Design decisions

In general, design followed Occam's razor paradigm. The application is still not ready for the real world after this homework, so some corners were cut to reduce complexity with the price of changes later.

There are endpoints and corresponding service methods that have currently no real value for the real use cases. Still they helped at least to sanity check the database state, assembling DTOs etc. during the development. Some examples are CommentController or enabling to retrieve all purchase orders.

Handling of employees was added, keeping security concerns for future. Requesting and approving engineer are therefore added with the sample data to the system and retrieving logged in user gets mocked. Later on, employee's role will be checked in order to authorize for the operations. It may also be necessary to enable several roles for the same employee row.

Total was made into Money value object, keeping open the possibility to add currency.

There are JSON objects RentItPOStatusUpdateDTO, StringDTO that serve the sole purpose to hold a value. There are also some DTOs like CreatePlantHireRequestDTO or RentItPurchaseOrderDTO that only serve to unmarshal necessary data from front-end or RentIt.

Maybe DTOs could be removed for business period and money, but currently the responsibilities of belonging to domain and transferring data were split up.

Comments are not meant to be modified, but were still kept as full domain citizens for the need to provide them with IDs and binding to a plant inventory hire request. The same is valid for construction site.

A decision was made to return data transfers already from the service layer. It could be argued that service layer should be the last layer that can edit the domain objects graph to support this decision.

A callback endpoint is provided in RentItCallbackController for RentIt to update its purchase order.

A major nuisance were hateoas links, that did not get generated correctly. This area needs a rework.

Transaction boundaries were marked at the service layer methods.

Updating a resource is implemented in PlantHireRequestService.updateRequest that gets the same data as creating one, but now optionally fields can be left empty and only the data that is provided gets updated.

Integrations with RentIt were written with keeping open the possibility that later on, connecting to other teams may cause disruptions in terms of data transfer formats, enum literals etc. So some inconsistencies between our internal versions of RentIt and BuildIt were introduced on purpose to make the design better support these external integrations from early on.