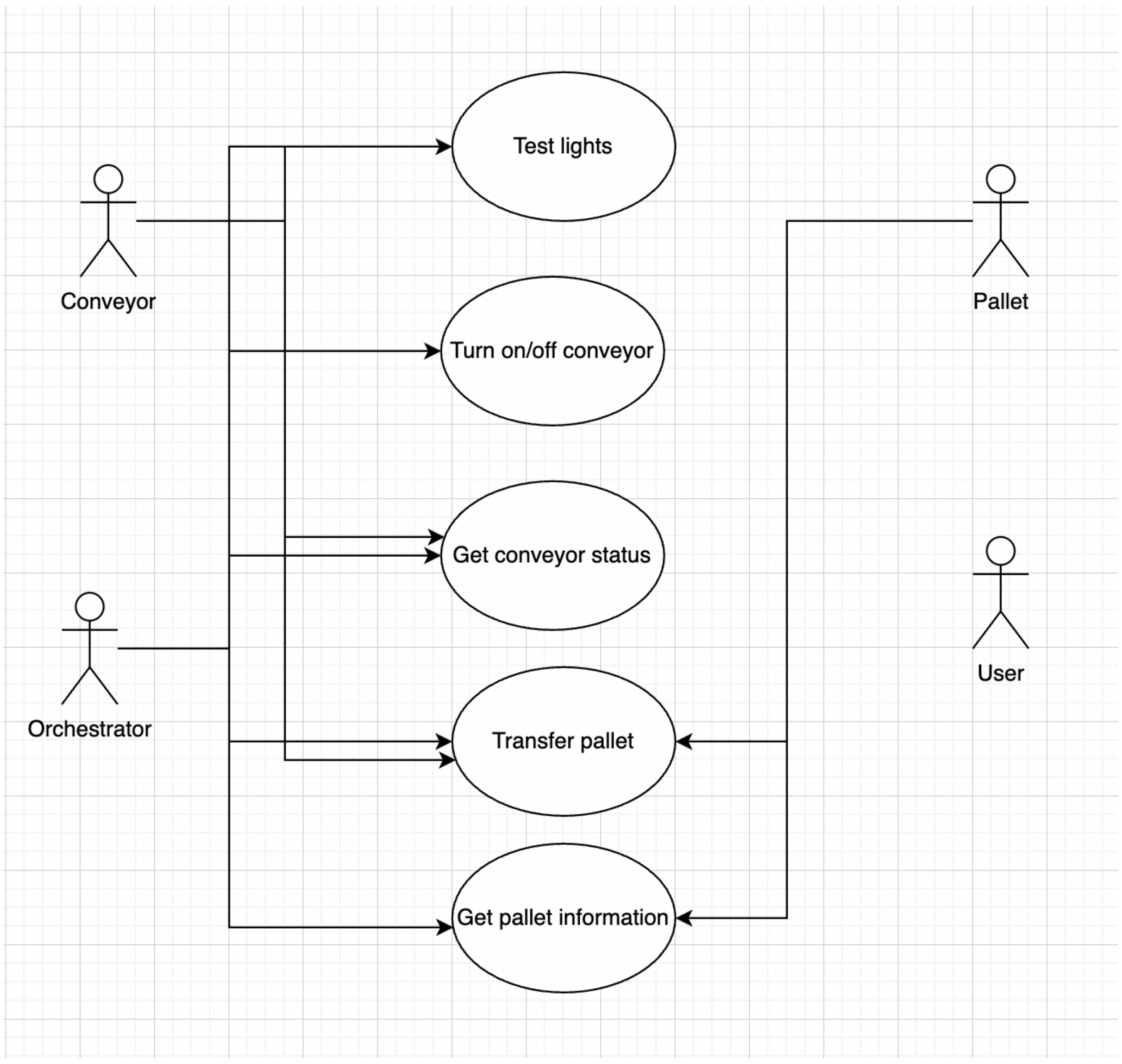
Design document

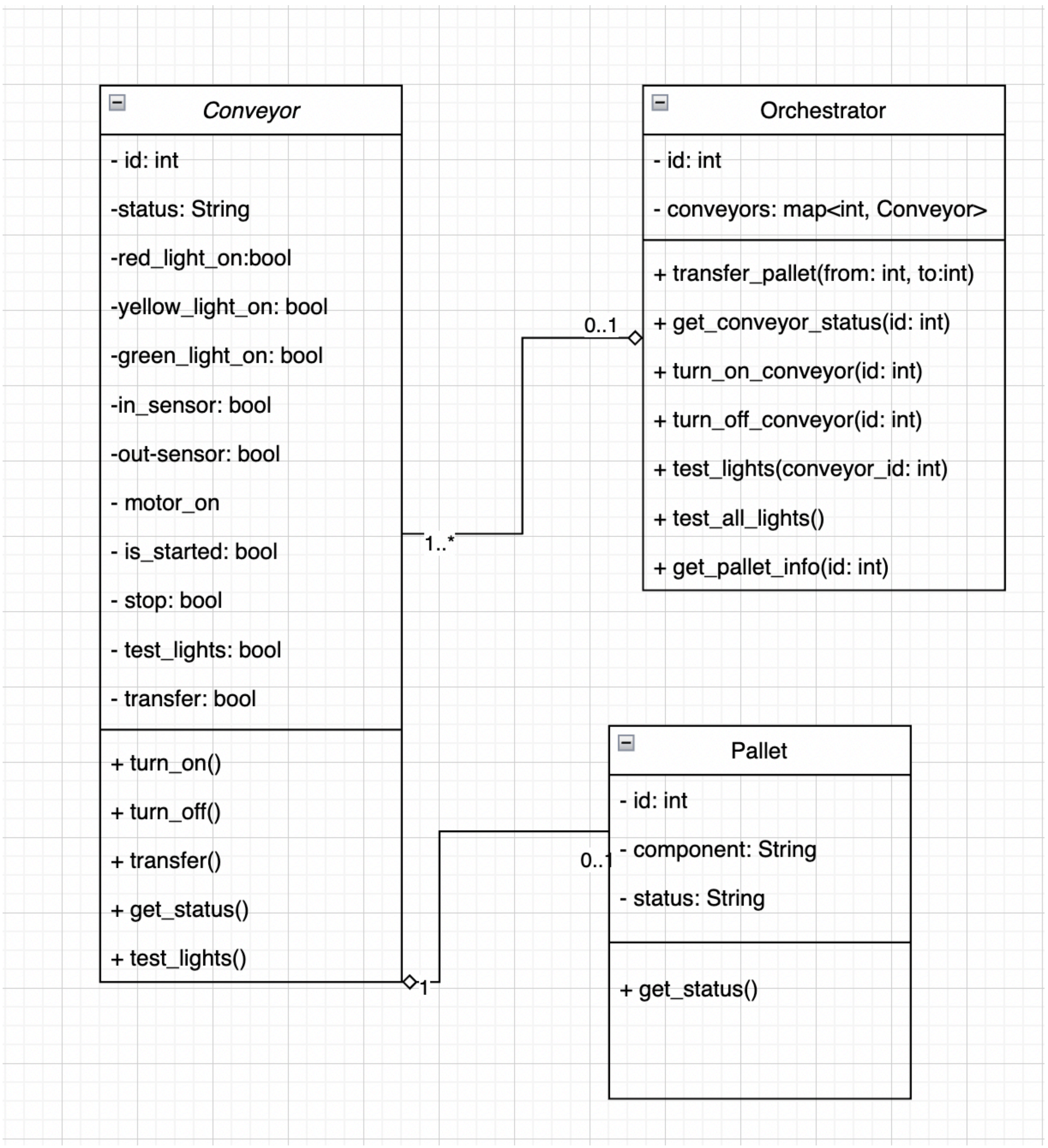
Use case diagram:

- From the requirements we identified these use casesd



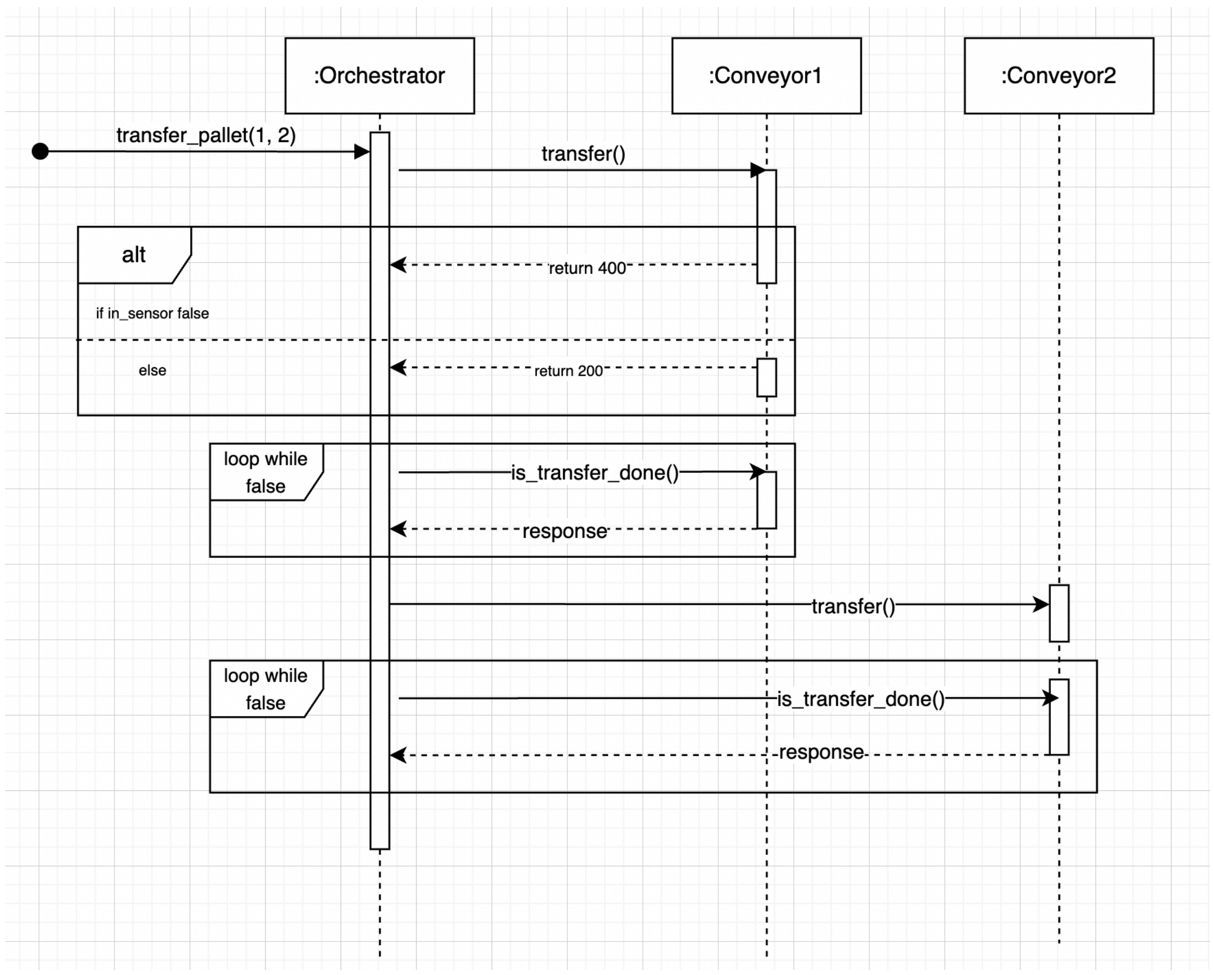
Class diagram:

- The system uses these classes: Conveyor, Pallet and Orchestrator
- For our purposes the pallet class is ignored and mocked by changing values of *in_sensor* and *out_sensor*



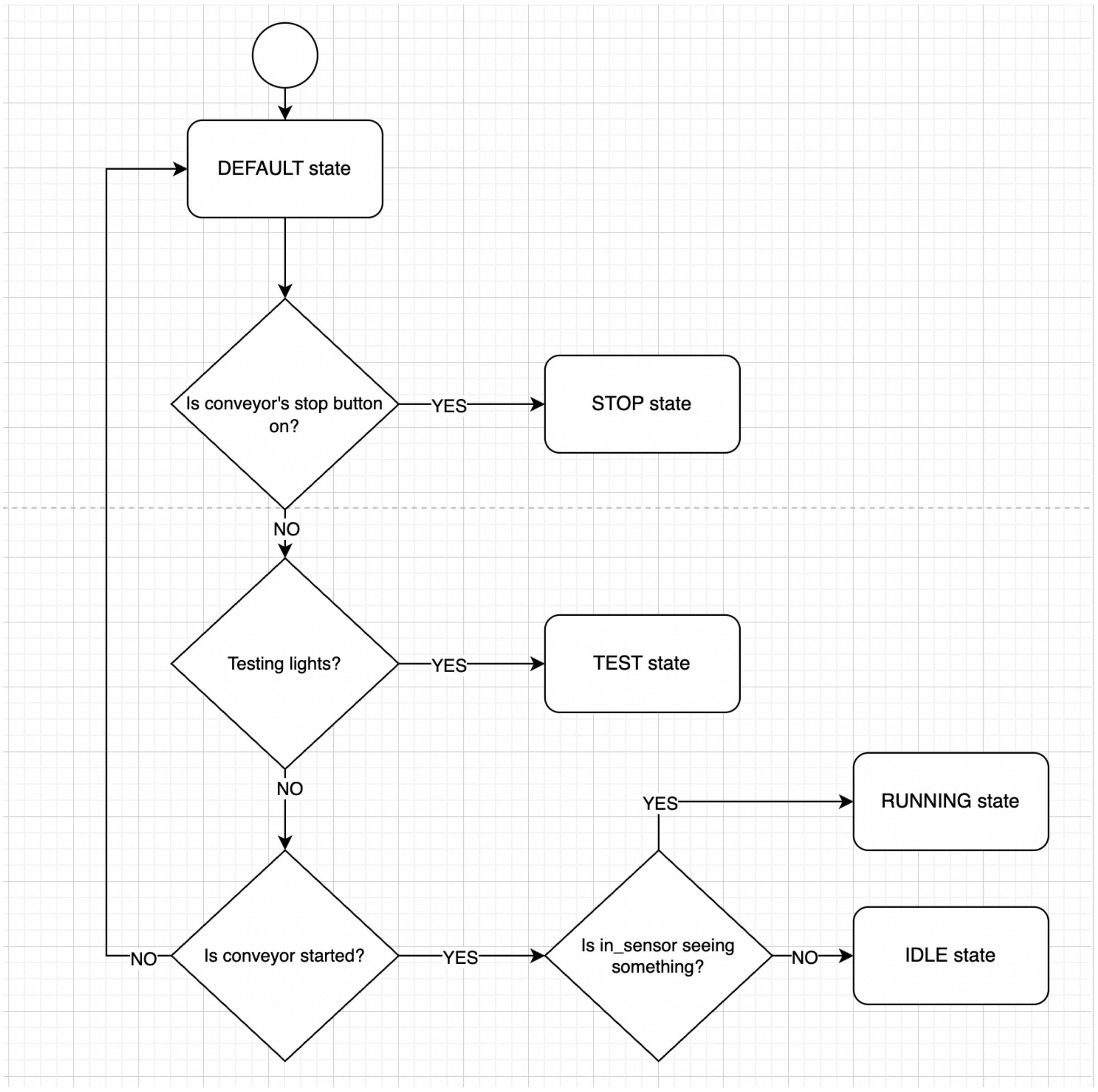
Sequence diagram - transfer_pallet(from, to):

- Main purpose of the system is transferring pallet between conveyors
- The logic behind the transferring is shown in the sequence diagram

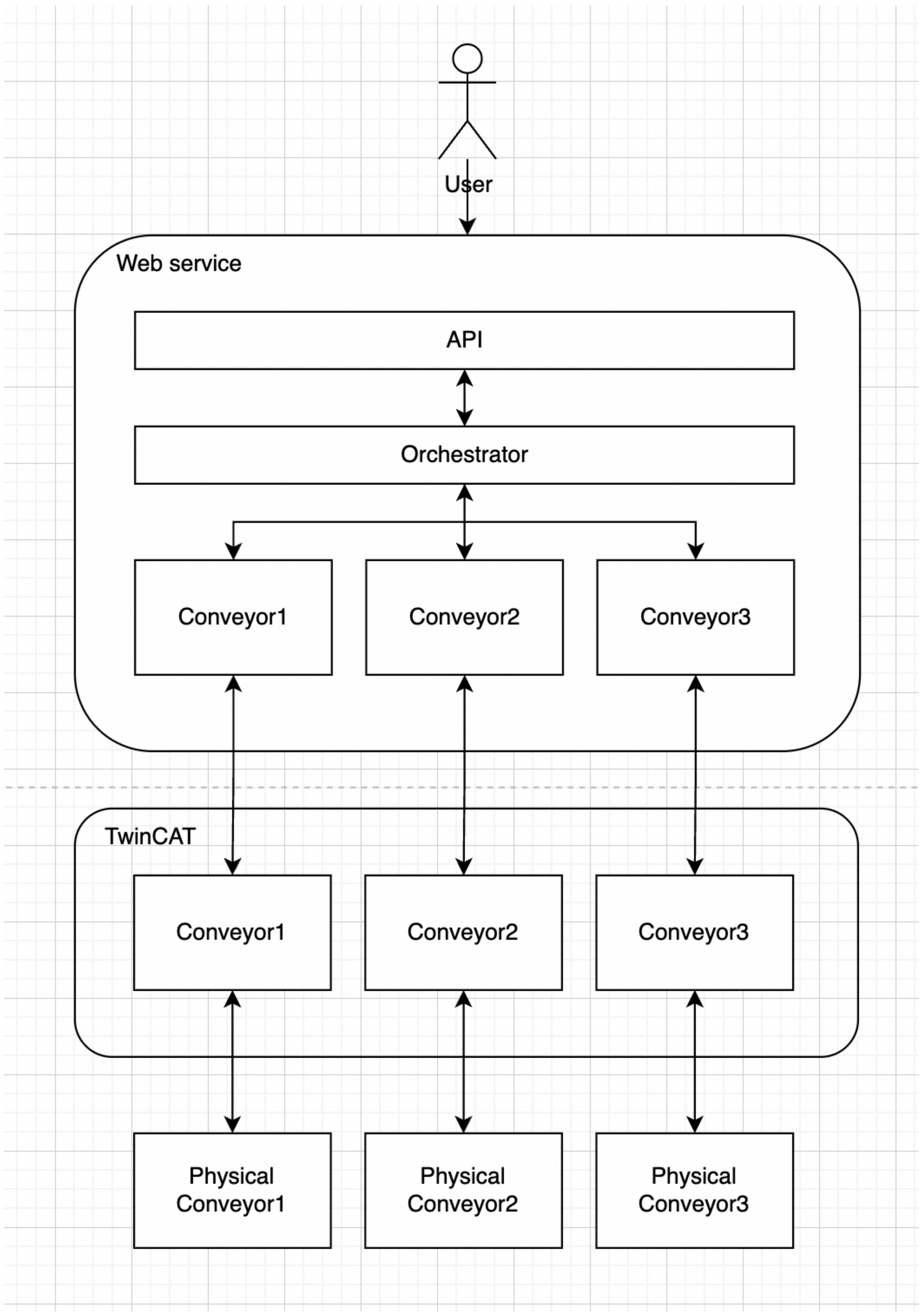


Conveyor state diagram:

- For understanding state machine of conveyors in the system we used the state diagram



System architecture



REST API design:

Conveyors:

GET:

- /conveyors/{id}
  - Get the status of a conveyor with given id

POST

- /conveyors/{id}/start
  - Turn on a conveyor with given id
- /conveyors/{id}/stop
  - Turn off a conveyor with given id
- /conveyors/{id}/testLights
  - Executes test of lights on conveyor with given id
- /conveyors/{id}/transfer
  - Executes transfer mechanism on conveyor with given id

Orchestrator:

POST

- /orchestrators/{id}/testAllLights
  - Executes test of lights on all conveyors through orchestrator
- /orchestrators/{id}/collectiveTransfer
  - Executes transfer mechanism from conveyor <*a*> to conveyor <*b*> given in body
  - Example request body:

```
{
  "from": 1,
  "to": 2
}
```