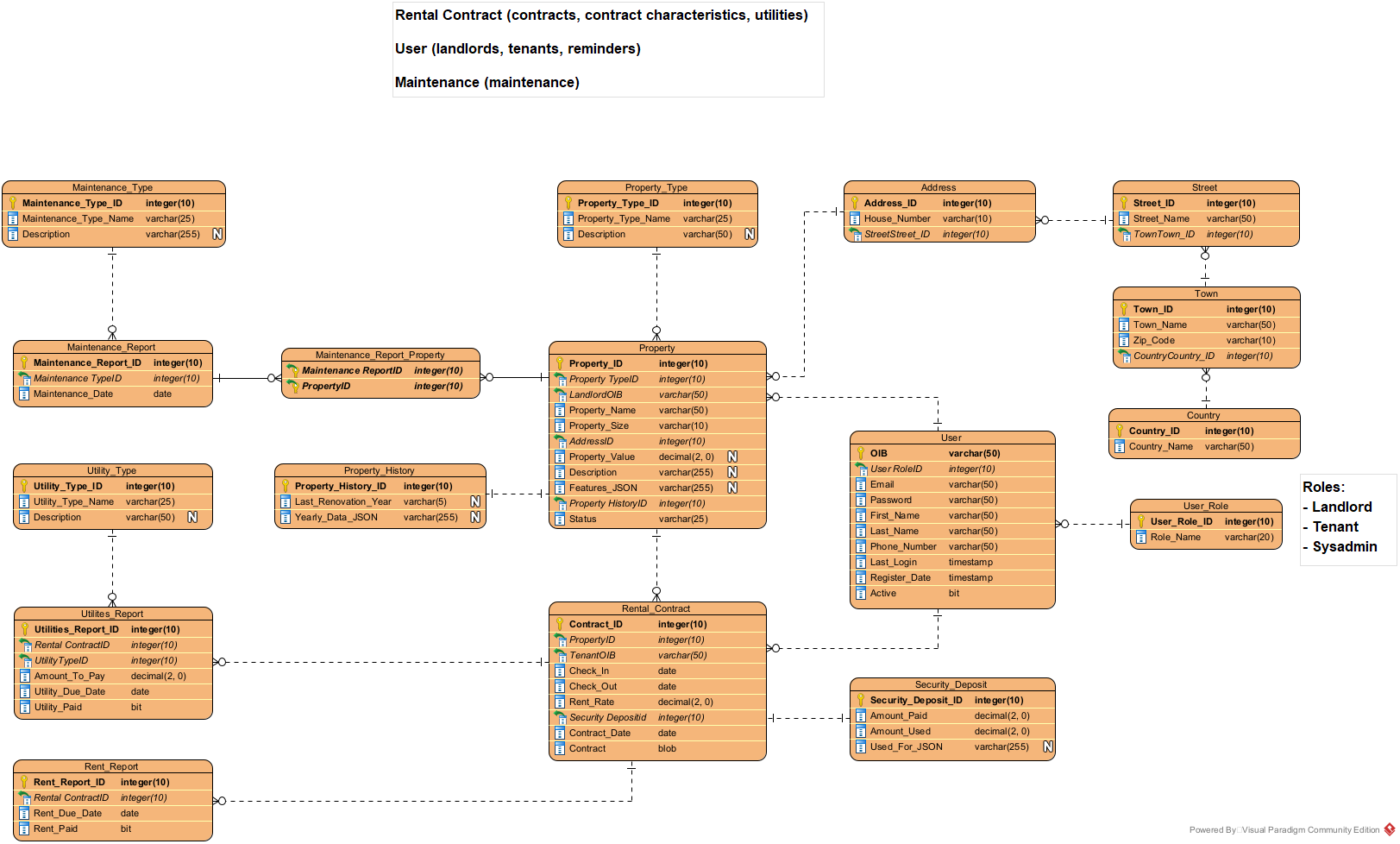
**Project topic:**

Rental management application (contracts, contract characteristics, landlords, tenants, reminders, maintenance, utilities, etc.); active + temporal + semi-structured databases (PostgreSQL with integrated data type XML or JSON and functions for working with these formats); graphical interface (optional)

**Team members:**

* Karlo Habdija (FOI)
* Lukáš Jancík (UNIZA)
* Luka Kemenović (FOI)
* Antonio Kunštek (FOI)
* Aleksa Stanković (FON)

**Phase 1: Data model creation**



Our data model has 17 tables. This being a rental management app, Property is in the middle and everything else connects to it. Property table consists of all information that is necessary for properties. Property history (rent data) is contained in property history table. We have 4 tables that contain exact address of the property (Address, Street, Town, Country). This was done so that we don’t have to put in the same address more than once when there are more than one apartments at the same address (apartment buildings). User table contains all info regarding users of our app, where the primary key is OIB because one user can have only one account. There are three user types defined in the User role table: Sysadmin, Landlord and Tenant. Rental contract table consists of everything that is important about the contract and it has one to one relationship with an entry in security deposit table which contains information about security deposit that was paid when contract was signed and what it was used for (if it was used). Final 5 tables contain data about maintenance, utilities and rents.

**Phase 2: Database creation**

Database was created following the data model defined in phase 1. Database technology that was used for this project is PostgreSQL. After we created the database, we started filling it with data. This process is described in phase 3.

**Phase 3: Filling the database with data**

At the start, we had about 6-10 records in all tables that were put in by hand just so we can test if everything works as we intended it. First step after that was generating some synthetic data to fill our database with. This was done using BizDataX software developed by Ekobit d.o.o. To check if it all works as we intended it to, we filled the database with 600-900 records per table. When we confirmed that all is in order, we generated more data to fill all our tables. In total, our database now consists of more than 1.25 million records, about 120.000 per table (some tables only need a few records, i.e. User role table). BizDataX could generate some basic data for us, but not the JSON and BLOB types so that had to be done by hand. Several different samples of JSONs were created and then inserted into all tables that have a JSON column (property history, security deposit and property features). For simplicity reasons, only one document was generated for BLOB column, but that is intended to be available to change by the landlord so it can be updated later. Insert scripts have been created for all tables. A few of functions and triggers we have implemented would be create\_security\_deposit\_and\_rental\_contract function, in which parameters we put in the rent rate, property id, tennant’s OIB, check in date, check out date, contract date and we upload the contract document, and the function then creates entries in rental contract table and security deposit table. Using this function will trigger 2 triggers: update\_property\_status\_trigger and insert\_rent\_report\_trigger, first of which sets the property’s status from ‘free’ to ‘leased’ and the latter inserts a new rent report for the following month. Also we have functions that work with semi structured data types, json to be precise. One of those we already mentioned, when creating a security deposit, a new used\_for\_JSON column is generated with amount paid and not yet used. Second one we have is the function to edit property features. This function takes property ID, feature that we’d like to update and a new value for it and then it updates the feature\_JSON column.

**Phase 4: Triggers and functions**

This project includes various SQL statements to alter sequences and create functions and triggers. The sequences are used to reset the values for various ID fields in the database, such as address\_id, country\_id, and property\_id because when filling the database with data, we use inserts that have IDs in them so it’s necessary to reset the sequences with the correct values after data insert. The functions and triggers are used to perform and automate specific actions in the application, such as creating a security deposit and rental contract, updating the status of a property when a contract is created, updating utility reports, and inserting rent reports. These functions and triggers are executed when certain events occur, such as an INSERT statement on the rental\_contract table.

**Phase 5: Application development**

For applications development, React was used for frontend development, and Node for backend. We chose this technology combination because we thought it would be the best way to show our data in order and make it look nice. Firstly we developed a simple backend that handles our requests, and then we started developing the frontend. Application has 3 roles: Sysadmin, Landlord and Tenant.

Graphical user interface, application

Description automatically generated

For now, we only had time to implement Landlord and Tenant because of a tight schedule, and we thought that those two are the most important and represent everything we have been working towards. Tenant is a bit simpler than Landlord. First feature we developed for tenant so that he can see all the free property that is available for lease and from there is available to contact the landlord.

A computer screen capture

Description automatically generated with medium confidence

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, application

Description automatically generated

He can also see all his current contracts for properties leased by him and all the rent reports and utility reports connected to that property. Here is visible all necessary information for the contract, such as check in and check out dates. Clicking on the contract in the list, a new page is opened where we can see all the utility reports, what is paid and what is not and our security deposit section where we can see how much security deposit was paid and what it was used for if it was used.

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, website

Description automatically generated

Graphical user interface, text, website

Description automatically generated

Graphical user interface, text, application

Description automatically generated

After that, we have Landlord’s application view which is a bit more complex. Landlord sees all the properties he owns divided in two sections: unleased (free/available) and leased properties. Clicking on a free property name, he can enter a new contract. After this is done a new contract is created, a new entry in security deposit is created with the amount paid in the height of a rent report. Triggers then generate entries in rent report and utilities report tables for the following month. After all that, the process is complete, property is leased and its status is updated from ‘free’ to ‘leased’.

Graphical user interface, text, application, email

Description automatically generated

When the leased property is clicked, user will be pushed to new page where he can see all important data about utility like history data, maintenace data and so on.

Also landlord can change feature for leased property. Here are listed features from list and he can applied some sort of value for feature he would like to update.

History can be downloaded as a json document.

Graphical user interface, text, application, email

Description automatically generated

Final thing we can see on the Landlord are two screens where all rent reports for all properties and all utility reports for all properties are visible. Rent reports are generated monthly, and utility reports will be entered by Landlord but sadly this feature is still not complete on the frontend.

Graphical user interface

Description automatically generated

Graphical user interface

Description automatically generated