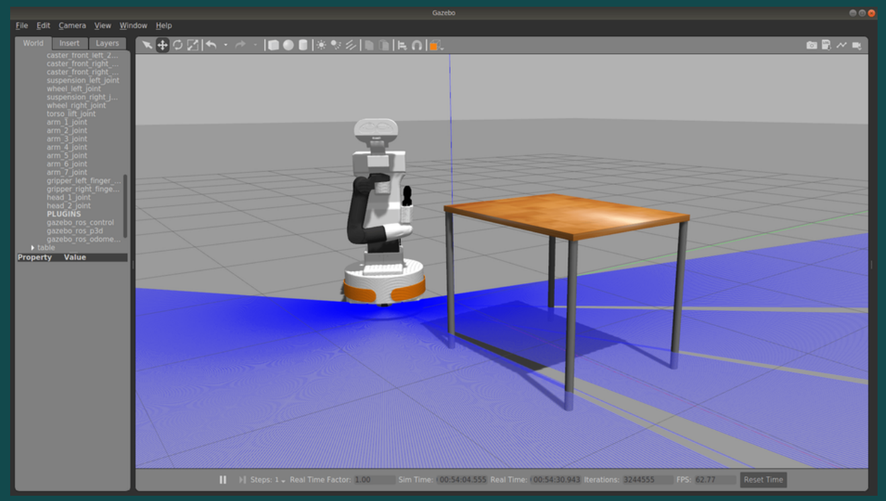
BT Studio

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

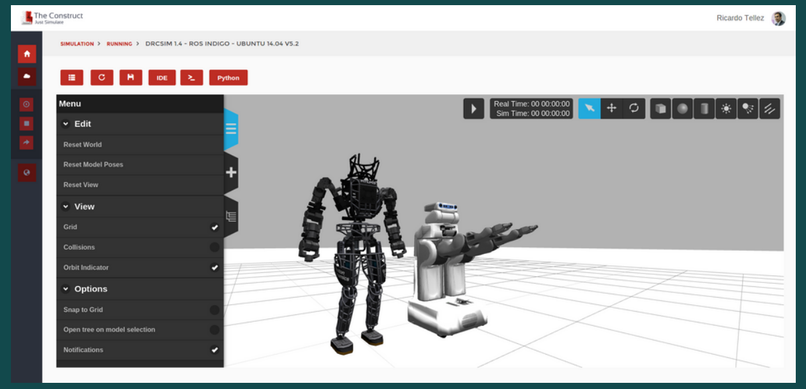
Autor: Óscar Martínez Martínez
Tutor: José María Cañas Plaza

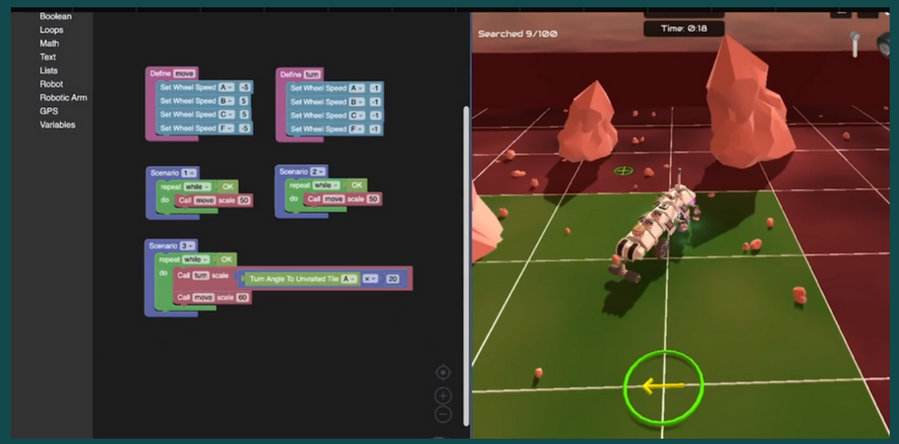
Introducción



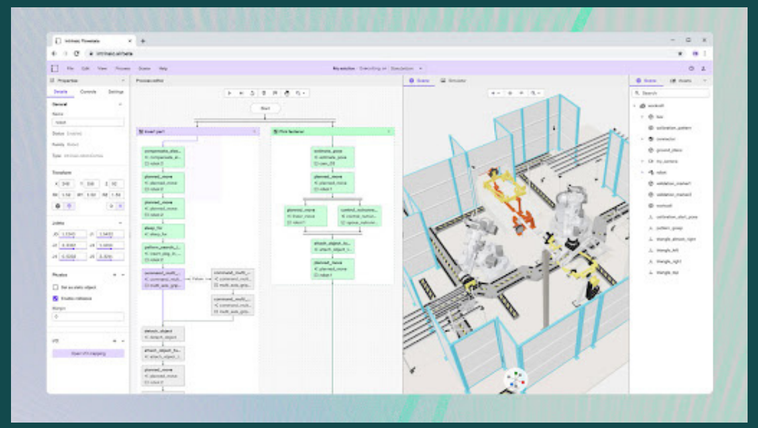
Introducción

Plataformas educativas





Plataformas profesionales



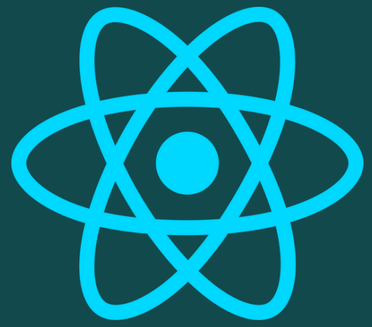
Objetivos

Desarrollo de un IDE web para la programación de aplicaciones robóticas

- **Editor visual**
- **Herramientas para generar código ejecutable**
- **Integración con entorno de ejecución dockerizado**
- **Integración en Unibotics**

Validación experimental: aplicaciones de ejemplo

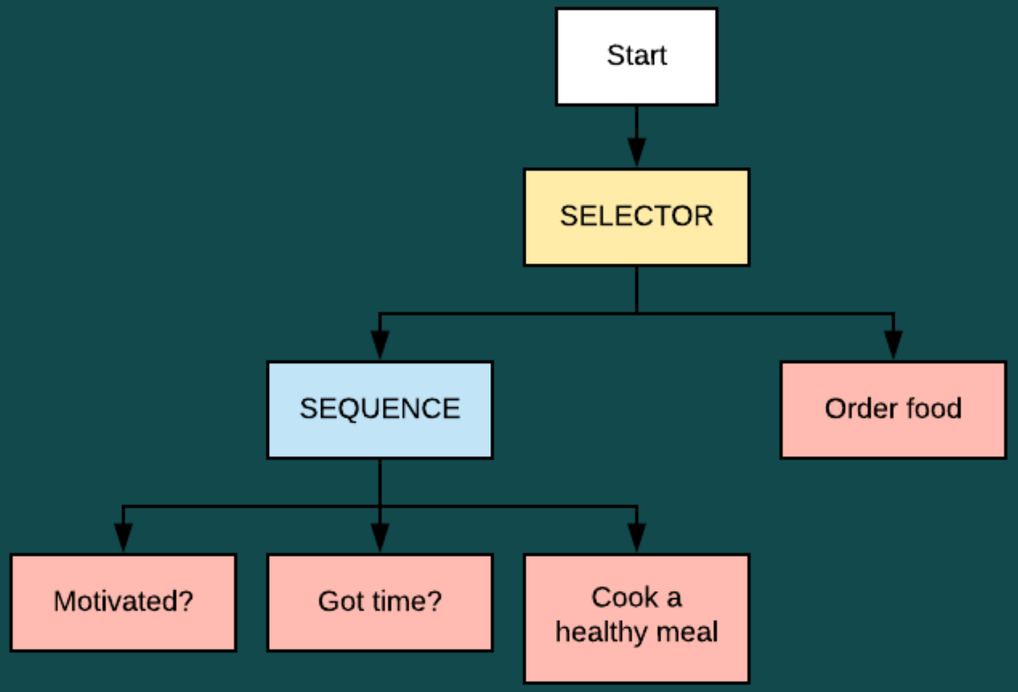
Fundamentos técnicos



React

django

Fundamentos técnicos



ROS 2

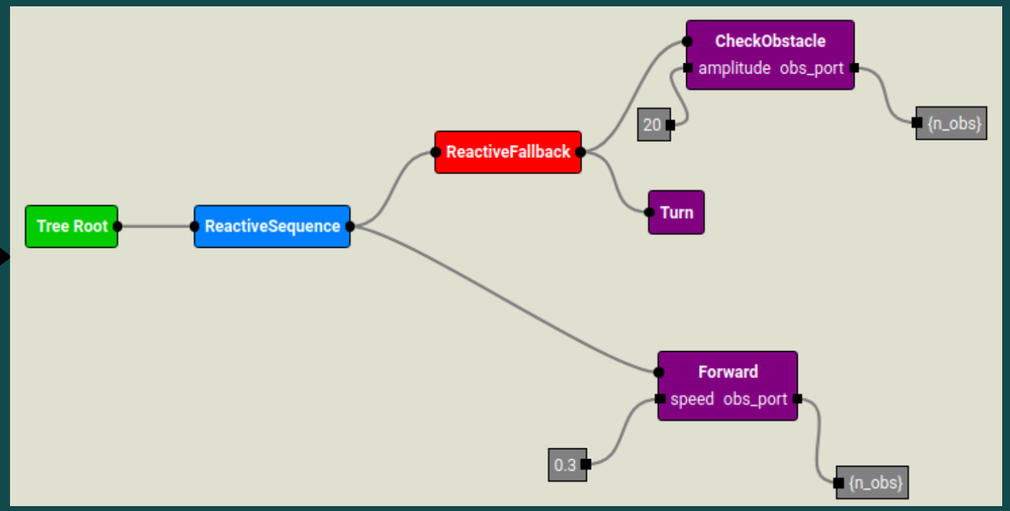
GAZEBO

BehaviorTree

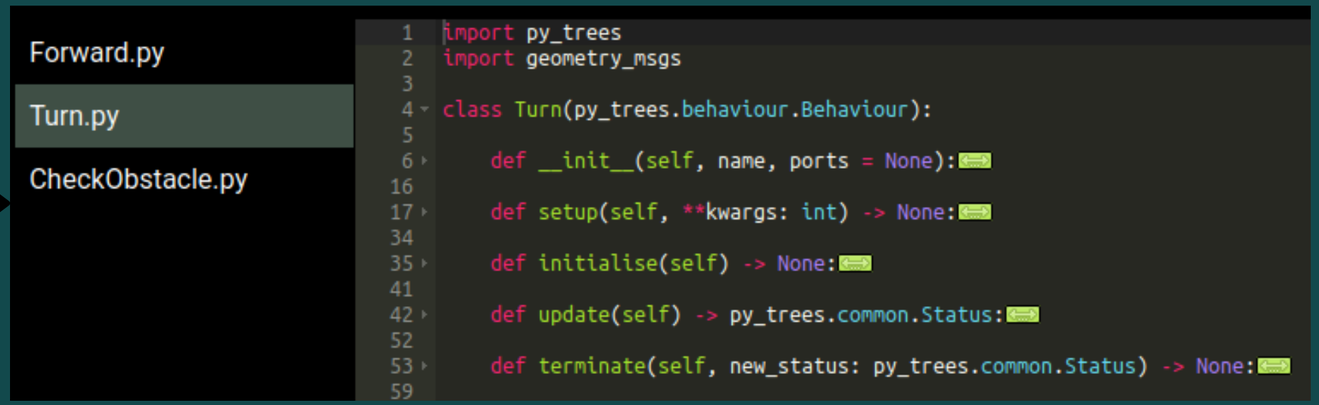
docker

BT Studio: definición de aplicaciones

Diagrama de árbol



Ficheros de las acciones en Python



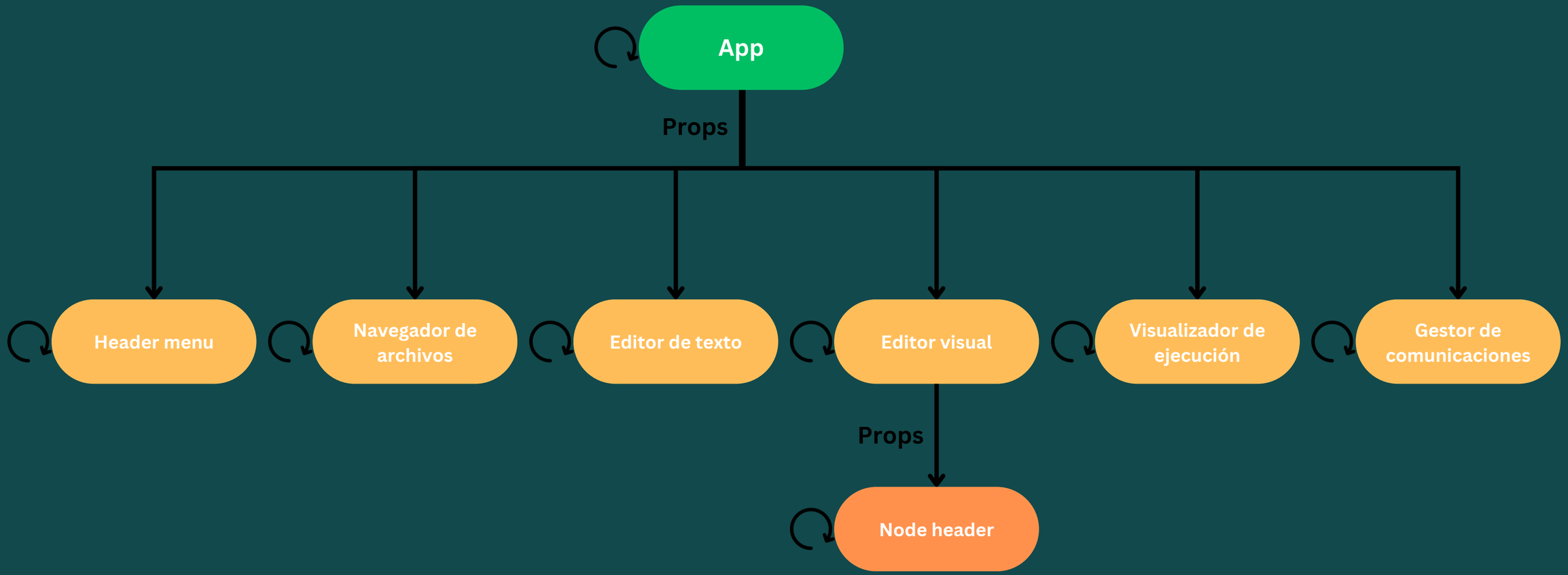
BT Studio: interfaz del IDE

Sevidor Django

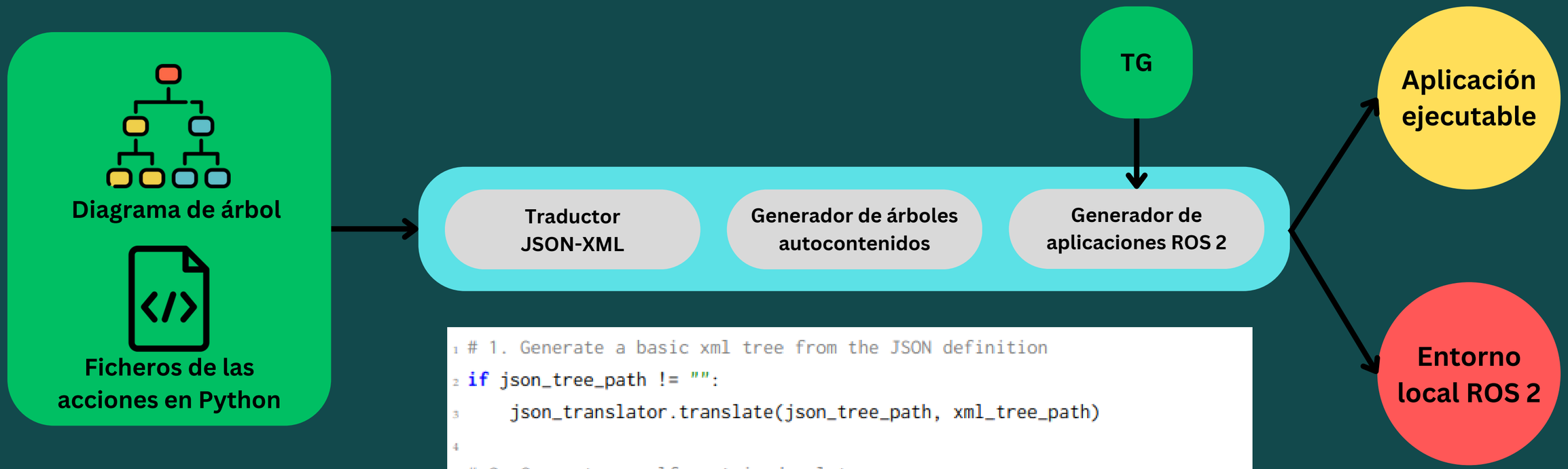
Frontend

Backend

BT Studio: interfaz del IDE

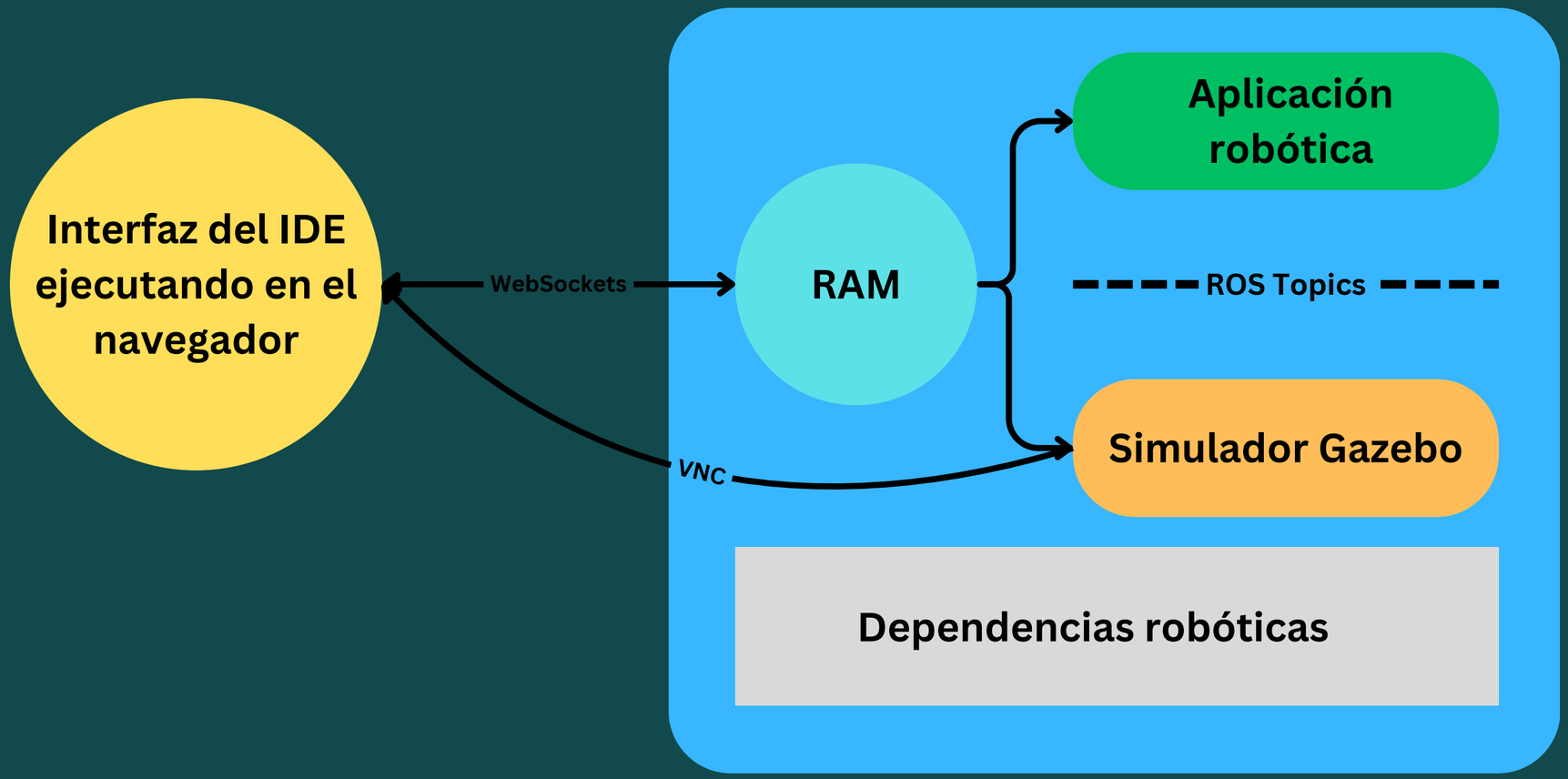


BT Studio: traductor de aplicaciones



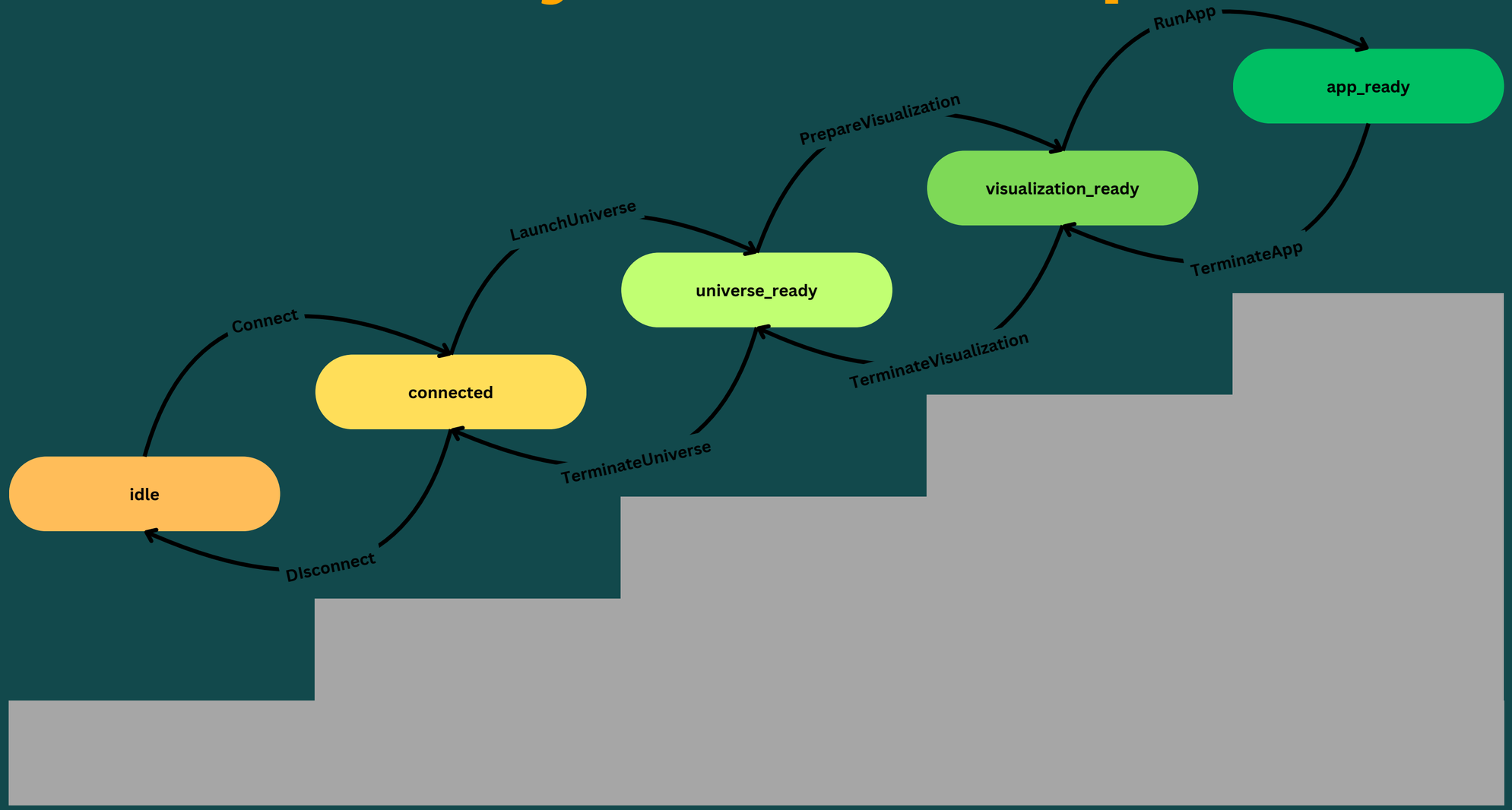

```
# 1. Generate a basic xml tree from the JSON definition
if json_tree_path != "":
    json_translator.translate(json_tree_path, xml_tree_path)

# 2. Generate a self-contained xml tree
tree_generator.generate(xml_tree_path, action_path,
    self_contained_tree_path)

# 3. Using the self-contained tree, package the ROS 2 app
zip_file_path = app_generator.generate(self_contained_tree_path, app_name
    , template_path, action_path, tree_gardener_src)
```

BT Studio: ejecutor de aplicaciones

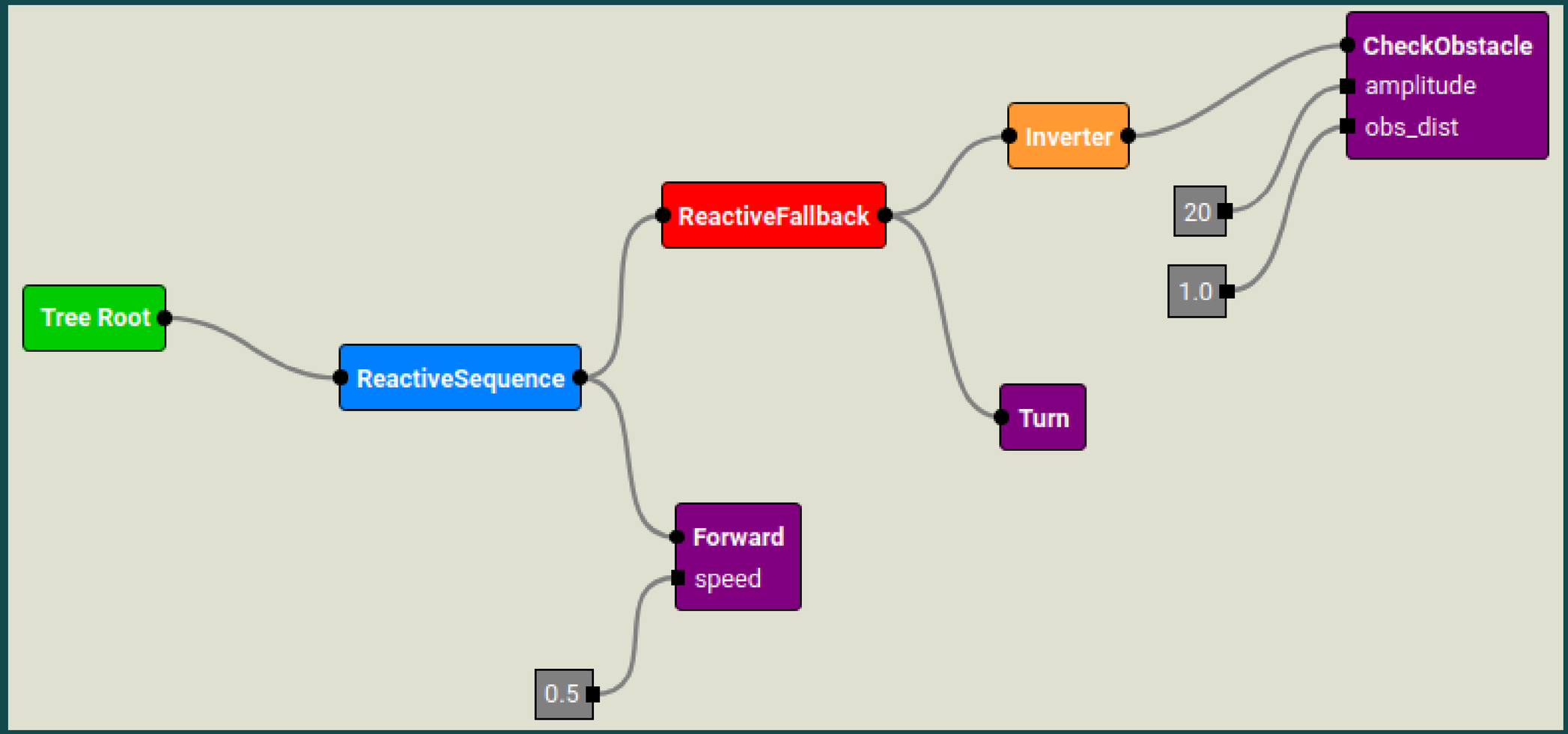


Ejecutor dockerizado

BT Studio: ejecutor de aplicaciones

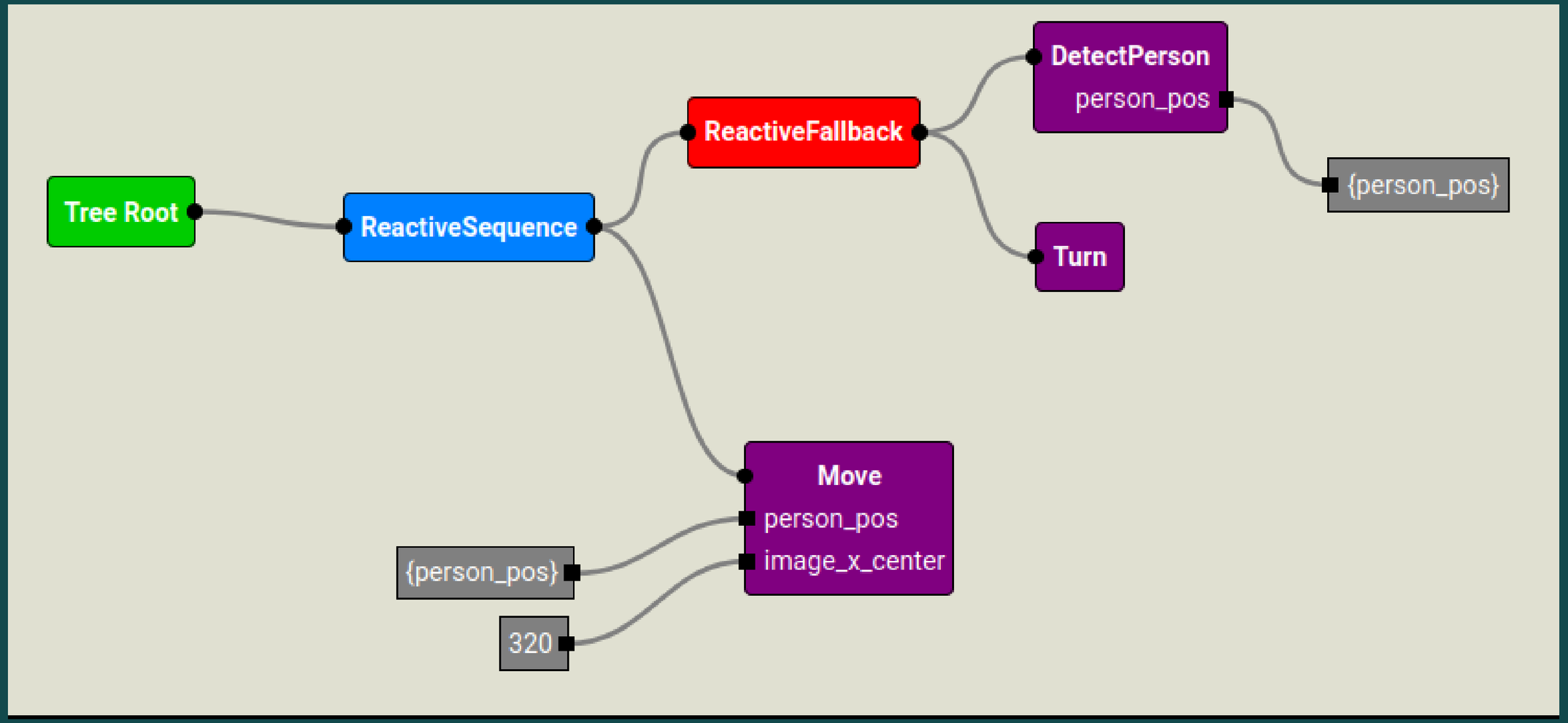


Validación experimental: Laser Bump and Go



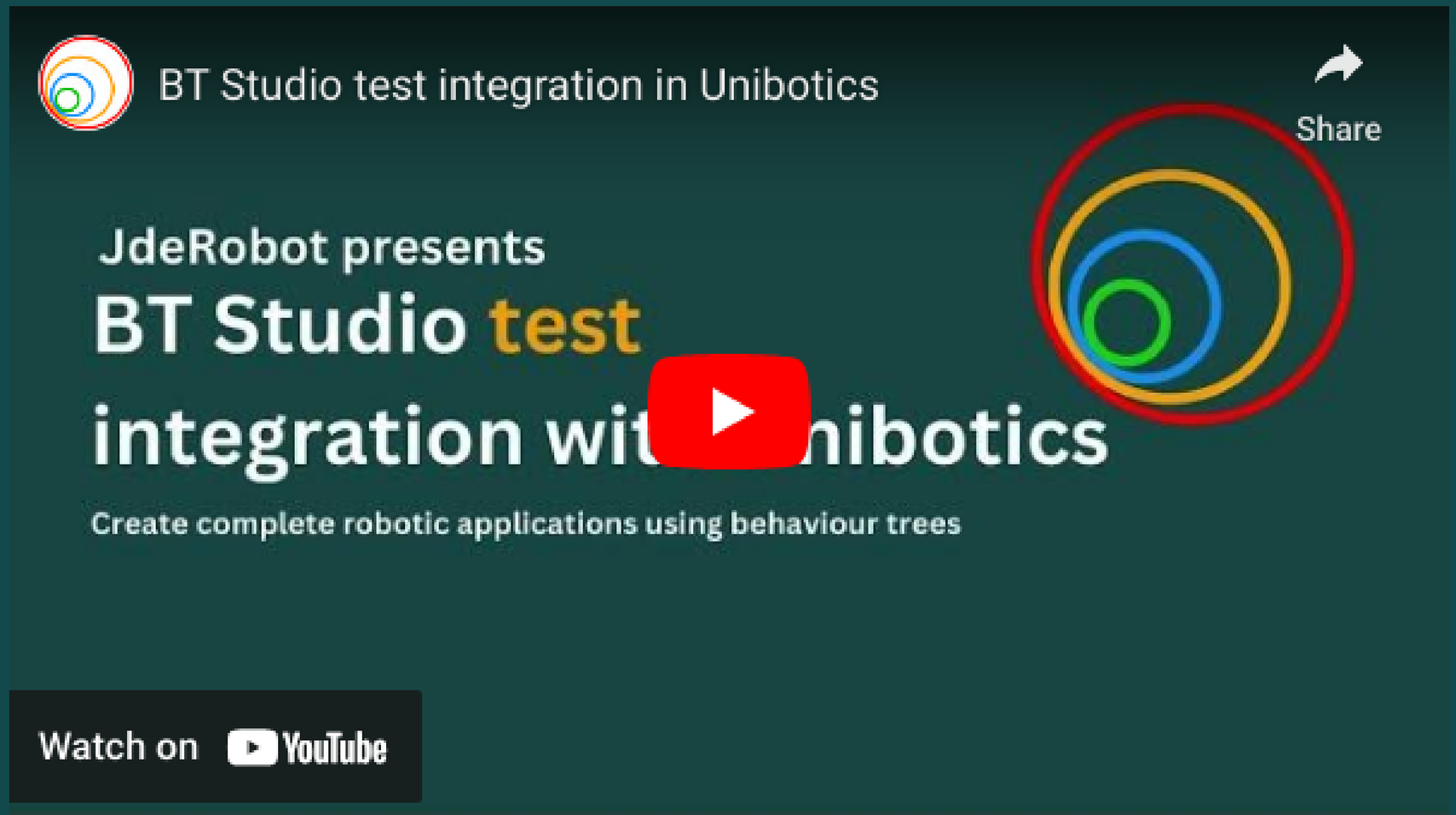
Validación experimental: Laser Bump and Go

Validación experimental: Visual Follow Person



Validación experimental: Visual Follow Person

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…
        </Resizable>

        <div>
          <DiagramEditor
          />
          <VncViewer
          />
        </div>

      </div>

    </div>
  );
};

export default App;
```

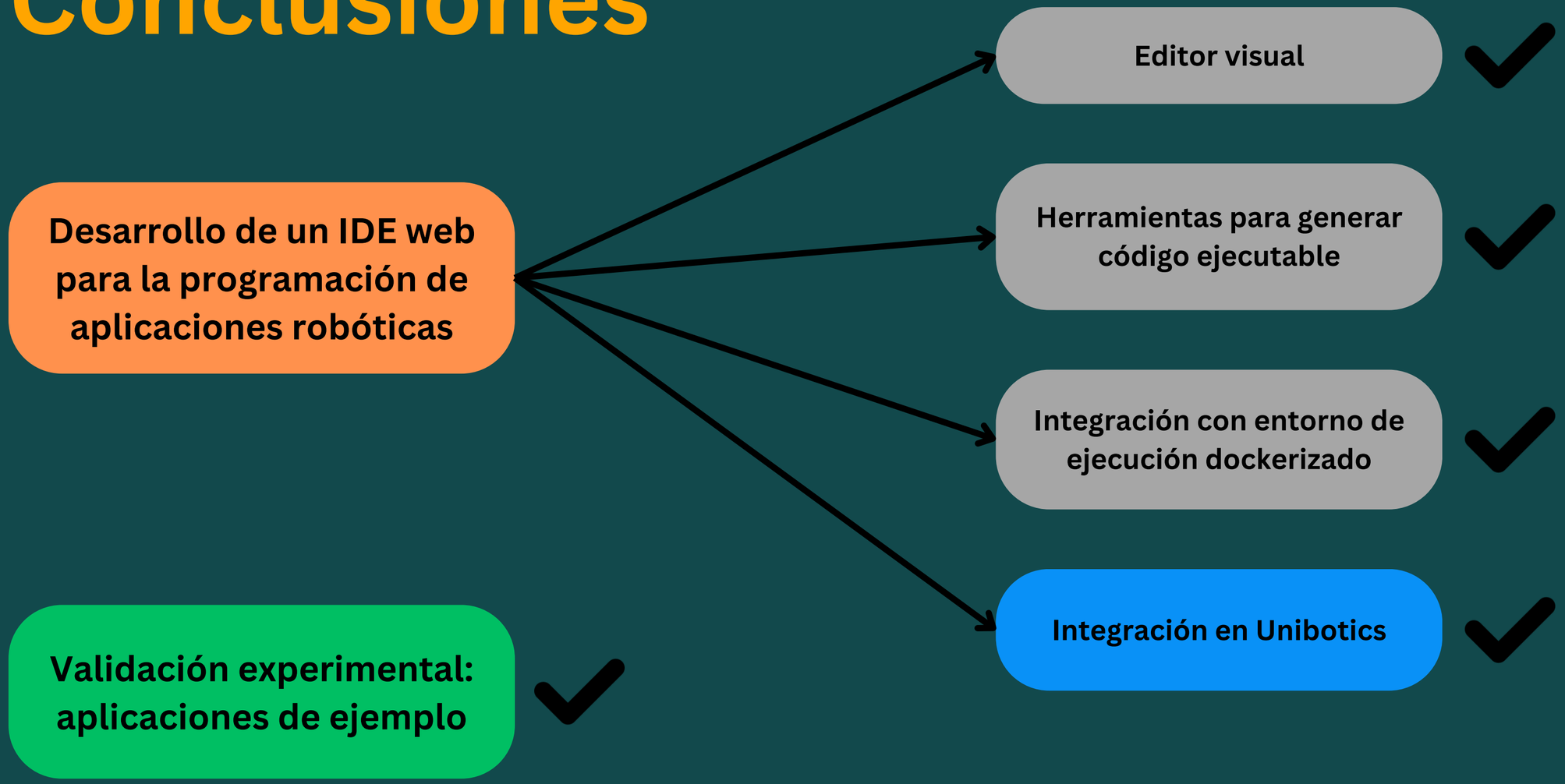
Conclusiones

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