University of Tartu
Faculty of Mathematics and Computer Science
Institute of Computer Science

Project Scenario

MTAT.03.229 Enterprise System Integration

Authors:
Ardi Aasmaa (A84120)
Kadri Oluwagbemi (B04877)
Kristjan Veskimäe (A20792)
Polad Mahmudov (B79594)
Tural Ismayilov (B79598)

Supervisor:
Luciano García-Bañuelos

Tartu 2018

Table of Contents

Introduction

Plant hiring is the equipment rental from specialized rental companies to construction companies. In this project, we model and implement the procurement process via interconnected enterprise systems in rental and building company, called respectively RentIt and BuildIt.

In general, the design was kept as simple as possible. Due to time limitations, handling edge cases is out of scope from this project. Basic functionality, however, is provided for the requirements by this solution.

Part I: BuildIt

In this first part, we present the models for building company domain.

Domain model

First we present domain model in Figure 1. Identifiers are omitted from this conceptual model, except for PlantInventoryEntry and PurchaseOrder that have external URL as identifier. PIE rental price is assumed to vary and is therefore not persisted into the DB. For simplicity, PHR can only be extended once.

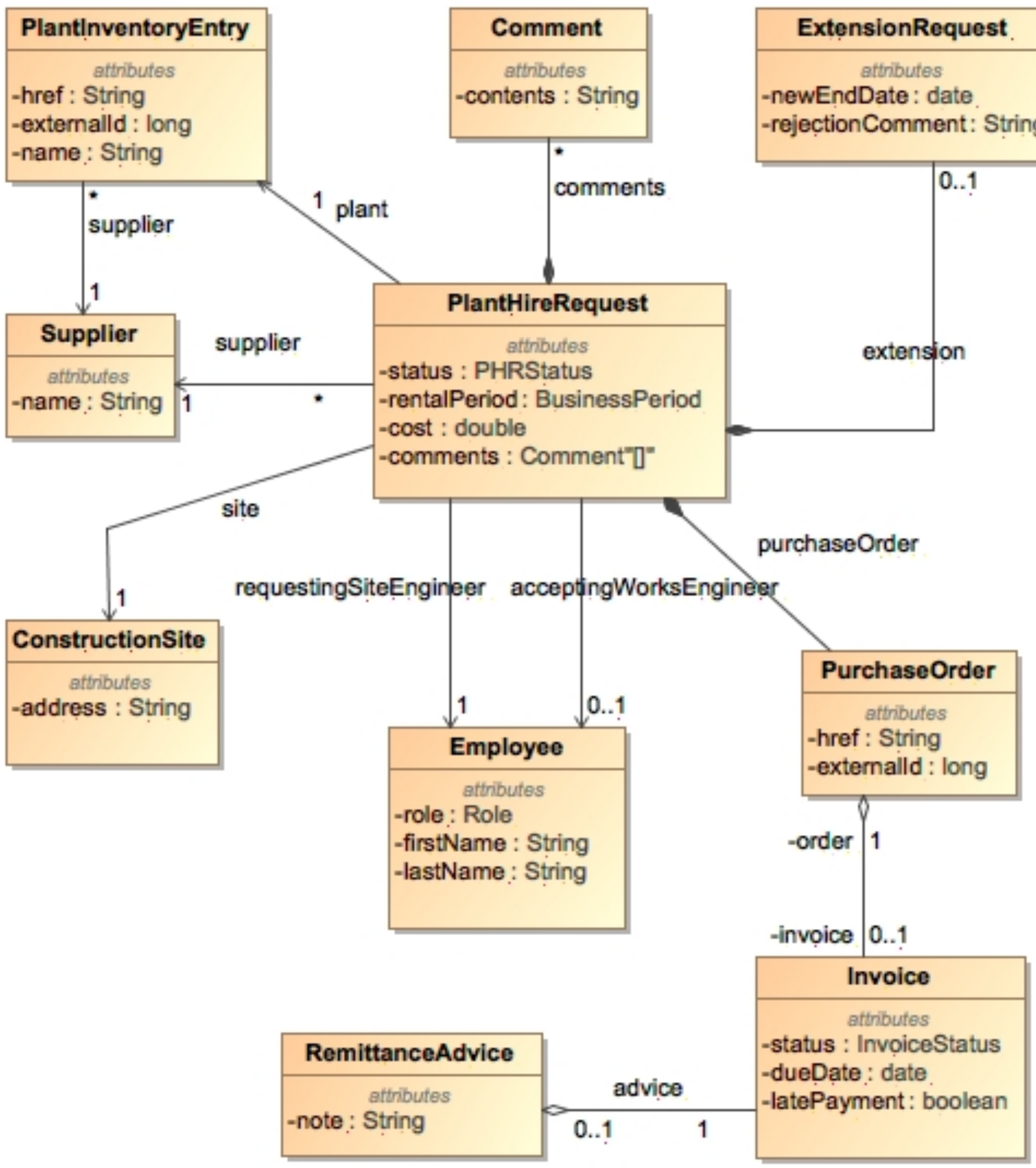


Figure 1 BuildIt domain model

The corresponding data types for BuildIt are in Figure 2.

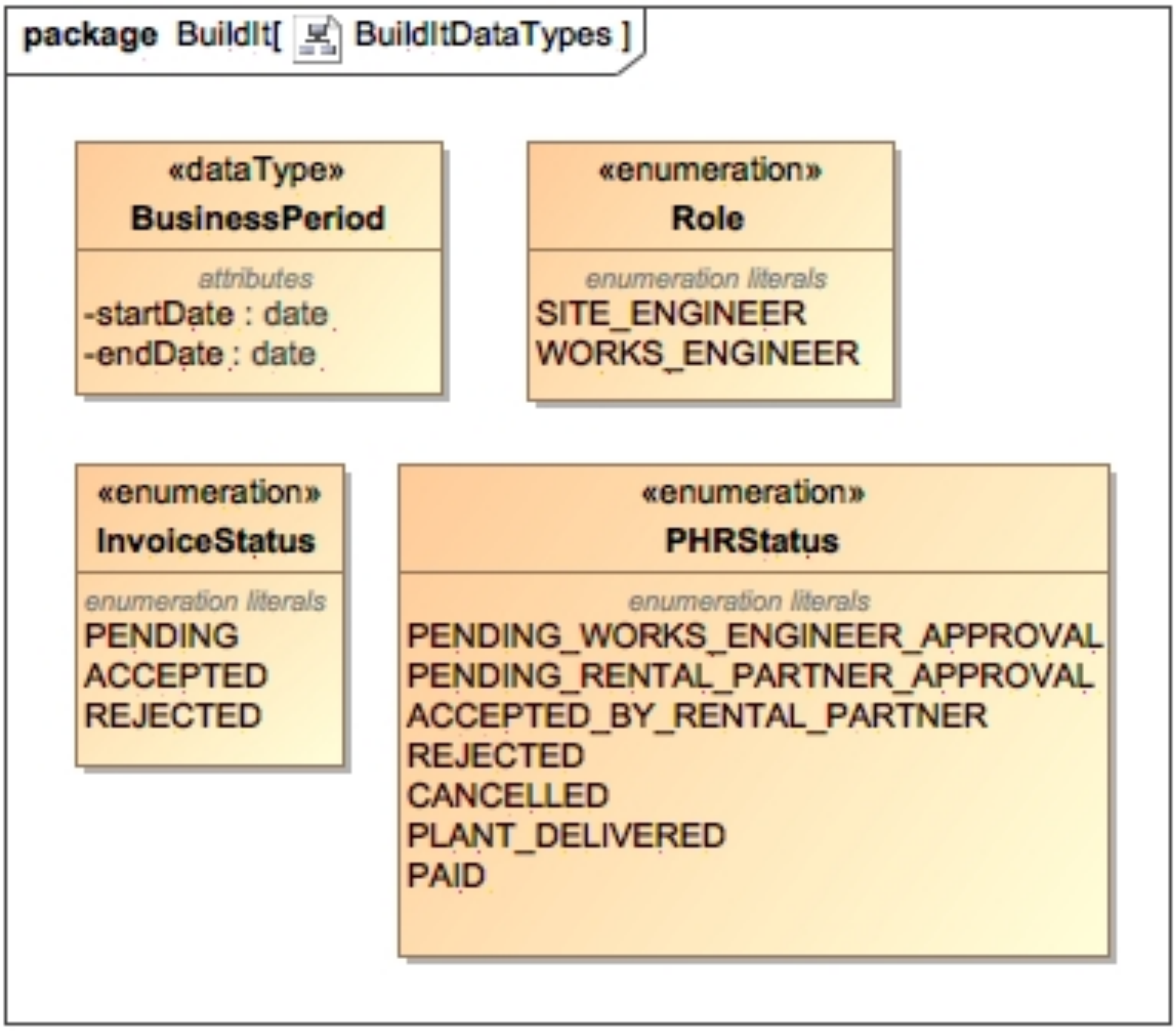
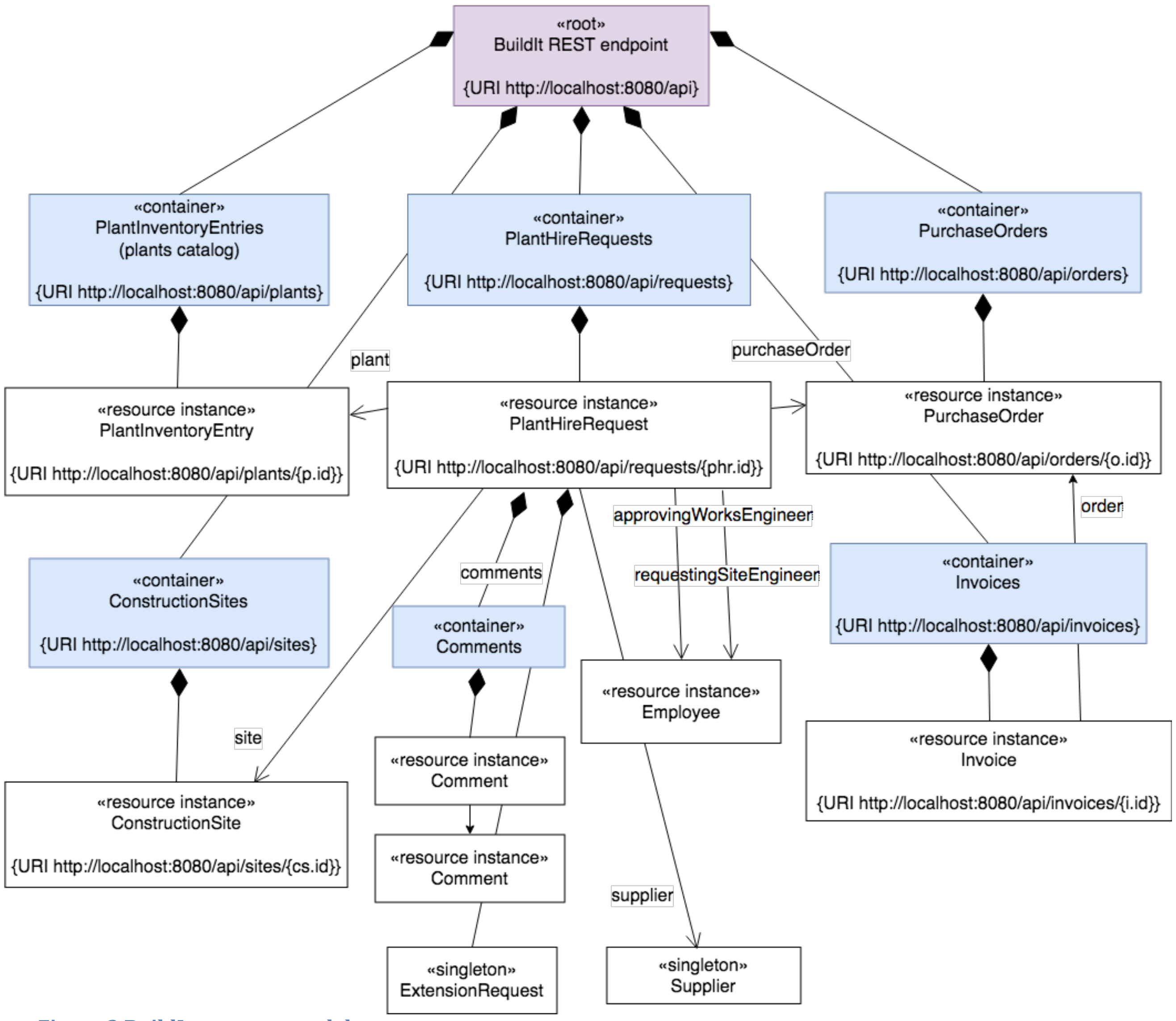


Figure 2 BuildIt Data Types

Resource model

Resource model helps designing RESTful applications. It is depicted in Figure 3 for BuildIt.



«root» BuildIt REST endpoint {URI http://localhost:8080/api}
«container» PlantInventoryEntries (plants catalog) {URI http://localhost:8080/api/plants}
«container» PlantHireRequests {URI http://localhost:8080/api/requests}
«container» PurchaseOrders {URI http://localhost:8080/api/orders}
plant
purchaseOrder
«resource instance» PlantInventoryEntry {URI http://localhost:8080/api/plants/{p.id}}
«resource instance» PlantHireRequest {URI http://localhost:8080/api/requests/{phr.id}}
«resource instance» PurchaseOrder {URI http://localhost:8080/api/orders/{o.id}}
approvingWorksEngineer
order
comments
requestingSiteEngineer
«container» Invoices {URI http://localhost:8080/api/invoices}
«container» ConstructionSites {URI http://localhost:8080/api/sites}
«container» Comments
«resource instance» Employee
site
«resource instance» Comment
«resource instance» Invoice {URI http://localhost:8080/api/invoices/{i.id}}
«resource instance» ConstructionSite {URI http://localhost:8080/api/sites/{cs.id}}
«resource instance» Comment
supplier
«singleton» ExtensionRequest
«singleton» Supplier

Figure 3 BuildIt resource model

State models

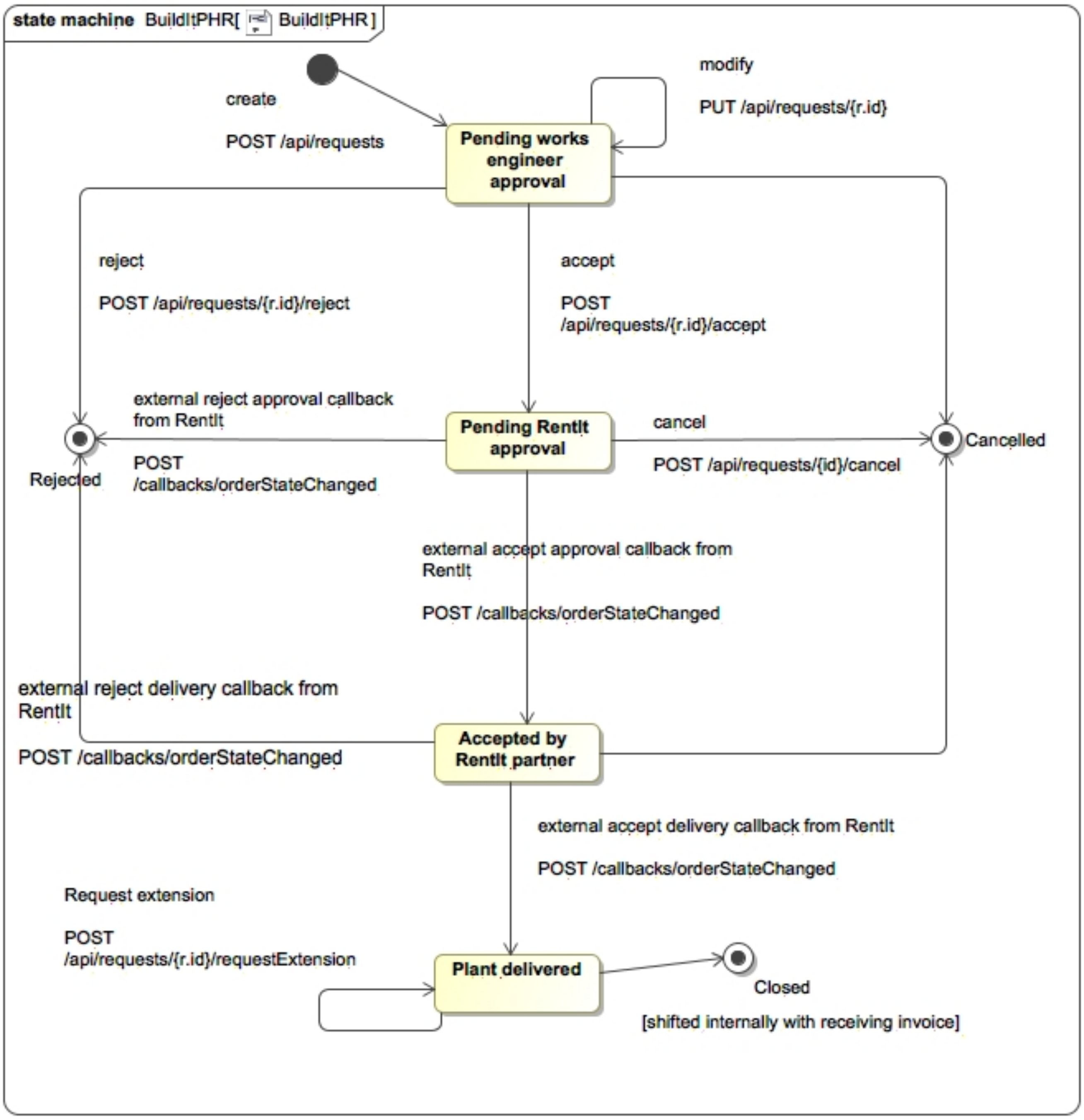
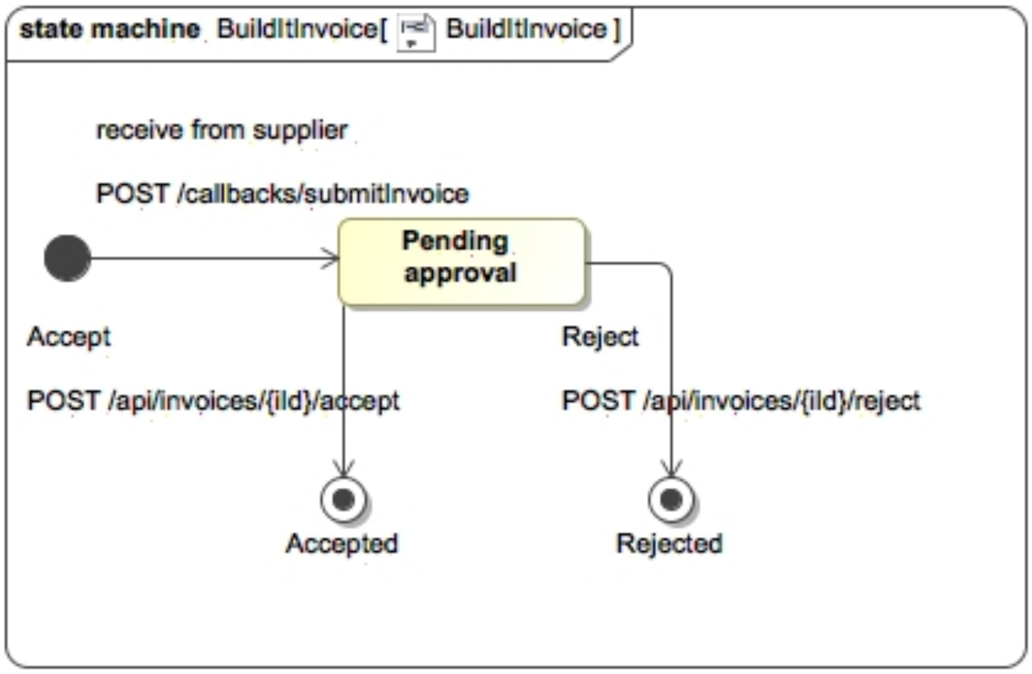

state machine BuildItPHR[BuildItPHR]
create
POST /api/requests
modify
PUT /api/requests/{r.id}
Pending works engineer approval
reject
POST /api/requests/{r.id}/reject
accept
POST /api/requests/{r.id}/accept
external reject approval callback from RentIt
POST /callbacks/orderStateChanged
Pending RentIt approval
cancel
POST /api/requests/{id}/cancel
Rejected
Cancelled
external accept approval callback from RentIt
POST /callbacks/orderStateChanged
external reject delivery callback from RentIt
POST /callbacks/orderStateChanged
Accepted by RentIt partner
external accept delivery callback from RentIt
POST /callbacks/orderStateChanged
Request extension
POST /api/requests/{r.id}/requestExtension
Plant delivered
Closed
[shifted internally with receiving invoice]

Figure 4 State model for plant hire request

State model for plant hire request is depicted in Figure 4.

Notes about plant hire request state model: "Pending extension" state goes back to plant delivered state regardless of the extension being accepted or rejected. If the extension is accepted, the end date of plant hire request gets updated. No history of extension success is kept - this is left as future work. RentIt side is responsible for updating state according to delivery. This should be amended with back-up process on BuildIt side.


state machine BuildItInvoice[BuildItInvoice]
receive from supplier
POST /callbacks/submitInvoice
Pending approval
Accept
POST /api/invoices/{iId}/accept
Reject
POST /api/invoices/{iId}/reject
Accepted
Rejected
Figure 5 State model for invoice

State model for invoice is depicted in Figure 5.

Apiary blueprint

The Apiary blueprint for BuildIt system is located at:

https://buildit28.docs.apiary.io/

BuildIt API exposes the following endpoints, with their function described:

- (CC1) Create Plant Hire Request (PHR)
    - Query plants catalog - site engineer can thereafter pick a plant to rent;
    - Retrieve list of construction sites;
    - Create Plant Hire Request (PHR) – by site engineer, JSON payload contains:
        - Plant ID with corresponding supplier ID (both can be retrieved from the chosen plant's data transfer, for end-user the supplier is quite transparent in here);
        - Construction site (e.g. site engineer can select a site from drop-down in a plant hire request form)
        - Rental period
- View all PHRs / (CC4) one PHR by ID – both site and works engineer;
- (CC2, CC5) Modify PHR by ID – both engineers;
- Trigger operation on PHR:
    - (CC5) Accept/reject – by works engineer;
    - (CC3) cancel - by site engineer;
    - Comment – both engineers;
    - (CC8) Request extension – by site engineer;
- (CC7) View all Purchase Orders (PO) – both engineers;
- Calls from external partner RentIt:
    - (CC6) Receive PHR status update callback - from integrated RentIt partner;
    - (CC8) Receive extension status update callback - from integrated RentIt partner;
    - (CC9, CC10) Receive invoice - from integrated RentIt partner – it must contain the corresponding PO ID;
- (C11) Approve Invoice
    - Retrieve all invoices, optionally filtering by status – so they could be approved;
    - (C11) Retrieve PO associated with an invoice
    - (C11) Approve/reject invoice

Part II: RentIt

In this second part, we present the models for plant hiring company domain.

Domain model

First we present domain model for RentIt sales module in Figure 6. Identifiers are again omitted from this conceptual model.

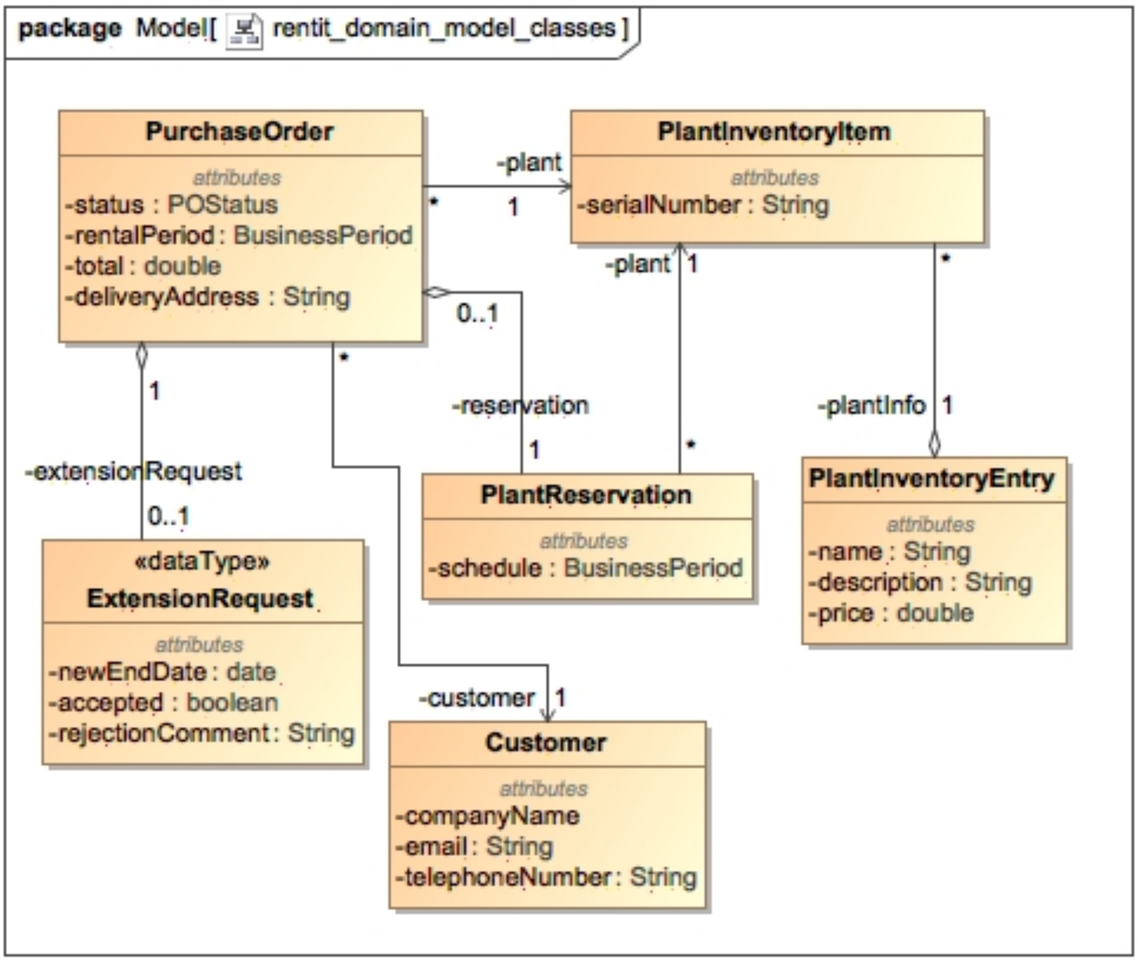


Figure 6 RentIt domain model for sales submodule

Plant reservation is rather redundant in here. It was still kept and points optionally to purchase order. In future, maintenance tasks can reserve a plant as well in addition to purchase orders.

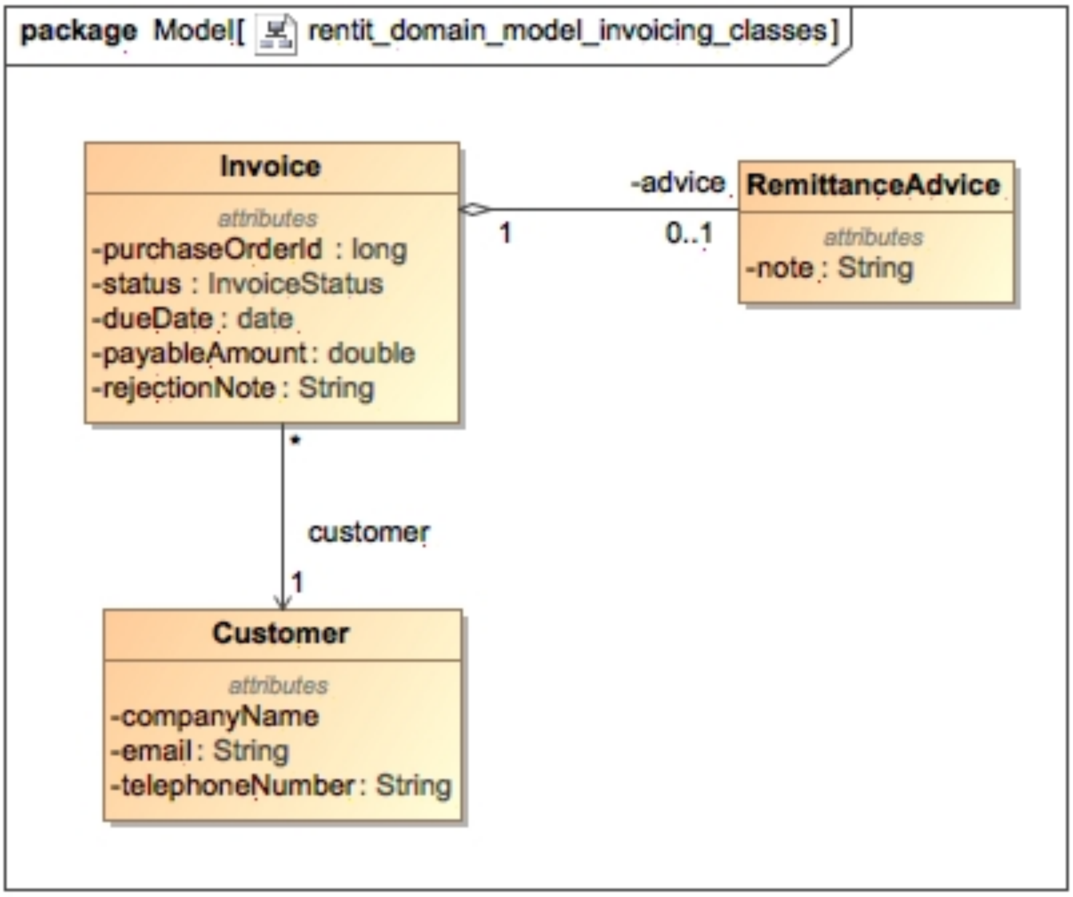


Figure 7 RentIt domain model for invoicing submodule

Domain model for invoicing is in Figure 7.

The corresponding data types for RentIt are in Figure 8.

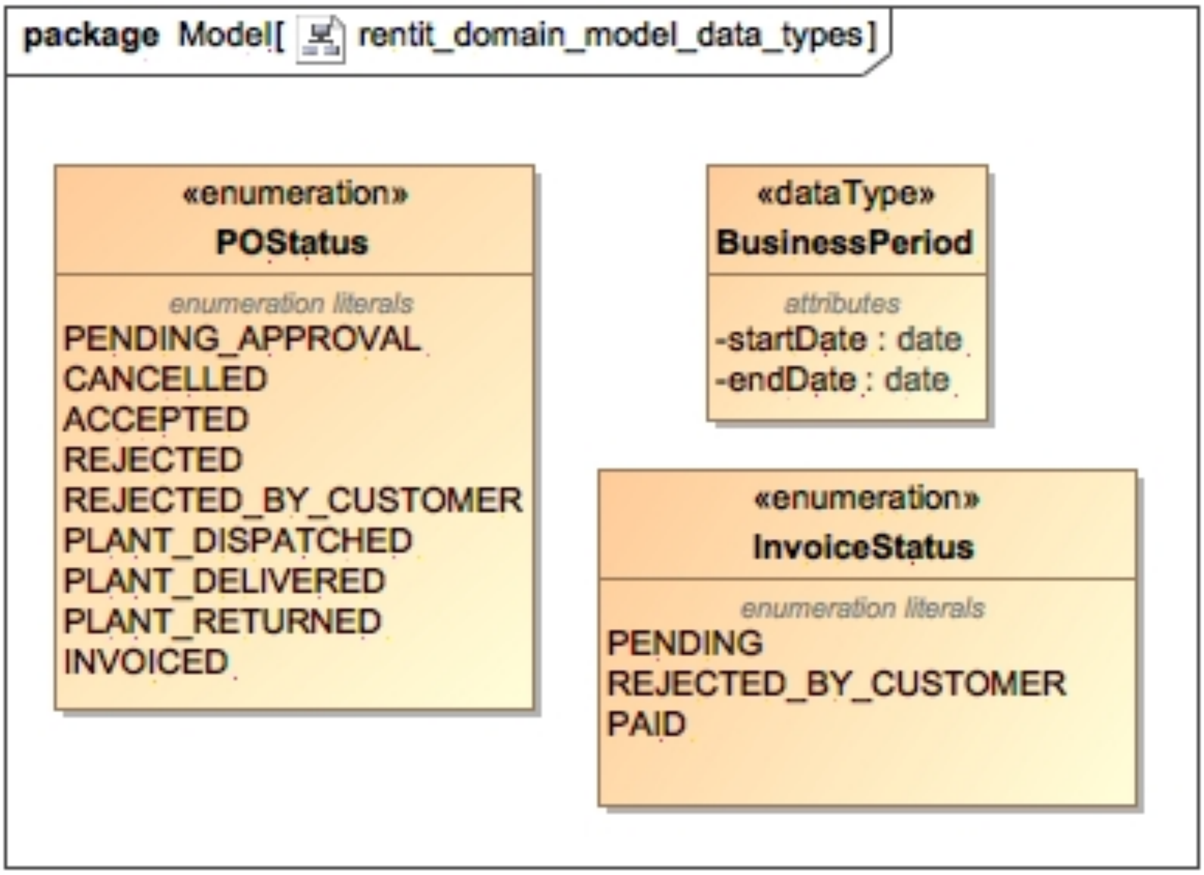


Figure 8 RentIt Data Types

Resource model

Resource model is depicted in Figure 9 for RentIt.

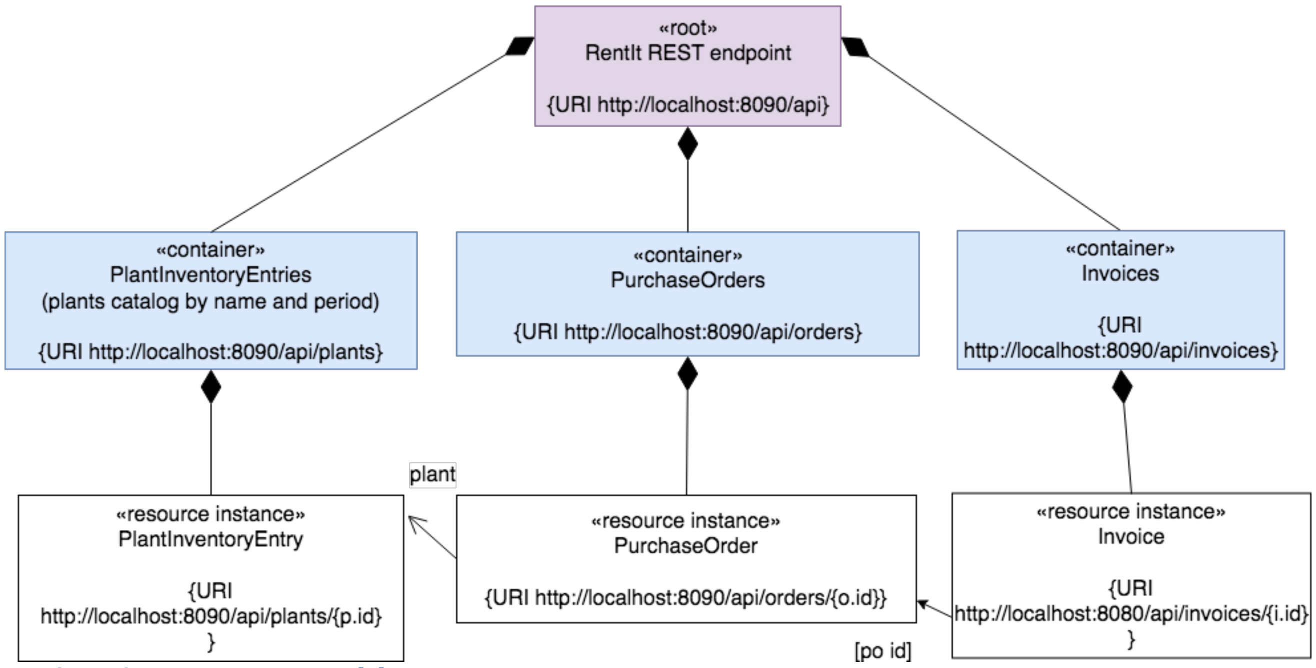

«root»
RentIt REST endpoint
{URI http://localhost:8090/api}
«container»
PlantInventoryEntries
(plants catalog by name and period)
{URI http://localhost:8090/api/plants}
«container»
PurchaseOrders
{URI http://localhost:8090/api/orders}
«container»
Invoices
{URI http://localhost:8090/api/invoices}
plant
«resource instance»
PlantInventoryEntry
{URI http://localhost:8090/api/plants/{p.id} }
«resource instance»
PurchaseOrder
{URI http://localhost:8090/api/orders/{o.id}}
«resource instance»
Invoice
{URI http://localhost:8080/api/invoices/{i.id} }
[po id]

Figure 9 RentIt resource model

State models

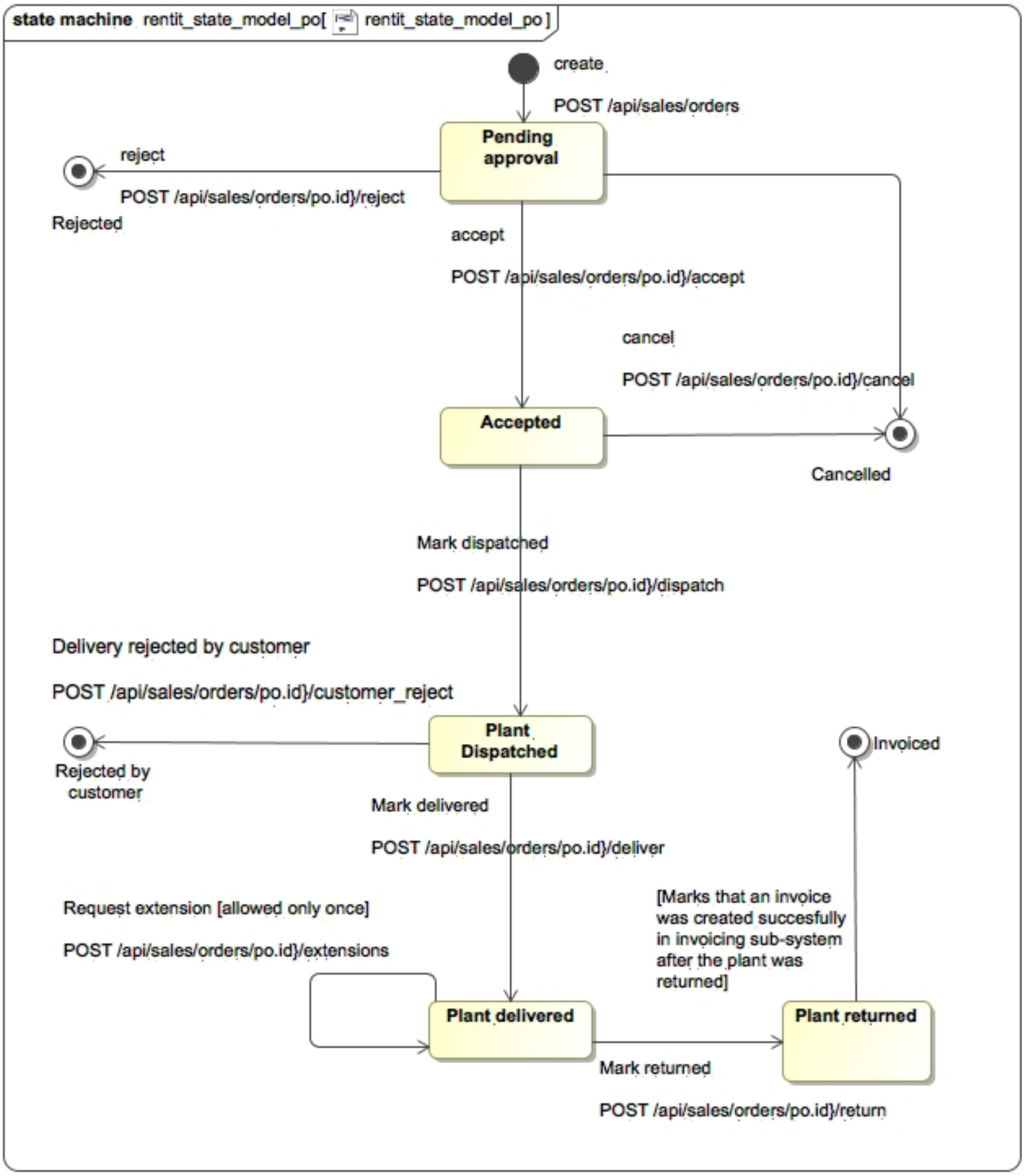
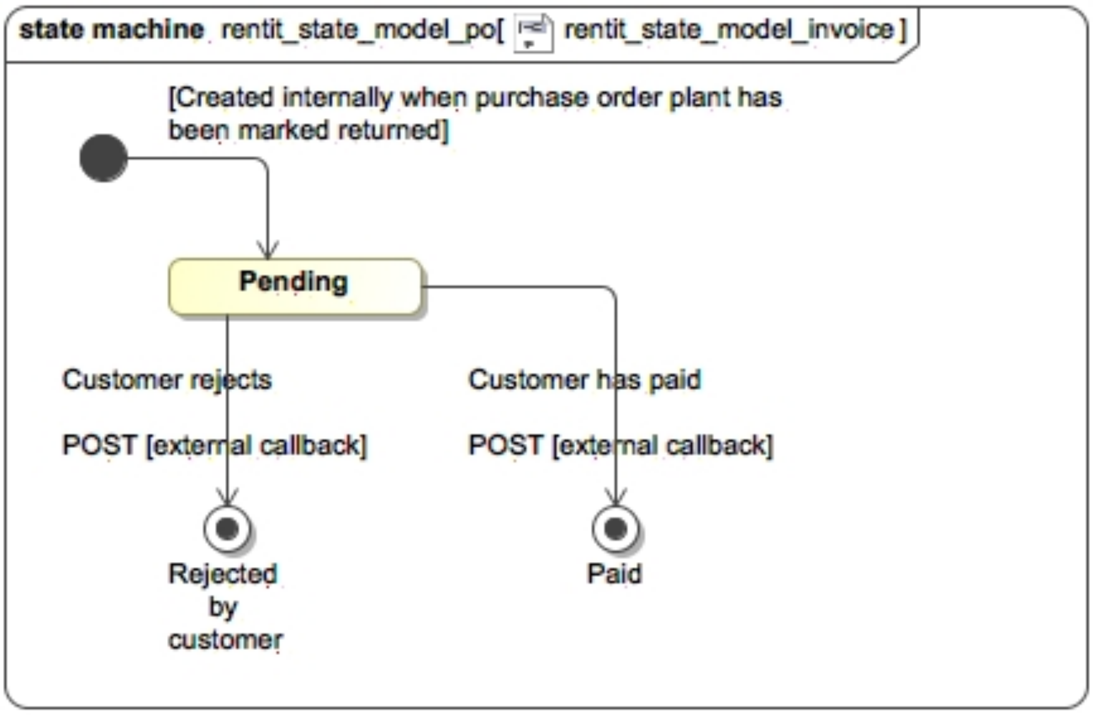

state machine rentit_state_model_po[rentit_state_model_po]
create
POST /api/sales/orders
Pending approval
reject
POST /api/sales/orders/po.id}/reject
Rejected
accept
POST /api/sales/orders/po.id}/accept
cancel
POST /api/sales/orders/po.id}/cancel
Accepted
Cancelled
Mark dispatched
POST /api/sales/orders/po.id}/dispatch
Delivery rejected by customer
POST /api/sales/orders/po.id}/customer_reject
Plant Dispatched
Rejected by customer
Invoiced
Mark delivered
POST /api/sales/orders/po.id}/deliver
Request extension [allowed only once]
POST /api/sales/orders/po.id}/extensions
[Marks that an invoice was created succesfully in invoicing sub-system after the plant was returned]
Plant delivered
Plant returned
Mark returned
POST /api/sales/orders/po.id}/return

Figure 10 State model for purchase order

State model for purchase order is depicted in Figure 10.



state machine rentit_state_model_po[rentit_state_model_invoice]
[Created internally when purchase order plant has been marked returned]
Pending
Customer rejects
POST [external callback]
Customer has paid
POST [external callback]
Rejected by customer
Paid

Figure 11 BuildIt invoice state model

State model for invoice is in Figure 11.

Apiary blueprint

The Apiary blueprint for RentIt system is located at:

https://rentit52.docs.apiary.io

RentIt API exposes the following endpoints, with their function described:
TODO

Integration

Our integration solution uses adapter pattern to provide the same operations for different partners. Every call to a partner is accompanied by partner identifier, which helps to pick the correct version of the partner service.

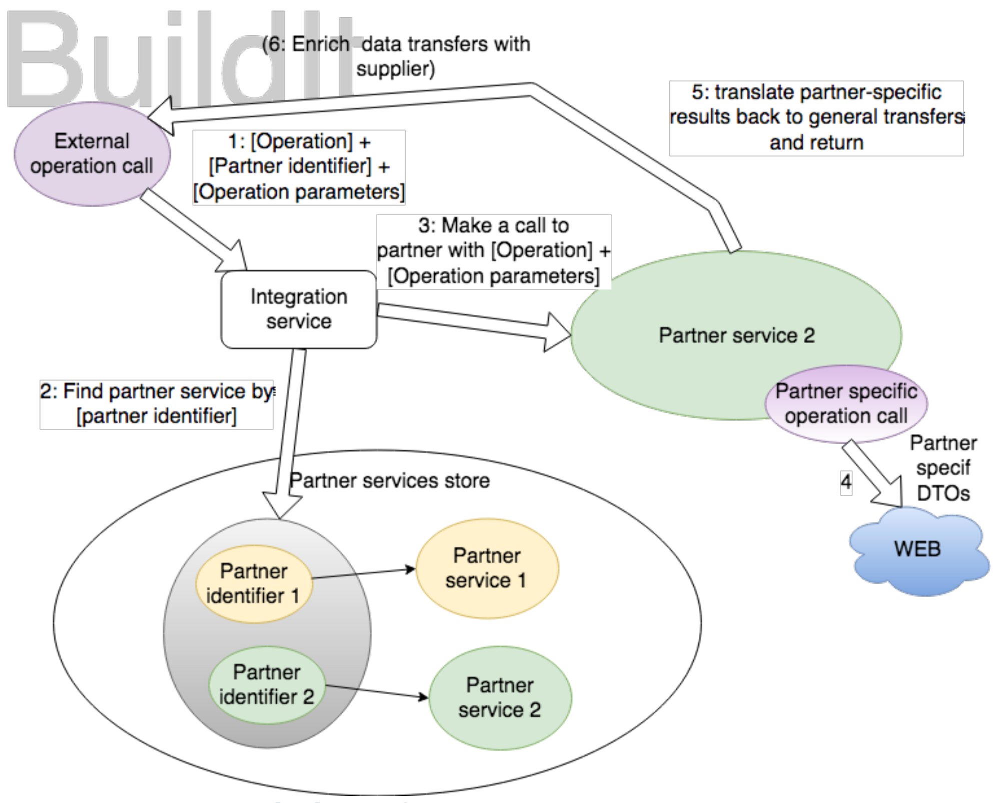

BuildIt
(6: Enrich data transfers with supplier)
5: translate partner-specific results back to general transfers and return
External operation call
1: [Operation] + [Partner identifier] + [Operation parameters]
3: Make a call to partner with [Operation] + [Operation parameters]
Integration service
Partner service 2
Partner specific operation call
2: Find partner service by [partner identifier]
4
Partner specif DTOs
WEB
Partner services store
Partner identifier 1
Partner service 1
Partner identifier 2
Partner service 2

Figure 12 Integration with adapters for partner integration

Our integration relies on push-like notifications to BuildIt from RentIt, e.g. about purchase order getting accepted by a RentIt employee. Obviously the drawback compared to polling is that integrating requires modifications in the supplier partner system.

Integration was done with “Team 2” (representative is B79605).

Implementation

The application was implemented with Spring/Vue.js stack. The final code is available at https://bitbucket.org/veskimaek/esiproject/ .

Summary

In this report, we modeled the integrated communication between enterprise systems of a construction company, and a plant equipment rental company.