, html

# Construction de la calculatrice
calc = html.TABLE()
calc <= html.TR(html.TH(html.DIV("0", id="result"), colspan=3) +
                html.TD("C"))
lines = ["789/", "456*", "123-", "0.=+"]

calc <= (html.TR(html.TD(x) for x in line) for line in lines)

document <= calc

result = document["result"] # accès direct à un élément par son id

def action(event):
    """Gère l'événement "click" sur un bouton de la calculatrice."""
    # L'élément sur lequel l'utilisateur a cliqué est l'attribut "target" de
    # l'objet event
    element = event.target
    # Le texte affiché sur le bouton est l'attribut "text" de l'élément
    value = element.text
    if value not in "=C":
        # mise à jour du contenu de la zone "result"
        if result.text in ["0", "erreur"]:
            result.text = value
        else:
            result.text = result.text + value
    elif value == "C":
        # remise à zéro
        result.text = "0"
    elif value == "=":
        # exécution de la formule saisie
        try:
            x = eval(result.text)
            result.text = x
        except:
            result.text = "erreur"

# Associe la fonction action() à l'événement "click" sur tous les boutons
# de la page.
for button in document.select("td"):
    button.bind("click", action)
</script>

</body>

```

Faire du web avec Python : Frontend

2ème approche : Interpréteur Python en WebAssembly =



<https://pyodide.org/en/stable/index.html>

```
<!doctype html>
<html>
  <head>
    <script src="https://cdn.jsdelivr.net/pyodide/v0.24.1/full/pyodide.js"></script>
  </head>
  <body>
    Pyodide test page <br>
    Open your browser console to see Pyodide output
    <script type="text/javascript">
      async function main(){
        let pyodide = await loadPyodide();
        console.log(pyodide.runPython(`
          import sys
          sys.version
        `));
        pyodide.runPython("print(1 + 2)");
      }
      main();
    </script>
  </body>
</html>
```




Éléments

Enregistreur

Console

Sources

Informations sur les performances



top ▼



Filtrer

Niveaux par défaut ▼

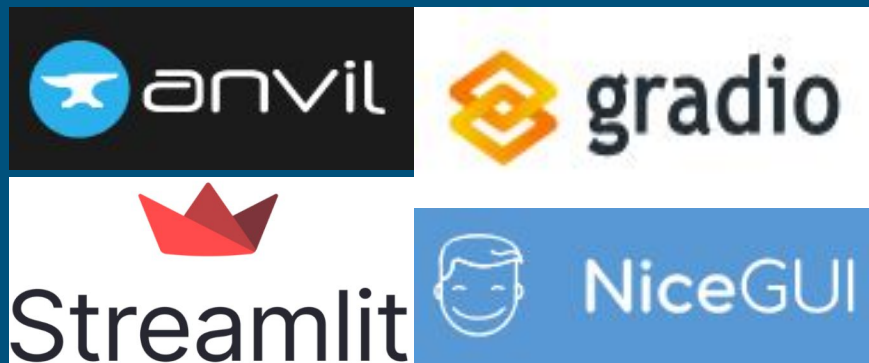
3.11.3 (main, Sep 25 2023, 20:45:01) [Clang 18.0.0 (<https://github.com/llvm/llvm-project>
d1e685df45dc5944b43d2547d013

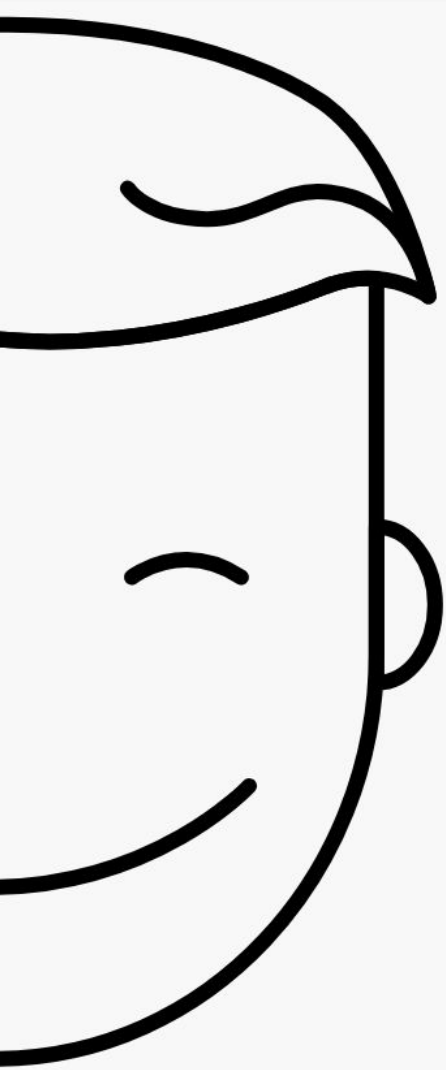
3



Faire du web avec Python : Full-Stack

Générer la partie frontend avec du code Python





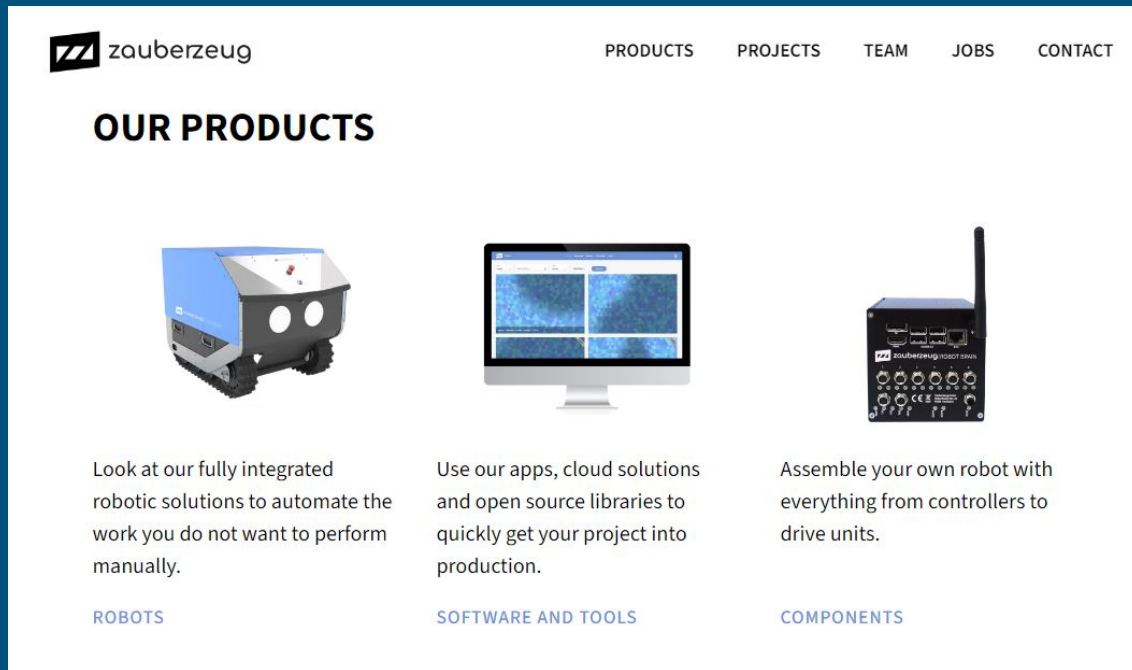
Meet the NiceGUI.

And let any browser be the
frontend of your Python code.






Types d'applications ciblées

- WebApp
- Robotics
- IoT
- Smart Home
- Machine Learning



The screenshot shows the 'OUR PRODUCTS' section of the zauberzeug website. It features three columns, each with an image, a description, and a category label.

PRODUCTS	PROJECTS	TEAM	JOBS	CONTACT
 <p>Look at our fully integrated robotic solutions to automate the work you do not want to perform manually.</p> <p>ROBOTS</p>	 <p>Use our apps, cloud solutions and open source libraries to quickly get your project into production.</p> <p>SOFTWARE AND TOOLS</p>	 <p>Assemble your own robot with everything from controllers to drive units.</p> <p>COMPONENTS</p>		

<https://zauberzeug.com/>

Installation en 1 pip

1.

Create **main.py**

```
from nicegui import ui
ui.label('Hello NiceGUI!')
ui.run()
```

2.

Install and launch

```
pip3 install nicegui
python3 main.py
```

3.

Enjoy!

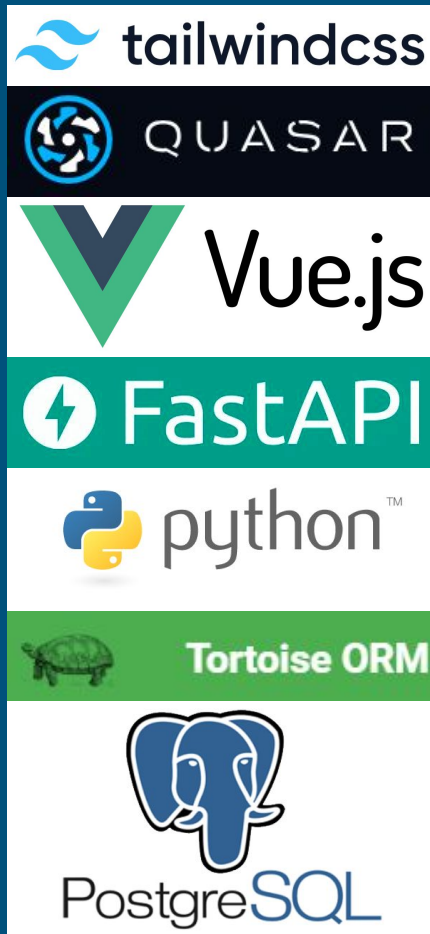
```
NiceGUI
Hello NiceGUI!
```

Stack

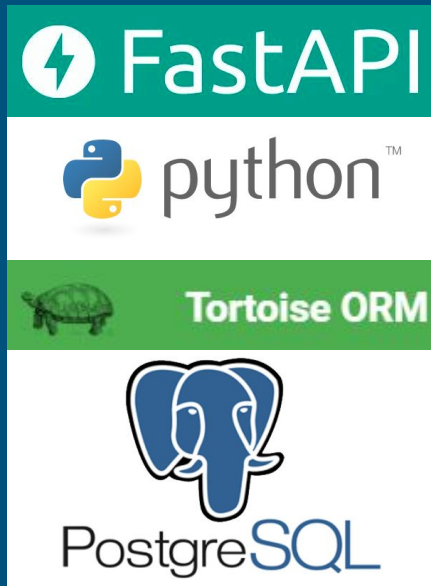


FRONTEND

BACKEND



Process



Pourquoi FastAPI ?



Meilleures performances que des frameworks Node.js comme Express ou NestJS
<https://hostadvice.com/blog/web-hosting/node-js/fastapi-vs-nodejs/>

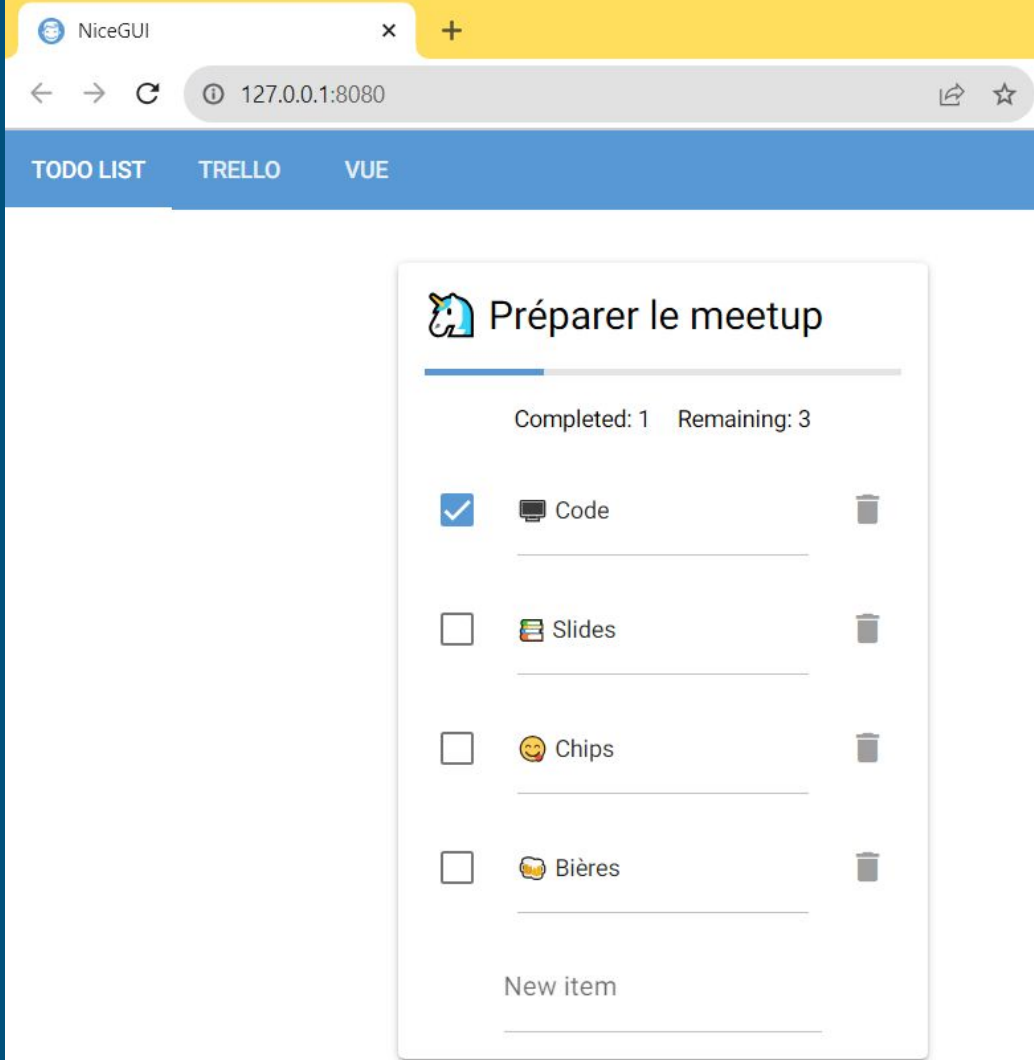
Démos

Application principale

- Installer NiceGUI : `pip install nicegui`
- Récupérer le projet sur Github : <https://github.com/zauberzeug/nicegui/>

App principale

- 3 Onglets
 - TODO List
 - Trello
 - Composant Vue
- Serveur local



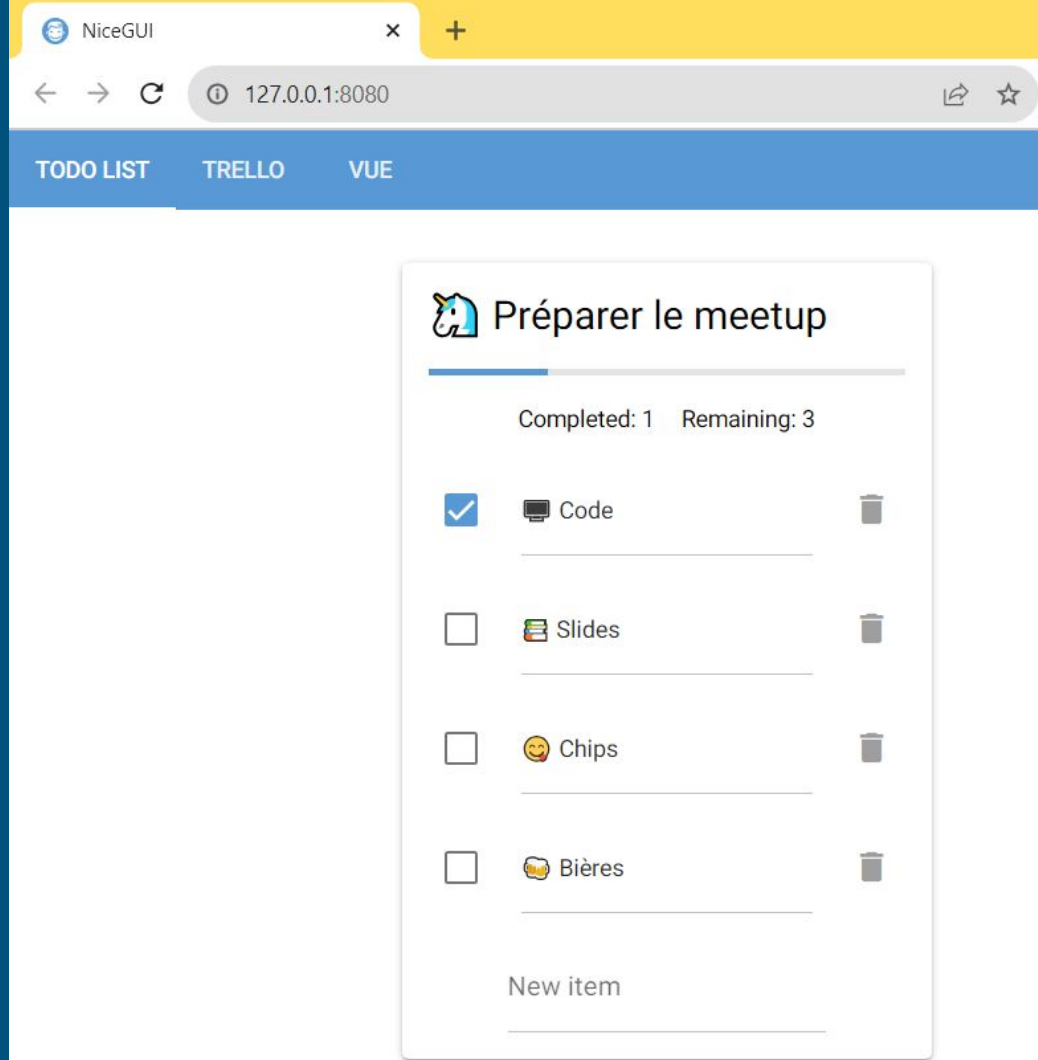
Onglets dans le menu supérieur

Création des onglets avec `ui.header` et `ui.tabs` : composants de Quasar (Vue)

Mise en forme dans classes avec Tailwind

```
with ui.header().classes(replace='row items-center') as header:  
    with ui.tabs() as tabs:  
        ui.tab('TODO List')  
        ui.tab('Trello')  
        ui.tab('Vue')
```

TODO List



TODO List

Relier les valeurs pour actualisation automatique avec `bind_text_from`

Ajouter des événements avec `on` et le nom de l'événement

Ajouter des callbacks avec des arguments via les lambdas

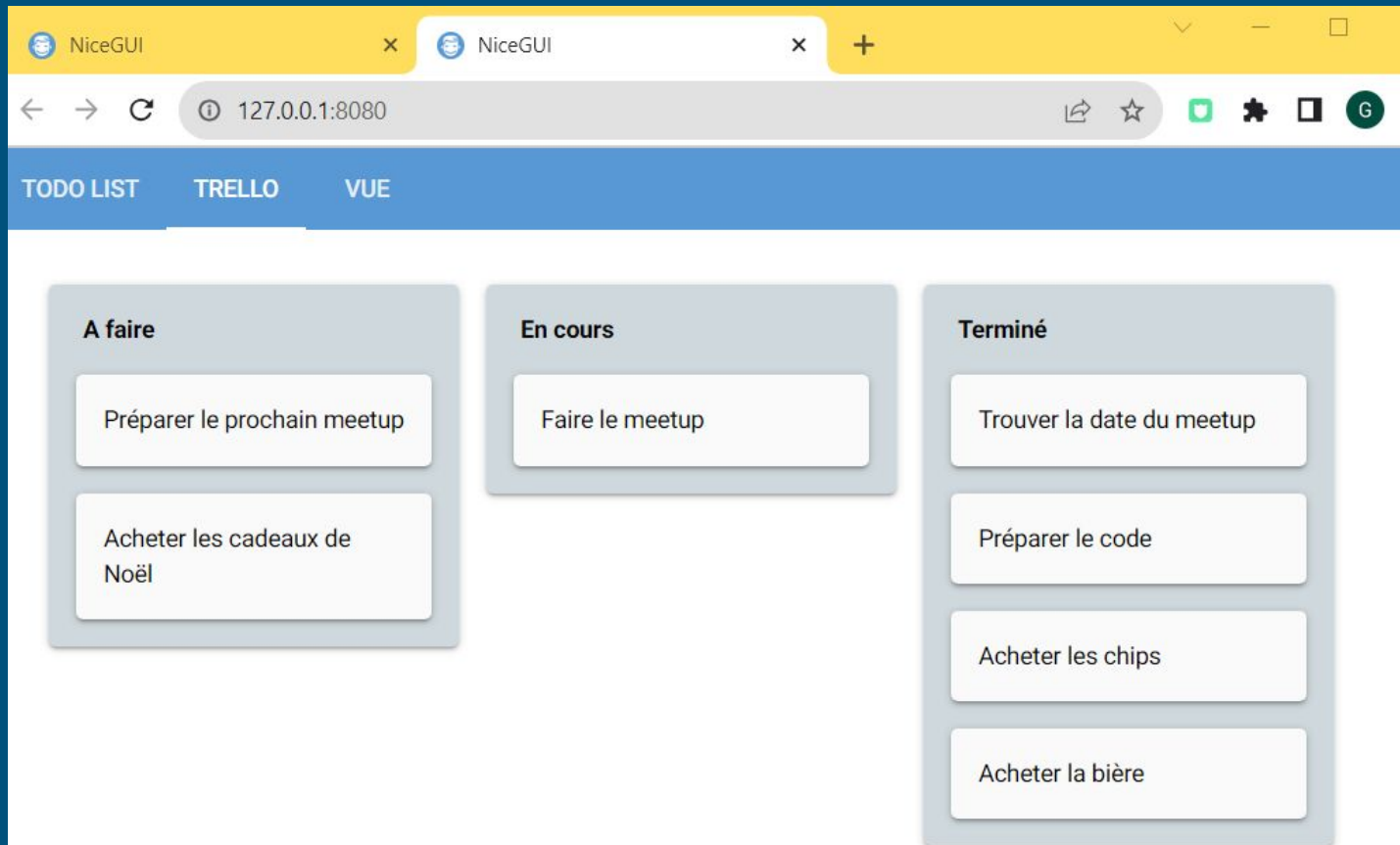
```
with ui.tab_panels(tabs, value='TODO List').classes('w-full'):
    with ui.tab_panel('TODO List'):
        with ui.column().classes('w-full items-center'):
            with ui.card().classes('w-80 items-stretch'):
                ui.label().bind_text_from(todo.todos, 'title').classes('text-semibold text-2xl')
                todo.todo_ui()
                add_input = ui.input('New item').classes('mx-12')
                add_input.on('keydown.enter', lambda: (todo.todos.add(add_input.value), add_input.set_value('')))
```

TODO List

Rafraîchir les composants avec le décorateur `@ui.refreshable`

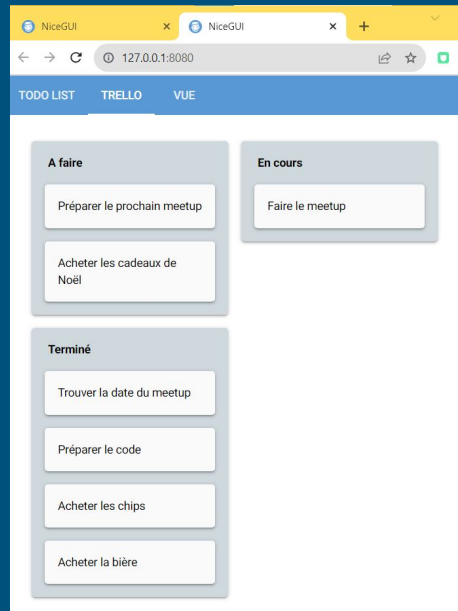
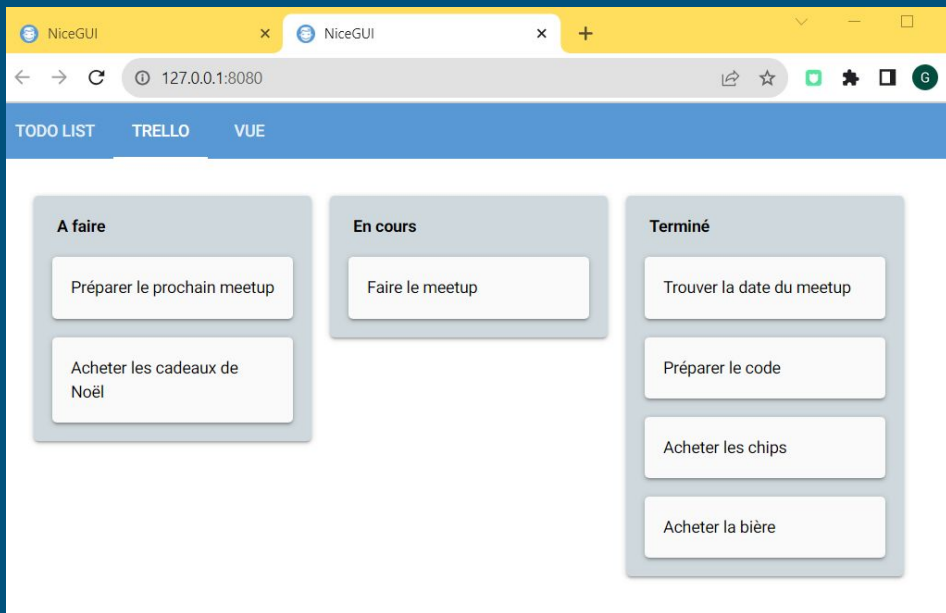
```
@ui.refreshable
def todo_ui():
    if not todos.items:
        ui.label('List is empty.').classes('mx-auto')
        return
    ui.linear_progress(sum(item.done for item in todos.items) / len(todos.items), show_value=False)
    with ui.row().classes('justify-center w-full'):
        ui.label(f'Completed: {sum(item.done for item in todos.items)}')
        ui.label(f'Remaining: {sum(not item.done for item in todos.items)}')
    for item in todos.items:
        with ui.row().classes('items-center'):
            ui.checkbox(value=item.done, on_change=todo_ui.refresh).bind_value(item, 'done')
            ui.input(value=item.name).classes('flex-grow').bind_value(item, 'name')
            ui.button(on_click=lambda item=item: todos.remove(item), icon='delete').props('flat fab-mini color=grey')
```

Trello



Trello

Responsive design avec Quasar et Tailwind



Trello

Extension des composants de NiceGUI par héritage

```
class column(ui.column):

    def __init__(self, name: str, on_drop: Optional[Callable[[Item, str], None]] = None) -> None:
        super().__init__()
        with self.classes('bg-blue-grey-2 w-60 p-4 rounded shadow-2'):
            ui.label(name).classes('text-bold ml-1')
        self.name = name
        self.on('dragover.prevent', self.highlight)
        self.on('dragleave', self.unhighlight)
        self.on('drop', self.move_card)
        self.on_drop = on_drop

    def highlight(self) -> None:
        self.classes(remove='bg-blue-grey-2', add='bg-blue-grey-3')

    def unhighlight(self) -> None:
        self.classes(remove='bg-blue-grey-3', add='bg-blue-grey-2')

    def move_card(self) -> None:
        global dragged
        self.unhighlight()
        dragged.parent_slot.parent.remove(dragged)
        with self:
            card(dragged.item)
        self.on_drop(dragged.item, self.name)
        dragged = None
```

Vue

Ajout de nouveaux
composants en Vue

Le nouveau compteur ultime est arrivé !

Clique sur le bouton et admire 🤩

Clics: 0

RÉINITIALISER

Vue

Côté Python

```
class Counter(Element, component='counter.js'):

    def __init__(self, title: str, *, on_change: Optional[Callable] = None) -> None:
        super().__init__()
        self._props['title'] = title
        self.on('change', on_change)

    def reset(self) -> None:
        self.run_method('reset')
```

```
with ui.column().classes('w-full items-center'):
    ui.markdown('''
    ### Le nouveau compteur ultime est arrivé !

    Cliquez sur le bouton et admire 🥰
    ''').classes('items-center')
    with ui.card().classes('items-center'):
        counter = Counter('Clics', on_change=lambda e: ui.notify(f'Wow ! La valeur a changé : {e.args} !'))

    ui.button('Réinitialiser', on_click=counter.reset).props('small outline')
```



Vue

Côté JavaScript

```
export default {  
  template: `  
    <button @click="handle_click">  
      <strong>{{title}}: {{value}}</strong>  
    </button>`,  
  data() {  
    return {  
      value: 0,  
    };  
  },  
  methods: {  
    handle_click() {  
      this.value += 1;  
      this.$emit("change", this.value);  
    },  
    reset() {  
      this.value = 0;  
    },  
  },  
  props: {  
    title: String,  
  },  
};
```

ORM

Tortoise ORM
compatible avec
FastAPI

Name	Age	
		+
Name	Age	
Dumbledore	95	
Name	Age	
Batman	37	

ORM

```
class User(models.Model):
    id = fields.IntField(pk=True)
    name = fields.CharField(max_length=255)
    age = fields.IntField()
```

```
register_tortoise(
    app,
    db_url='sqlite:///db.sqlite3',
    modules={'models': ['orm_tab.orm_models']}, # tortoise will look for models in this main module
    generate_schemas=True, # in production you should use version control migrations instead
)

@ui.refreshable
async def list_of_users() -> None:
    async def delete(user: orm_models.User) -> None:
        await user.delete()
        list_of_users.refresh()

    users: List[orm_models.User] = await orm_models.User.all()
    with ui.column().classes('w-full items-center'):
        for user in reversed(users):
            with ui.card():
                with ui.row().classes('items-center'):
                    ui.input('Name', on_change=user.save) \
                        .bind_value(user, 'name').on('blur', list_of_users.refresh)
                    ui.number('Age', on_change=user.save, format='%.0f') \
                        .bind_value(user, 'age').on('blur', list_of_users.refresh).classes('w-20')
                    ui.button(icon='delete', on_click=lambda u=user: delete(u)).props('flat')
```

Autres démos : à vous de choisir !

- chatgpt, slideshow, login, tableau éditable, map, pandas...

Merci !

