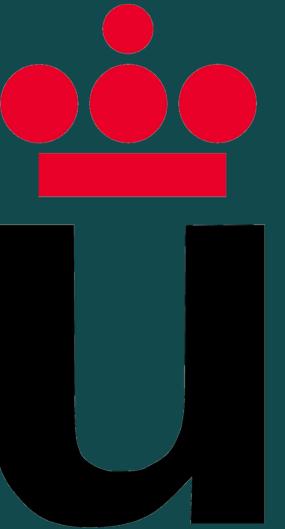


BT Studio

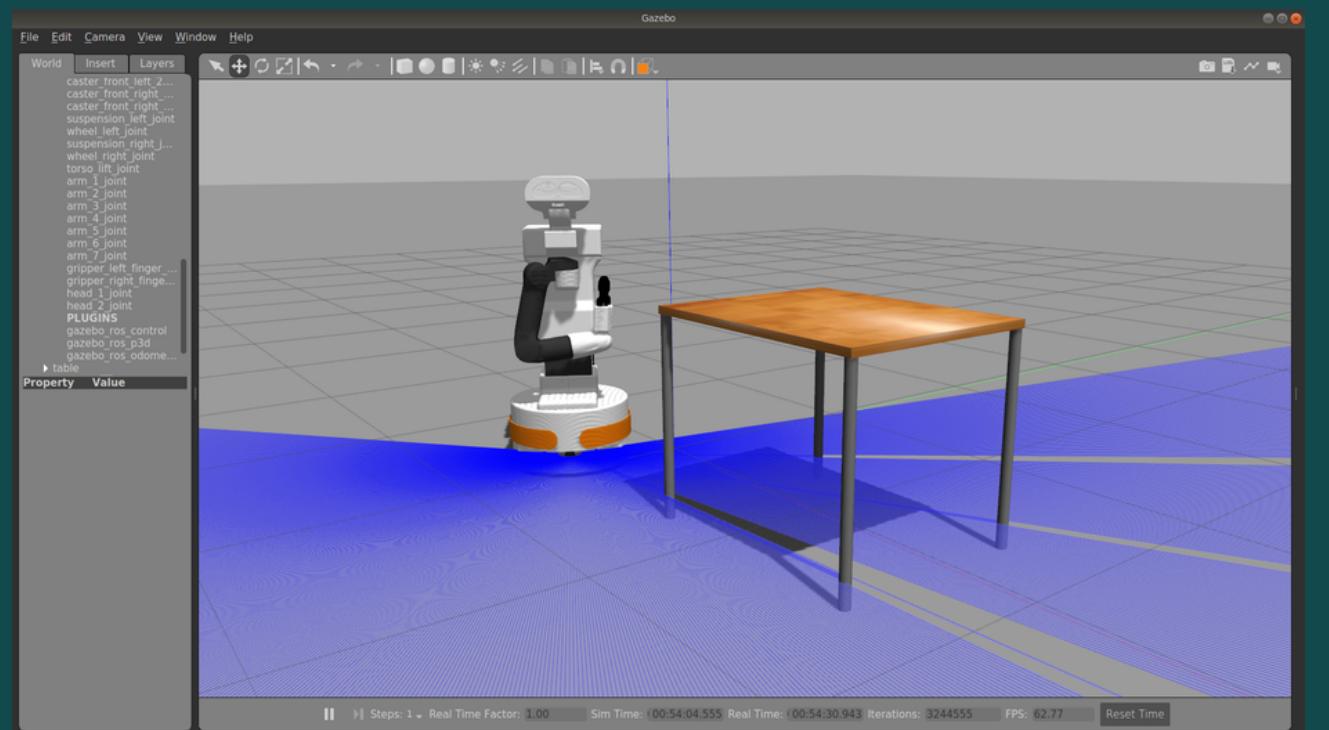
Un IDE web para la programación de
aplicaciones robóticas con Behavior Trees



Autor: Óscar Martínez Martínez
Tutor: José María Cañas Plaza

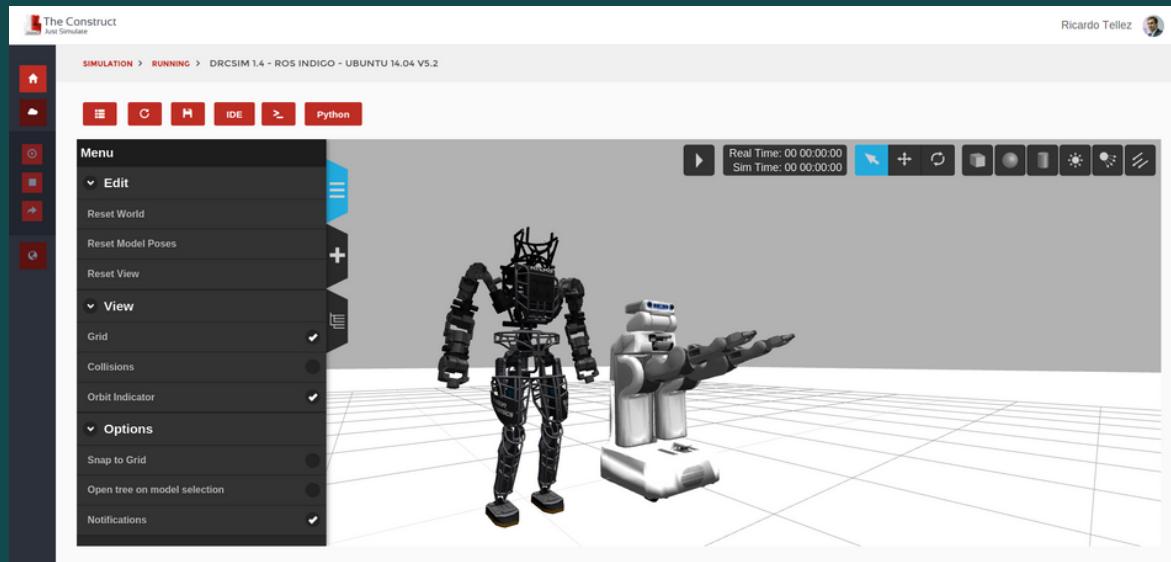


Introducción

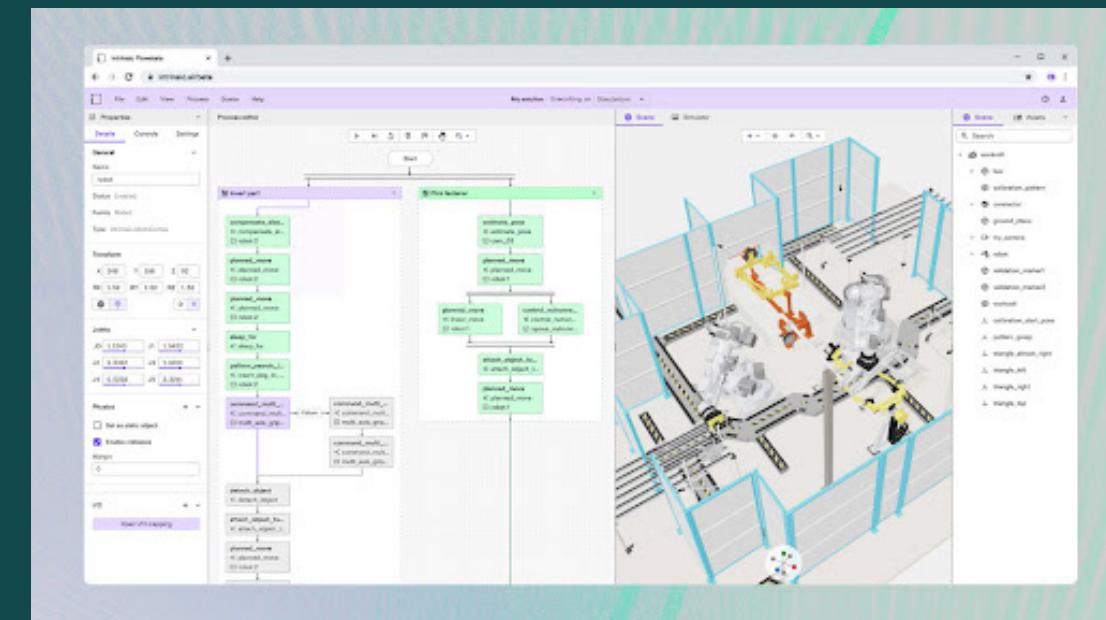


Introducción

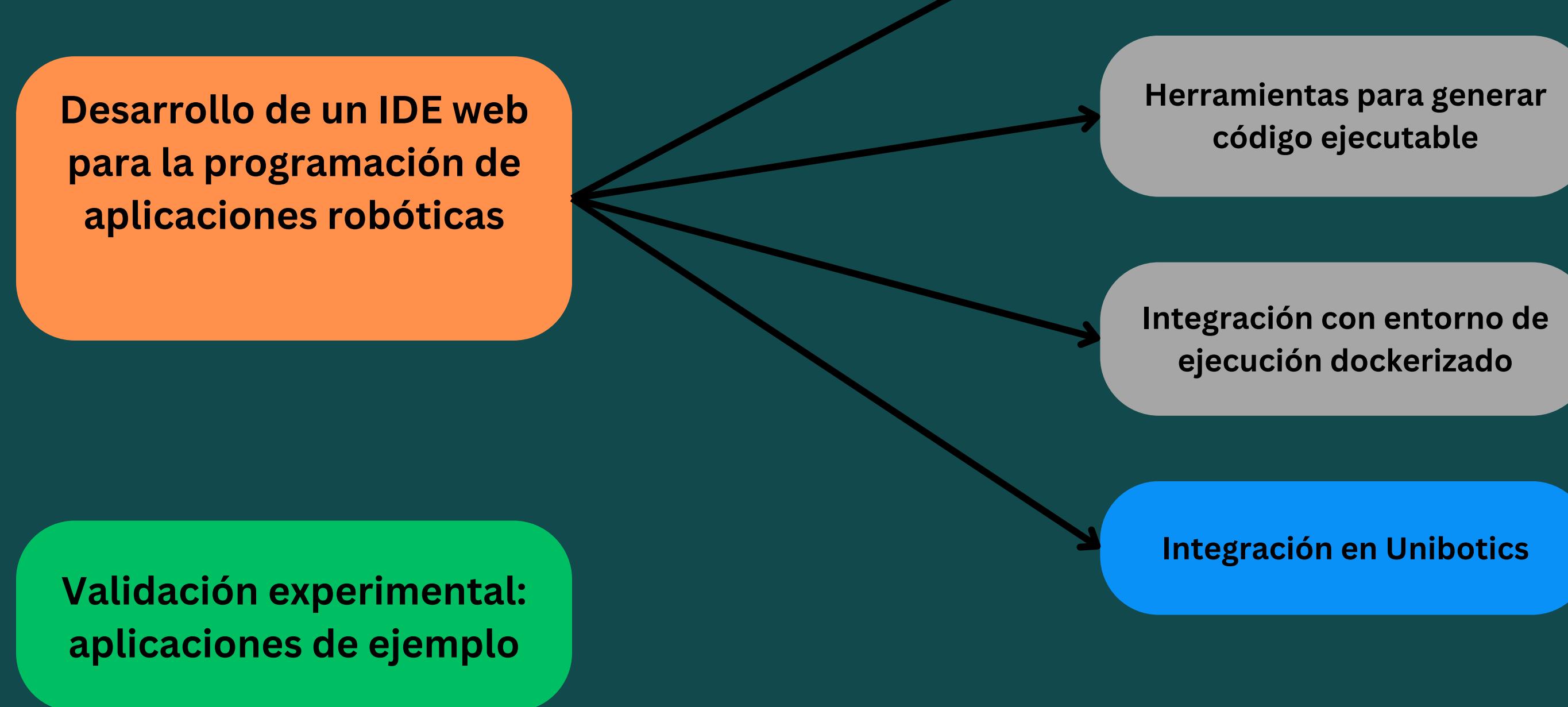
Plataformas educativas



Plataformas profesionales



Objetivos

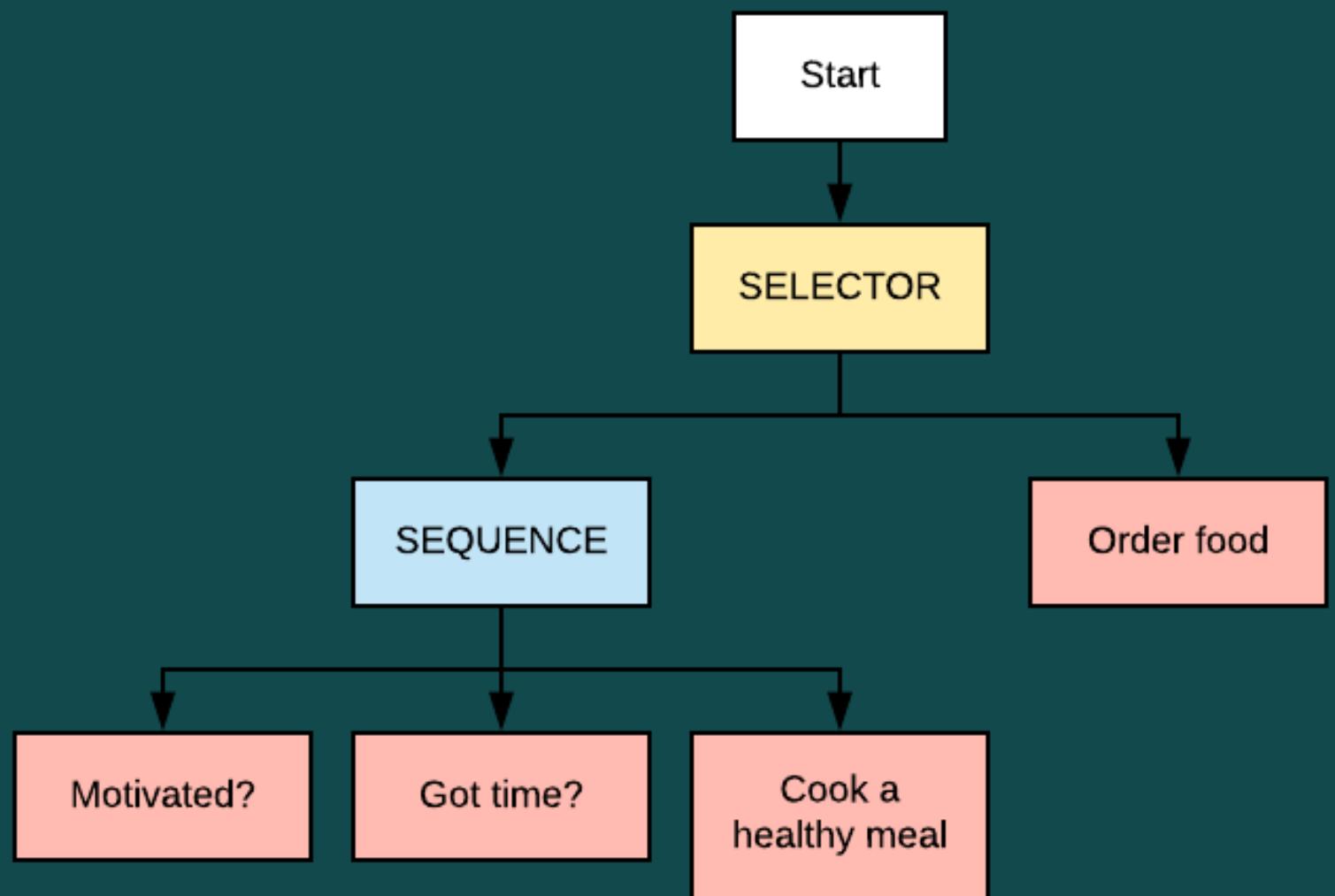


Fundamentos técnicos

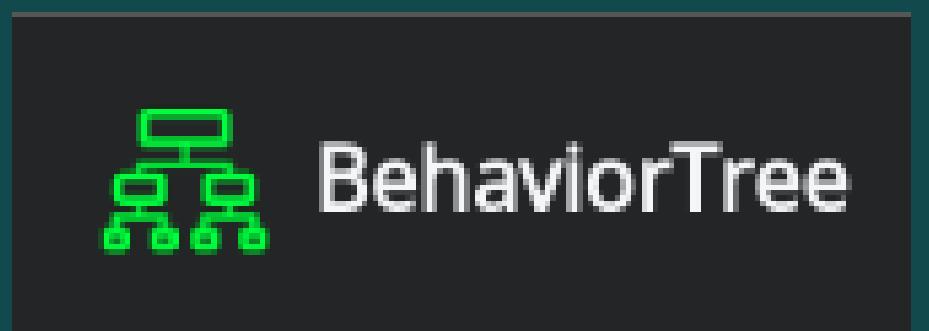
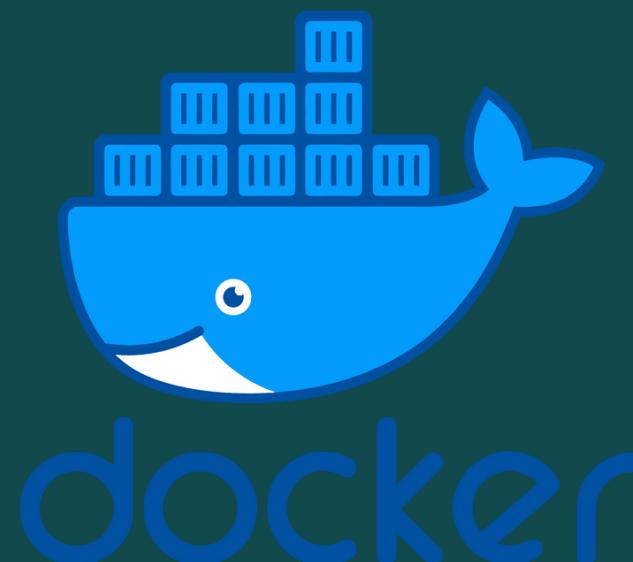


django

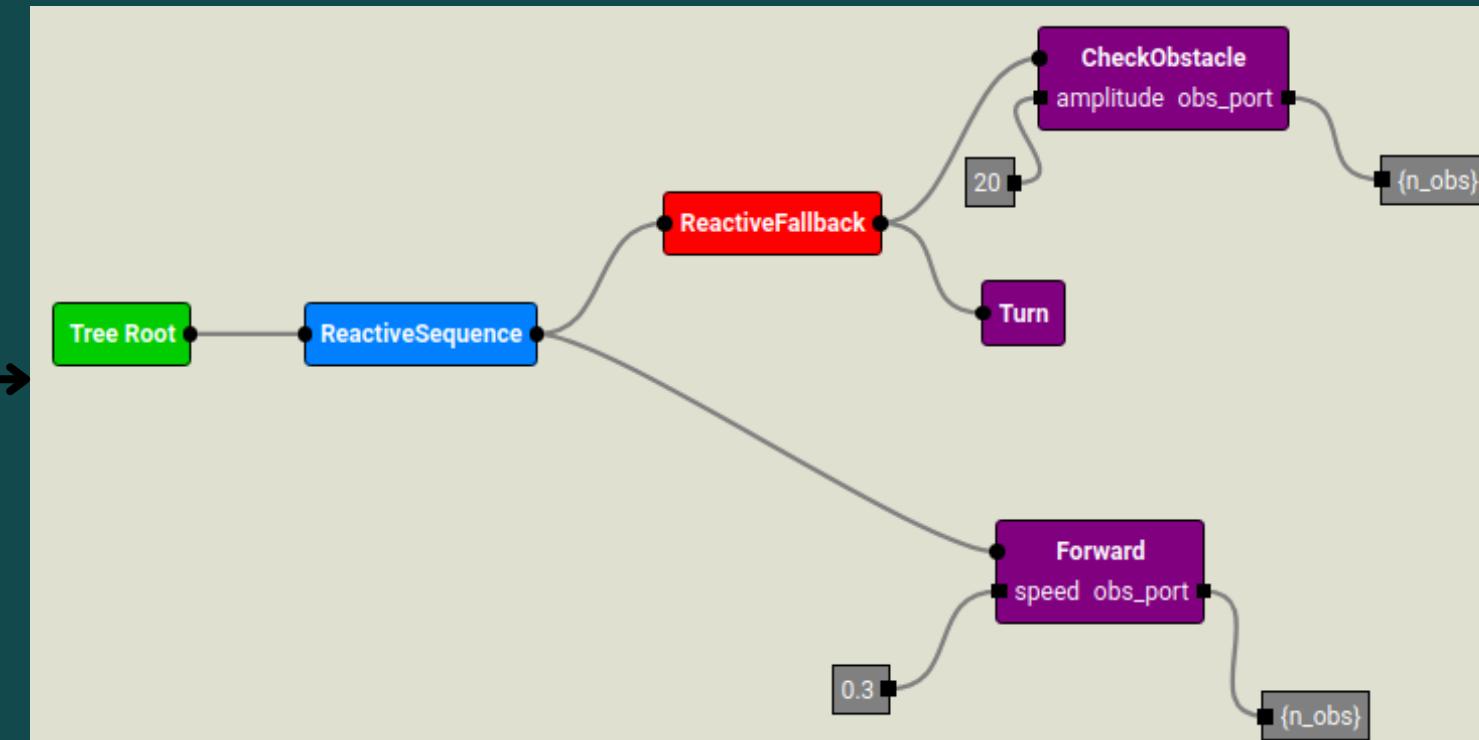
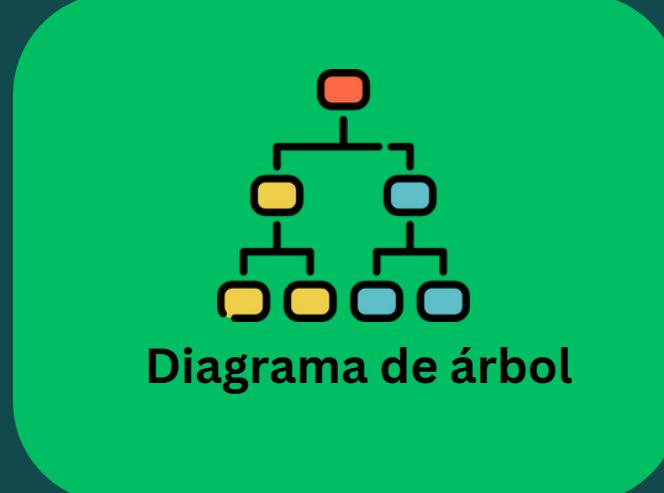
Fundamentos técnicos



2



BT Studio: definición de aplicaciones



Ficheros de las
acciones en Python

Forward.py

```
1 import py_trees
2 import geometry_msgs
3
4 class Turn(py_trees.behaviour.Behaviour):
5
6     def __init__(self, name, ports = None):
7
8         super().__init__(name)
9         self.ports = ports if ports is not None else []
10
11     def setup(self, **kwargs: int) -> None:
12
13         pass
14
15     def initialise(self) -> None:
16
17         pass
18
19     def update(self) -> py_trees.common.Status:
20
21         pass
22
23     def terminate(self, new_status: py_trees.common.Status) -> None:
24
25         pass
```

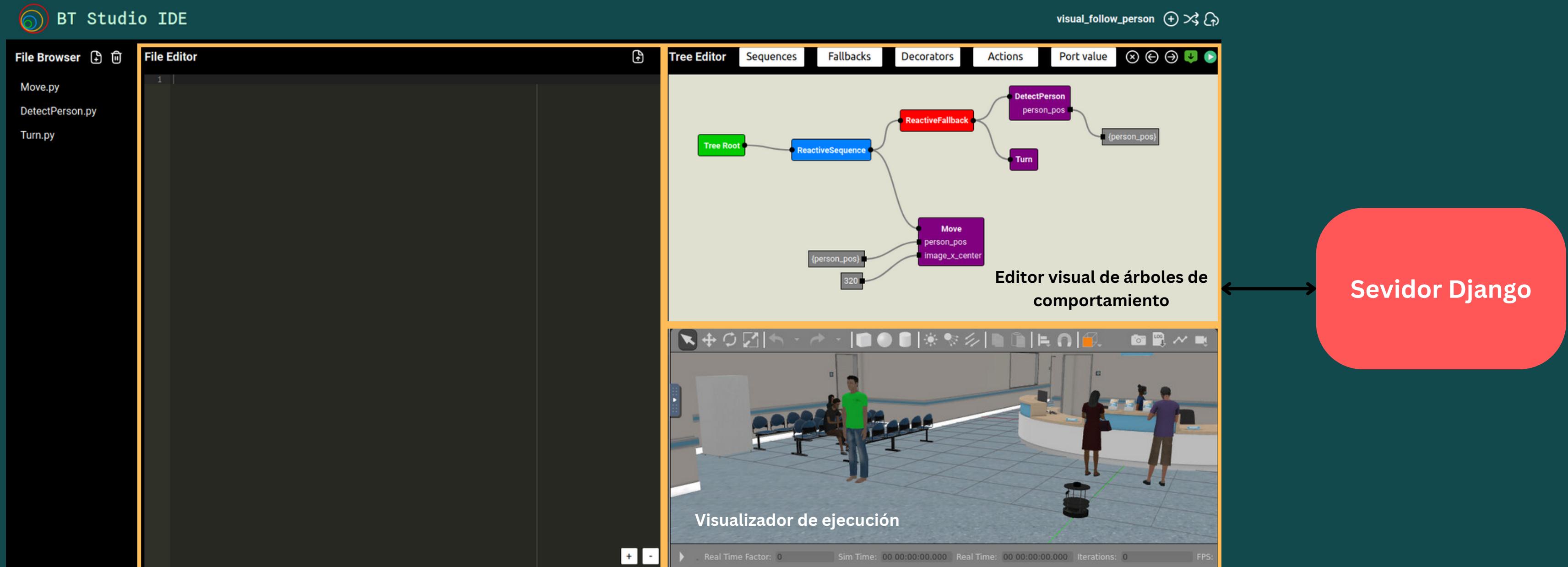
Turn.py

```
1 import py_trees
2 import geometry_msgs
3
4 class CheckObstacle(py_trees.behaviour.Behaviour):
5
6     def __init__(self, name, ports = None):
7
8         super().__init__(name)
9         self.ports = ports if ports is not None else []
10
11     def setup(self, **kwargs: int) -> None:
12
13         pass
14
15     def initialise(self) -> None:
16
17         pass
18
19     def update(self) -> py_trees.common.Status:
20
21         pass
22
23     def terminate(self, new_status: py_trees.common.Status) -> None:
24
25         pass
```

CheckObstacle.py

```
1 import py_trees
2 import geometry_msgs
3
4 class Forward(py_trees.behaviour.Behaviour):
5
6     def __init__(self, name, ports = None):
7
8         super().__init__(name)
9         self.ports = ports if ports is not None else []
10
11     def setup(self, **kwargs: int) -> None:
12
13         pass
14
15     def initialise(self) -> None:
16
17         pass
18
19     def update(self) -> py_trees.common.Status:
20
21         pass
22
23     def terminate(self, new_status: py_trees.common.Status) -> None:
24
25         pass
```

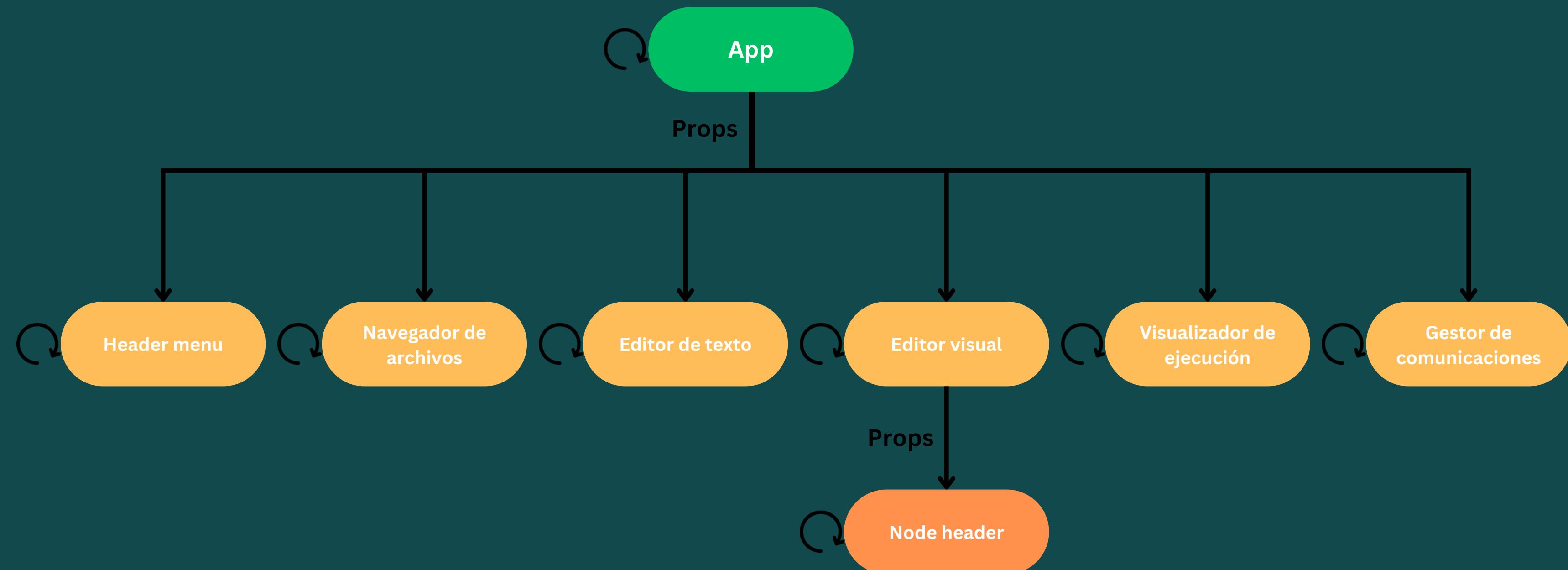
BT Studio: interfaz del IDE



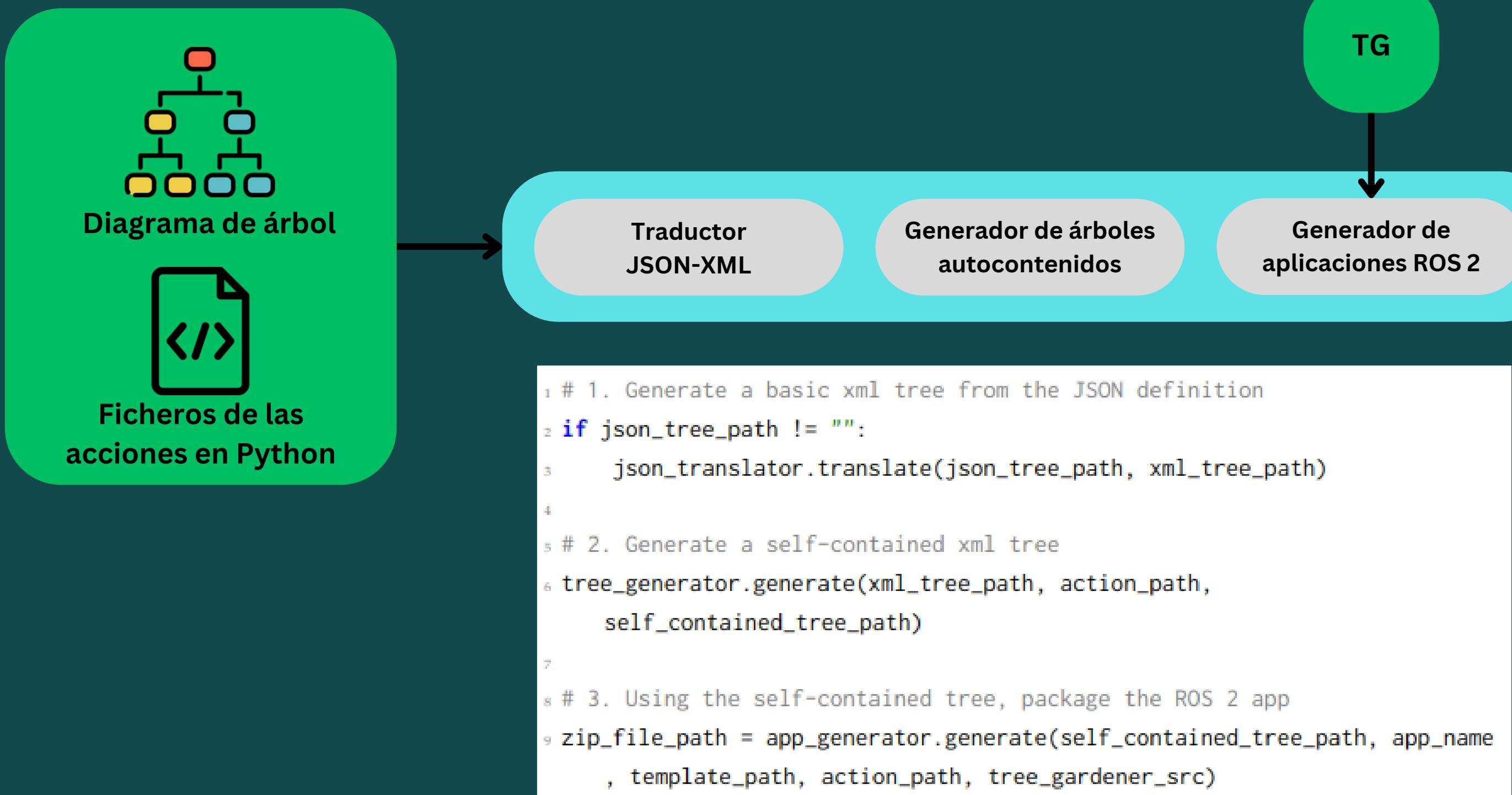
Frontend

Backend

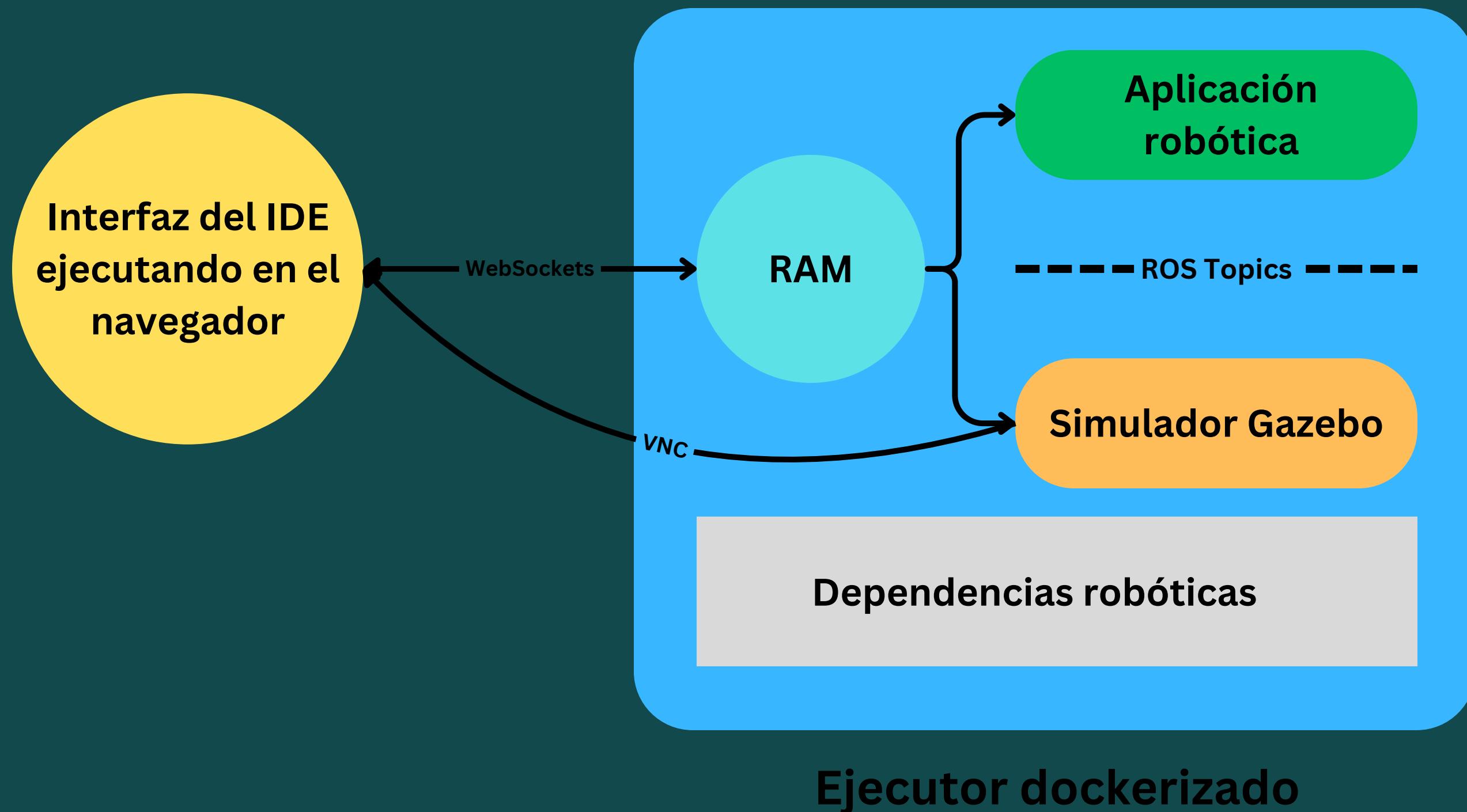
BT Studio: interfaz del IDE



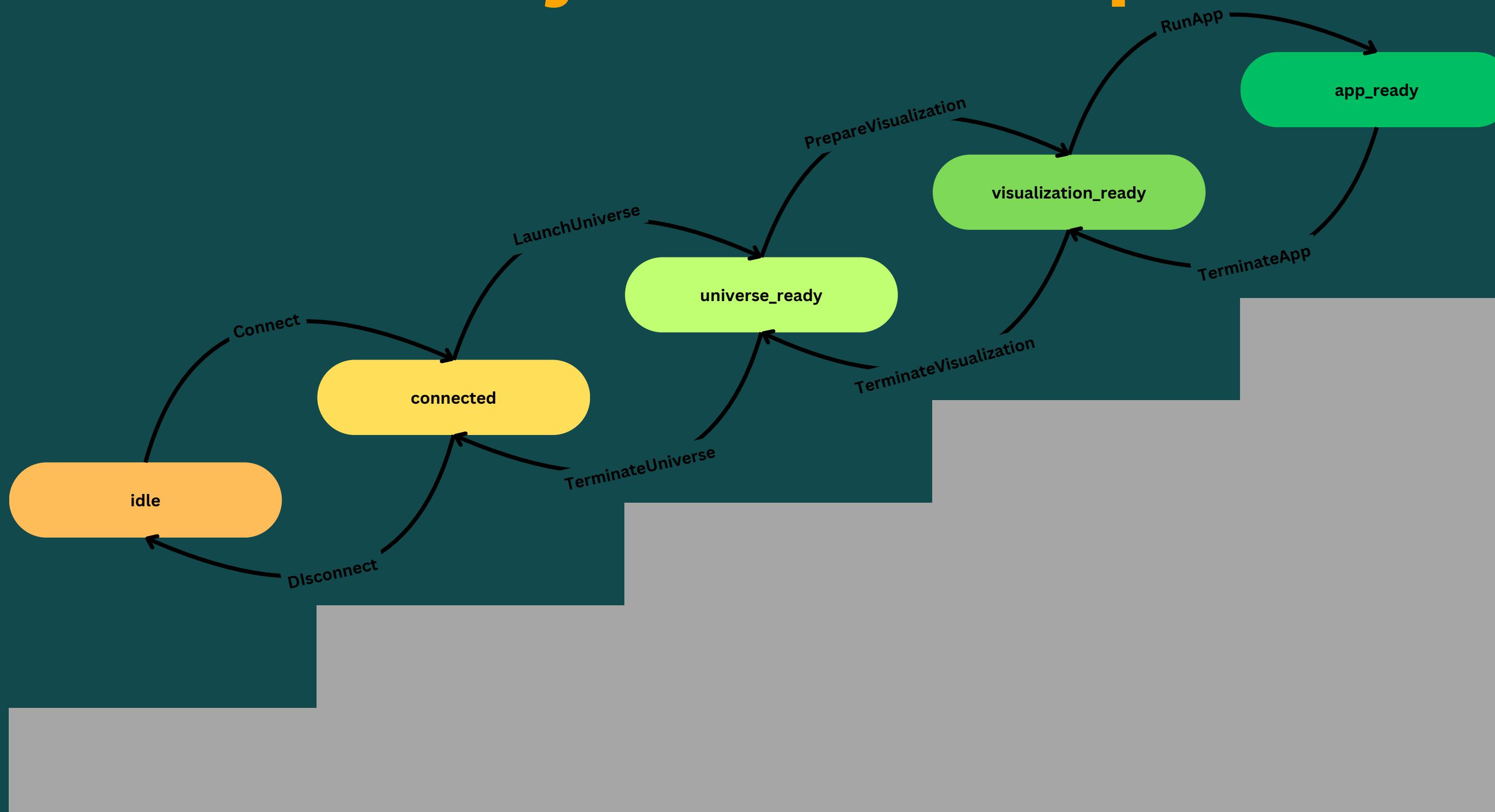
BT Studio: traductor de aplicaciones



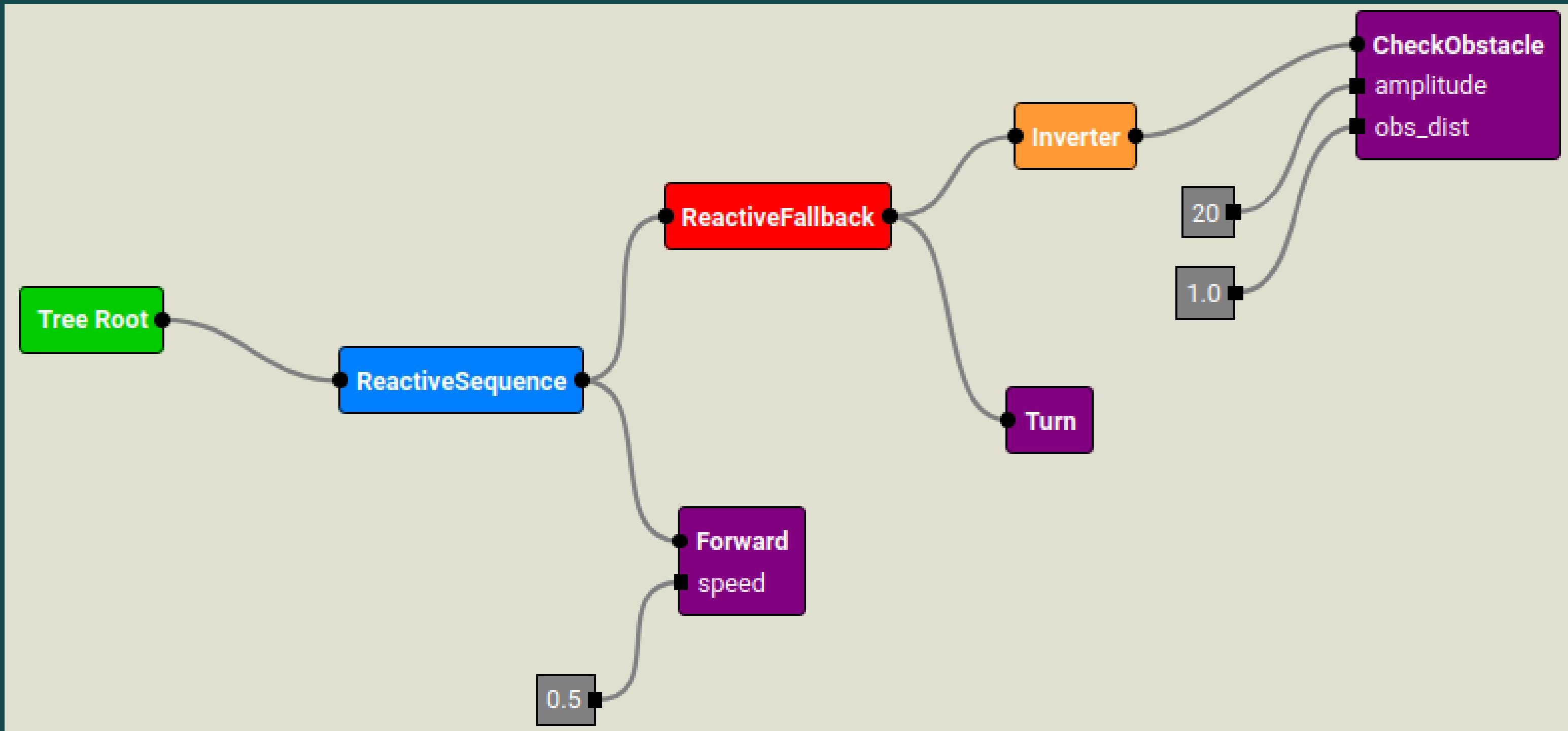
BT Studio: ejecutor de aplicaciones



BT Studio: ejecutor de aplicaciones



Validación experimental: Laser Bump and Go



Validación experimental: Laser Bump and Go



BT Studio 0.3 Laser Bump And Go

JdeRobot presents

BT Studio 0.3

A ROS Behavior-Tree web IDE

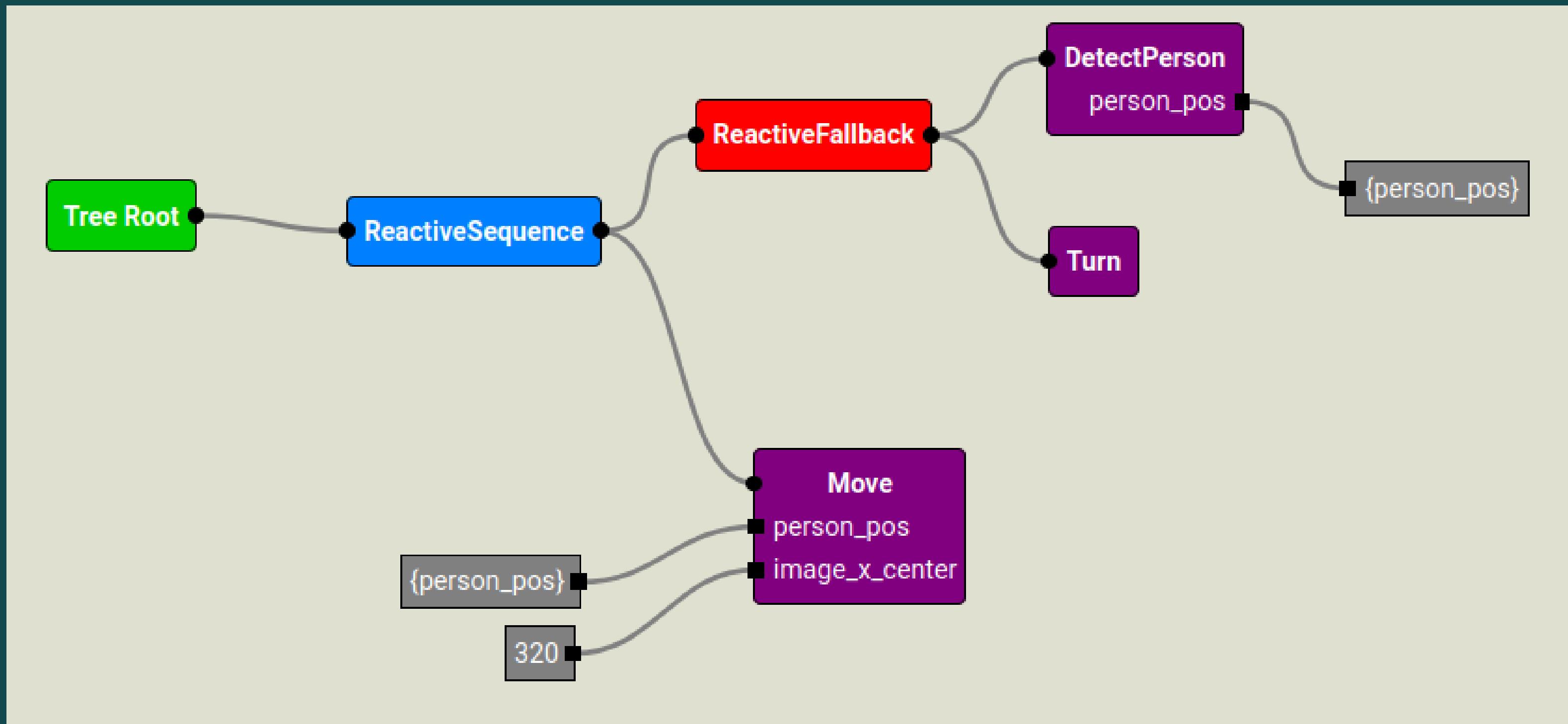
Share

Application demo: Laser Bump and Go

Watch on  YouTube

The image shows a YouTube video thumbnail for a demonstration of the BT Studio 0.3 software. The thumbnail features a dark background with a red, yellow, and blue concentric circle logo in the center. A white play button icon is overlaid on the left side. Text on the left includes the title 'BT Studio 0.3 Laser Bump And Go', the developer's name 'JdeRobot presents', the product name 'BT Studio 0.3', and a description 'A ROS Behavior-Tree web IDE'. On the right, there is a 'Share' button with a white arrow icon. At the bottom, there is a call-to-action 'Application demo: Laser Bump and Go' followed by a 'Watch on YouTube' link with the YouTube logo.

Validación experimental: Visual Follow Person



Validación experimental: Visual Follow Person



The image is a YouTube thumbnail for a video titled "BT Studio 0.3 Visual Follow Person". The thumbnail features a dark teal background. In the top left corner, there is a circular logo composed of three concentric circles in red, orange, and green. To the right of the logo, the title "BT Studio 0.3 Visual Follow Person" is written in white. Below the title, the text "JdeRobot presents" is followed by a large, bold "BT Studio 0.3". Underneath that, it says "A ROS Behavior-Tree web IDE". To the right of the text, there is a red YouTube play button icon. In the top right corner of the thumbnail, there is a "Share" button with a white arrow pointing upwards.

BT Studio 0.3 Visual Follow Person

JdeRobot presents

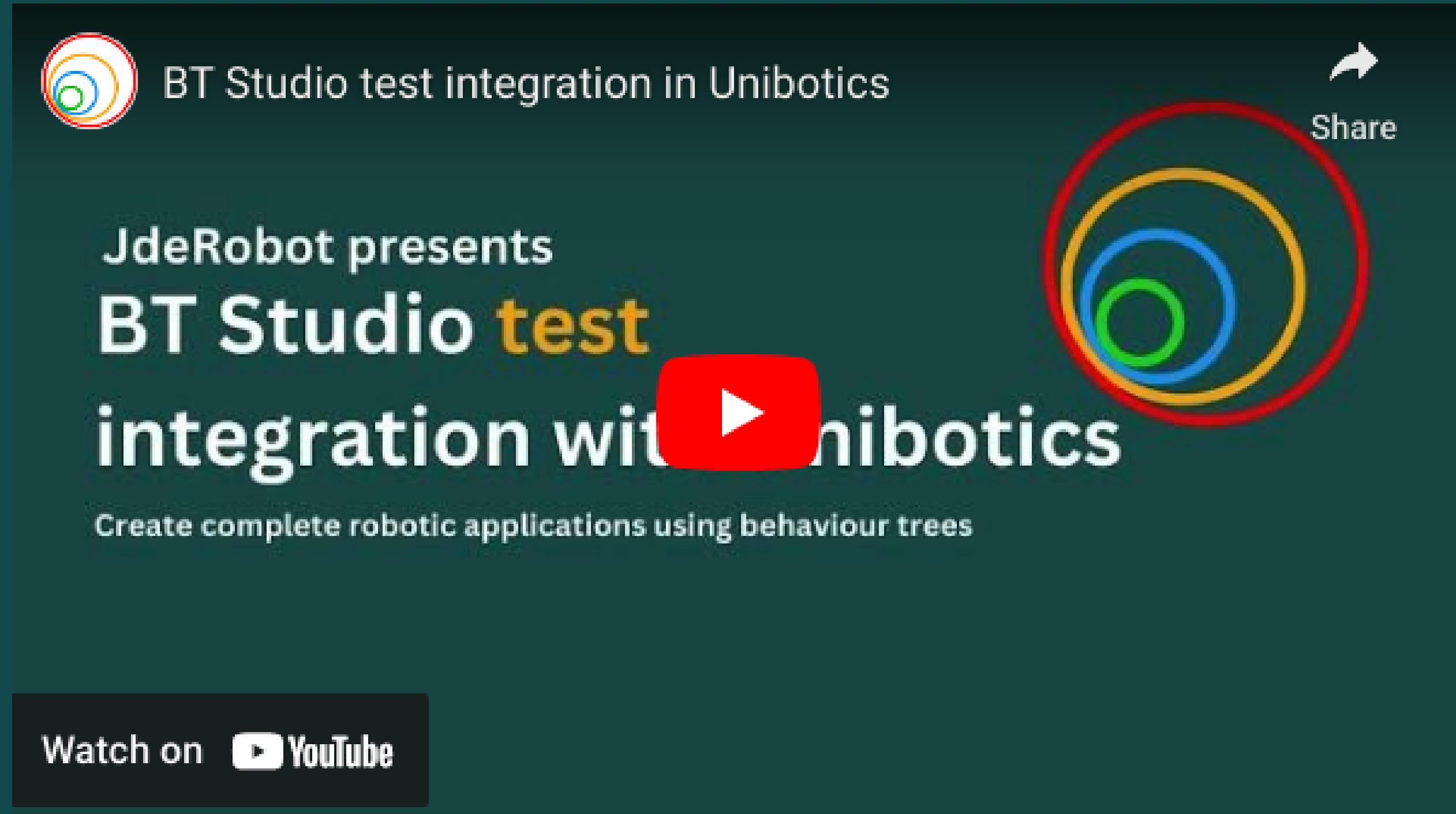
BT Studio 0.3

A ROS Behavior-Tree web IDE

Application demo: Visual Follow Person

Watch on  YouTube

Validación experimental: integración Unibotics



Conclusiones

```
#####
# Tree classes
#####

class ReactiveSequence(py_trees.composites.Composite):...
class SequenceWithMemory(py_trees.composites.Composite):...
class ReactiveFallback(py_trees.composites.Composite):...
class Delay(py_trees.decorators.Decorator):...

#####
# Auxiliary variables
#####

factory = { ... }

#####
# Auxiliary functions
#####

def get_branches(element):...
def construct_behaviour_tree_from_xml(doc):...
def add_actions_to_factory(doc):...

#####
# Tree factory
#####

class TreeFactory(rclpy.node.Node):

    def __init__(self):...

    def create_tree_from_file(self, tree_path, timeout=1000):
        # Open the self contained xml file
        self_file = open(tree_path, 'r')
        self_contained_tree = self_file.read()
        xml_doc = ET.fromstring(self_contained_tree)

        # Add actions to factory
        add_actions_to_factory(xml_doc)

        # Init tree
        root = construct_behaviour_tree_from_xml(xml_doc)
        self.tree = py_trees_ros.trees.BehaviourTree(root=root, unicode_tree_debug=False)

        # Setup tree
        try:
            self.tree.setup(timeout=timeout)
        except py_trees_ros.exceptions.TimedOutError as e:
            console.logerror(console.red + "Failed to setup tree" + console.reset)
            self.tree.shutdown()
            return None
        except KeyboardInterrupt:
            # not a warning, nor error, usually a user-initiated shutdown
            console.logerror("Tree setup interrupted")
            self.tree.shutdown()
            return None

    return self.tree
```

```
class GlobalBlackboard:

    _instance = None

    @staticmethod
    def get_instance():

        if GlobalBlackboard._instance is None:
            GlobalBlackboard._instance = py_trees.blackboard.Client(name="Global")

        return GlobalBlackboard._instance

    def get_port_content(port_value):

        if port_value.startswith("{}") and port_value.endswith("{}"):
            bb_key = port_value.strip("{}")
            blackboard = GlobalBlackboard.get_instance()

            # Return the value of the blackboard entry if it exists, otherwise None
            return getattr(blackboard, bb_key, "")

        else:
            return port_value

    def set_port_content(port_value, value):

        if not (port_value.startswith("{}") and port_value.endswith("{}")):
            raise ValueError(f"'{port_value}' is a read-only port. Only ports connected to the blackboard are writable")

        # Extract the key from the port_value
        key = port_value.strip("{}")

        blackboard = GlobalBlackboard.get_instance()

        # Lazy creation: if key doesn't exist, register it
        if not hasattr(blackboard, key):
            blackboard.register_key(key=key, access=py_trees.common.Access.WRITE, required=True)

        # Set the value for the key in the blackboard
        setattr(blackboard, key, value)
```

```
const App = () => {

    const [editorWidth, setEditorWidth] = useState(807);
    const [currentFilename, setCurrentFilename] = useState('');
    const [currentProjectname, setCurrentProjectname] = useState('visual_follow_person');
    const [modelJson, setModelJson] = useState('');
    const [projectChanges, setProjectChanges] = useState(false);
    const [gazeboEnabled, setGazeboEnabled] = useState(false);
    const [manager, setManager] = useState(null);

    var universe_config = { ... }

    useEffect(() => { ... }, [ ]);

    useUnload(() => {
        manager.disconnect();
    });

    const connectWithRetry = () => { ... }

    useEffect(() => { ... }, [manager]); // Re-run if the manager instance changes

    const onResize = (key, size) => { ... };

    return [
        <div className="App">
            <HeaderMenu />
            <div className="App-main" style={{ display: 'flex' }}>
                <div style={{ width: '200px' }}> ...
                </div>
                <Resizable ...
                    <div>
                        <DiagramEditor />
                        <VncViewer />
                    </div>
                </div>
            </div>
        </div>;
    ];
};

export default App;
```

Conclusiones



JdeRobot/bt-studio: A ROS-based visual IDE for creating Behavior Trees

A ROS-based visual IDE for creating Behavior Trees -
JdeRobot/bt-studio

 GitHub

Conclusiones



Desarrollos futuros

**Biblioteca de universos y
robots propios de los
usuarios.**

**Biblioteca de árboles de
comportamiento**

**Gracias por su
atención**



Preguntas