Università della Svizzera italiana

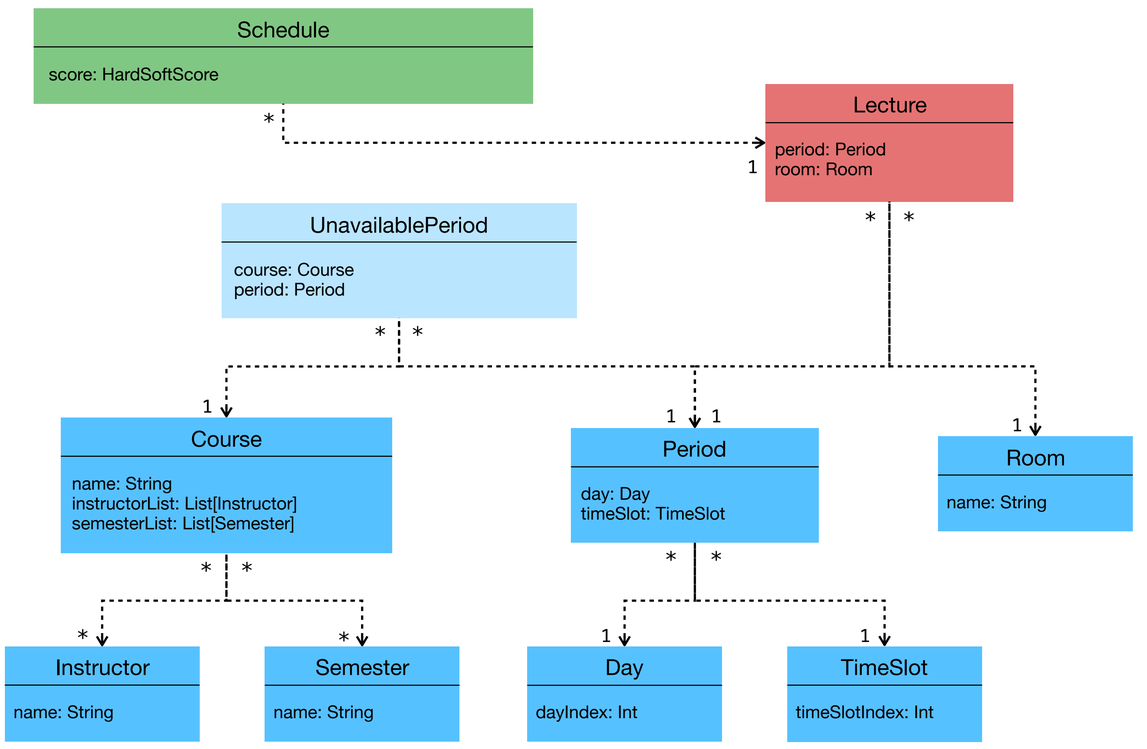
Faculty of Informatics

University timetable scheduler

Student: Aron Fiechter **Advisor:** Prof. Dr. Michele Lanza **Assistants:** Dr. A. Mocci, Dr. L. Ponzanelli

Background

Creating and managing timetables of courses is an issue for many institutions because of the various constraints that need to be respected in the planning.
This is an even harder problem if the data needed to describe the planning problem and its constraints is not well curated and stored in several different places, a situation that often leads to a long and tedious manual work in creating the timetables.
In this project, we automate the process by means of state of the art tools to solve planning problems. The chosen tool is OptaPlanner, a constraint solver that requires to model the domain of the problem by means of a **Solution** class, for which a score needs to be computed. The **Solution** has a **PlanningEntity** which has variables that can be changed to improve the score. Everything else is constant for each problem instance and is referred to as a **ProblemFact**.

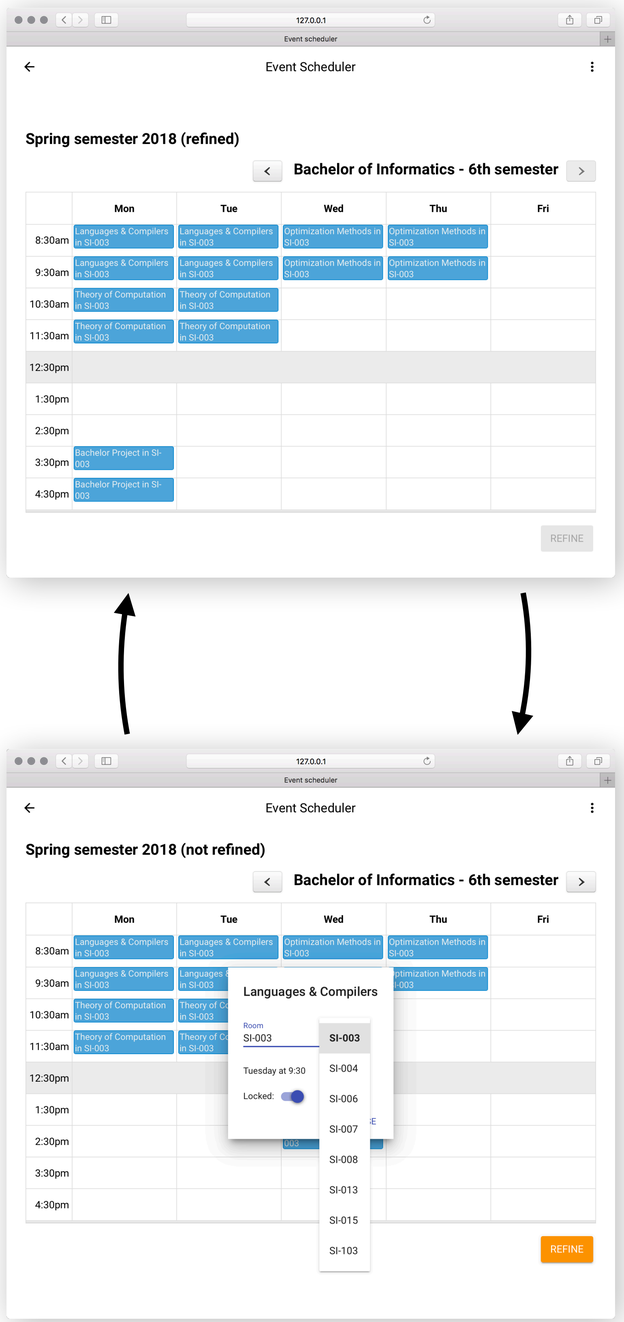


Constraints

The constraints include room capacity, room occupancy, the fact that an instructor cannot teach two courses at the same time, the conflicts between lectures of the same semester.
On the back end, all these constraints are given to the solver by specifying them using the language below.

```
createSchedule {
  courses {
    name
    instructors
    semesters
    studentSize
    numberOfLectures
    minDifferentDays
  }
  rooms {
    name capacity
  }
  unavailablePeriods {
    course period
  }
```

```
// Room capacity: A room's capacity should not be
// less than the number of students in the course.
// Each student above the capacity counts as 1 point
// of penalty.
rule "roomCapacity"
  when
    $room : Room($capacity : capacity)
    Lecture(
       room == $room,
       course.studentSize > $capacity,
       $studentSize : course.studentSize
    )
  then
    scoreHolder.addHardConstraintMatch(
       kcontext,
       ($capacity - $studentSize)
    );
end
```



Initial solution

The input data is converted to the format needed by the solver, then the solver runs for some seconds and finally returns the computed schedule.

The schedule is shown as a weekly view separated by semester. Each lecture shows the course name and the room in which it takes place.

Editing and refinement

Using the web application, once a schedule is opened, it is possible to move lectures to different periods and different rooms, and to lock lectures in place to prevent the solver from moving them if the schedule is sent to the backend for refinement.

In that case, the solver keeps existing locked lectures in place and builds a new improved solution around them.

Future work

The web application allows to create a schedule for a predefined set of rooms and courses, with predefined unavailable periods. In the future it would be better to be able to select specific courses, rooms and other constraints from the front end application.

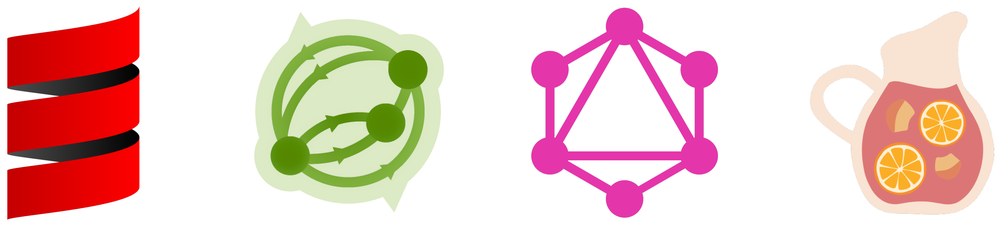
Other possible extensions include allowing to accept a final schedule and applying it to a specific time period, and then visualise real events on a calendar which offers various aggregated views, such as events by room, by instructor or by day.

This calendar would then allow to add single events such as seminars, or conferences.

Technologies

Front end

Back end



codelounge

REVEAL

USI