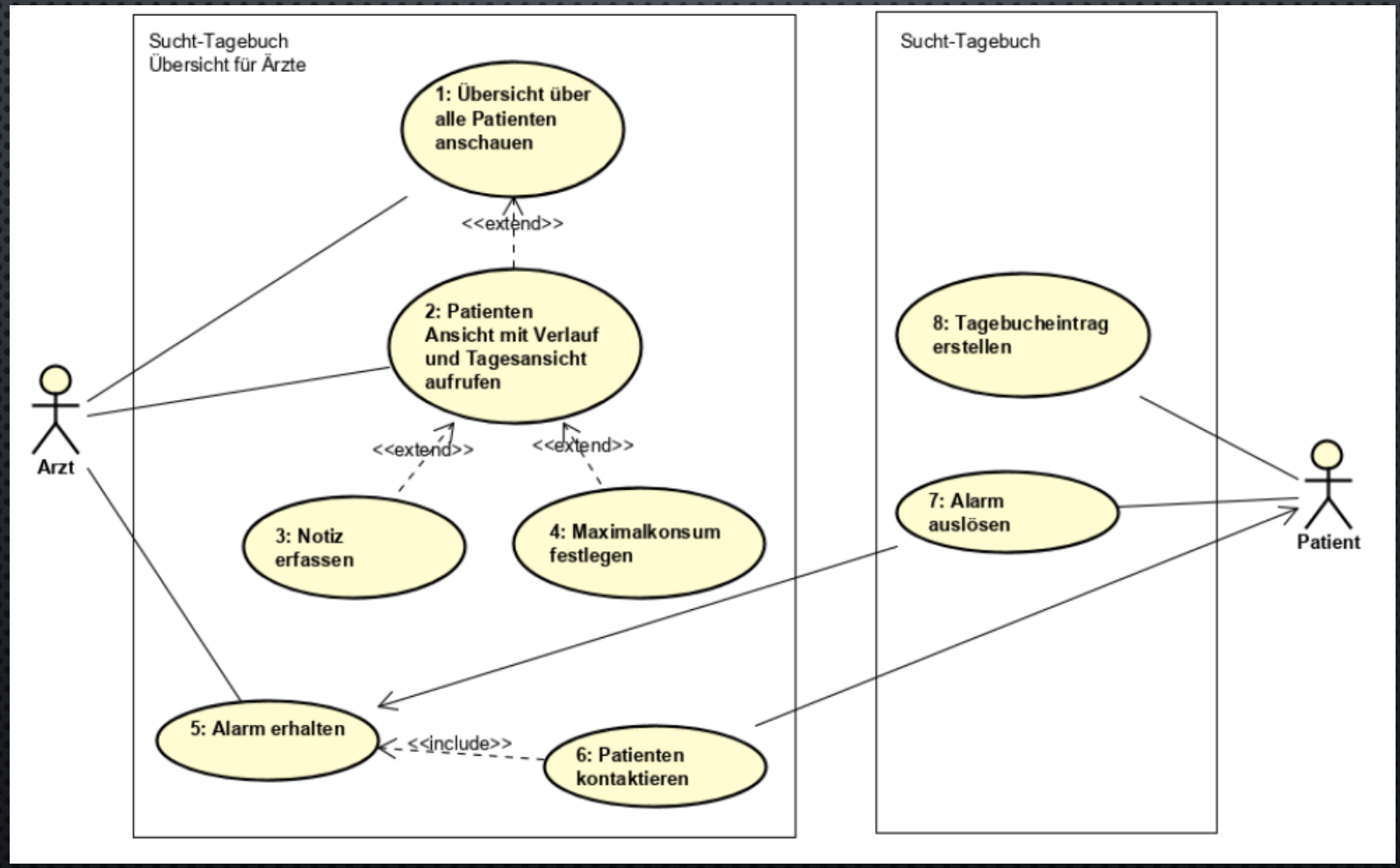
FINAL PRESENTATION

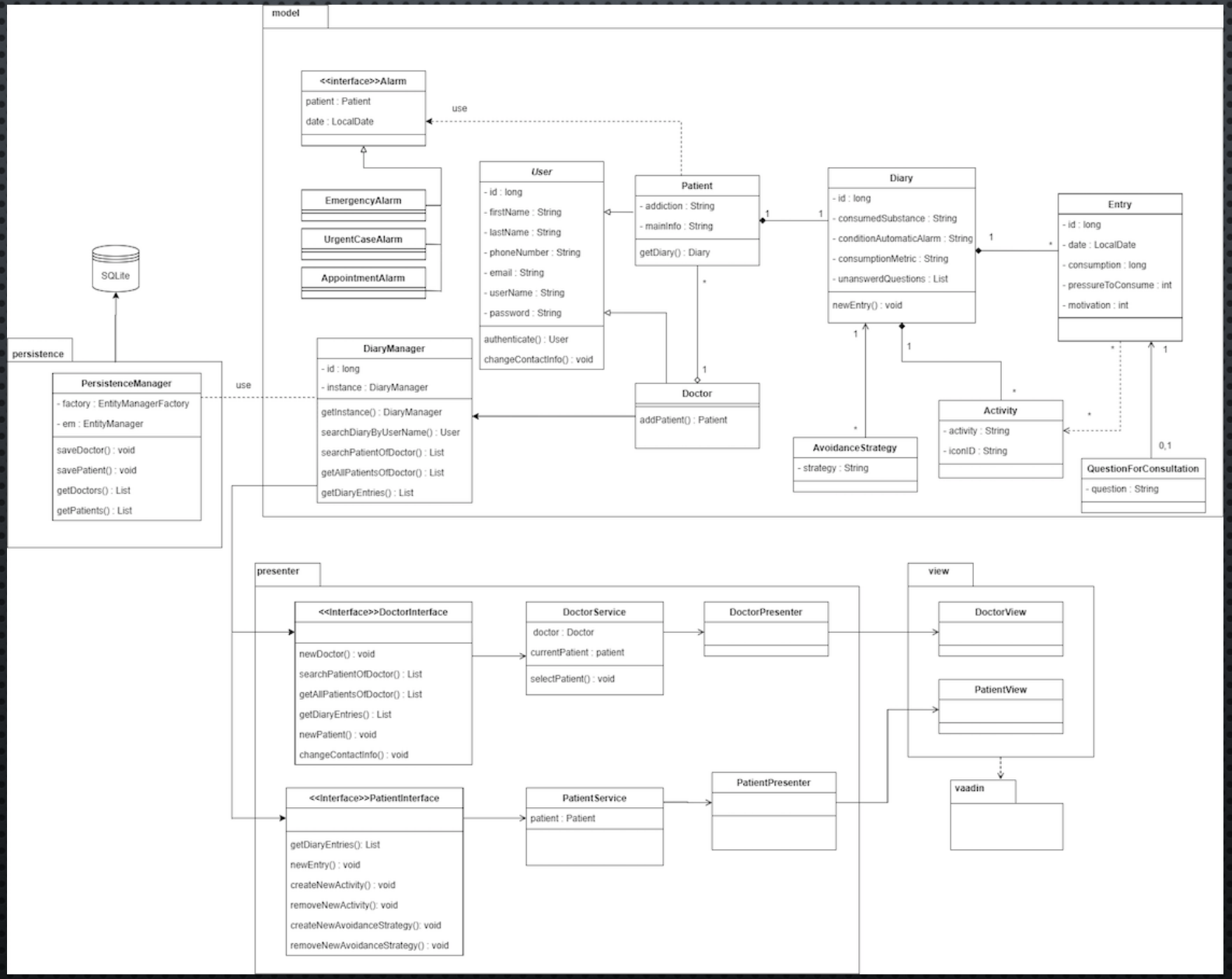
Team red

Dénervaud Natalya, Rodriguez Julián, Meyer Remo, Pelts Dmytriy,
Kirupananthan Kaurisanker

FEATURES IMPLEMENTED VS. DESIGN THINKING IDEAS



ARCHITECTURE



CODE HIGHLIGHT

```java
public class Condition {

    private Metric metric;
    private Operand operand;
    private int thresholdValue;
    private int thresholdDays;
    private int amountOfDays;

    NextCondition nextCondition = NextCondition.NONE;

    Condition condition;

    public Condition(Metric metric, Operand operand, int thresholdValue, int thresholdDays, int amountOfDays) {
        this.metric = metric;
        this.operand = operand;
        this.thresholdValue = thresholdValue;
        this.thresholdDays = thresholdDays;
        this.amountOfDays = amountOfDays;
    }

    protected Condition decorateAnd(Condition condition) {
        this.nextCondition = NextCondition.AND;
        this.condition = condition;
        return this.condition;
    }

    protected Condition decorateOr(Condition condition) {
        this.nextCondition = NextCondition.OR;
        this.condition = condition;
        return this.condition;
    }

    protected boolean isGiven(List<Entry> entries) {
        Collections.sort(entries);
        switch (this.isGiven(ConditionState.UNKNOWN, entries)) {
        case FALSE:
            return false;
        case TRUE:
            return true;
        case UNKNOWN:
            return false;
        default:
            return false;
        }
```

```java
protected ConditionState isGiven(ConditionState state, List<Entry> entries) {
    switch (state) {
    case FALSE:
        switch (this.nextCondition) {
        case AND:
            return this.condition.isGiven(ConditionState.FALSE, entries);
        case NONE:
            return ConditionState.FALSE;
        case OR:
            return this.condition.isGiven(ConditionState.UNKNOWN, entries);
        }
    case TRUE:
        return ConditionState.TRUE;
    case UNKNOWN:
        switch (this.nextCondition) {
        case AND:
            if (this.check(entries)) {
                return this.condition.isGiven(ConditionState.UNKNOWN, entries);
            } else {
                return this.condition.isGiven(ConditionState.FALSE, entries);
            }
        case NONE:
            if (this.check(entries)) {
                return ConditionState.TRUE;
            } else {
                return ConditionState.FALSE;
            }
        case OR:
            if (this.check(entries)) {
                return this.condition.isGiven(ConditionState.TRUE, entries);
            } else {
                return this.condition.isGiven(ConditionState.UNKNOWN, entries);
            }
        }
    }
    return state;
}
```

CODE HIGHLIGHT

```
private boolean check(List<Entry> entries) {
    int res = 0;
    switch (this.operand) {
    case ISEQUAL:
        for (Integer i : getValues(entries)) {
            if (i == this.thresholdValue) {
                res++;
            }
        }
        break;
    case ISEQUALORLOWER:
        for (Integer i : getValues(entries)) {
            if (i <= this.thresholdValue) {
                res++;
            }
        }
        break;
    case ISHIGHER:
        for (Integer i : getValues(entries)) {
            if (i > this.thresholdValue) {
                res++;
            }
        }
        break;
    case ISHIGHEROREQUAL:
        for (Integer i : getValues(entries)) {
            if (i >= this.thresholdValue) {
                res++;
            }
        }
        break;
    case ISLOWER:
        for (Integer i : getValues(entries)) {
            if (i < this.thresholdValue) {
                res++;
            }
        }
        break;
    default:
        break;
    }
    if (res >= this.thresholdDays) {
        return true;
    } else {
        return false;
    }
}
```

```
private List<Integer> getValues(List<Entry> entries) {
    List<Integer> res = new ArrayList<Integer>();
    switch (this.metric) {
    case CONSUMPTION:
        for (int i = 0; i < this.amountOfDays; i++) {
            res.add(Math.toIntExact(entries.get(i).getConsumption()));
        }
        break;
    case MOTIVATION:
        for (int i = 0; i < this.amountOfDays; i++) {
            res.add(entries.get(i).getMotivation());
        }
        break;
    case PRESSURE:
        for (int i = 0; i < this.amountOfDays; i++) {
            res.add(entries.get(i).getPressureToConsume());
        }
        break;
    default:
        return null;
    }
    return res;
}
```

Lessons Learnt

Pro:

- Vaadin als Frontendtechnologie kennengelernt
- Scrum Methode kennengelernt
- Detaillierte Diskussion der Architektur
- Nach MVP Pattern implementiert
- Arbeitserfahrung als Gruppe
- Codereviews

Con:

- Vaadin documentation
- DB-Anbindung

DEMO


Edit patient data
All entries
Activities
Open questions
Strategies
Patient overview
Log out
First name:
Natalya
Last name:
Dénervaud
Addiction:
Alcohol
Info:
Had control over her addiction for 3 years. Relapse on january 2020
Edit Info
Show all entries
Activities
Open questions
Strategies
Back to patient list
Edit automatic alarm condition
Log out
Statistic of patient's addiction
Consumption
Max Level Consumption
Motivation
7.5 mm
6 mm
4.5 mm
3 mm
1.5 mm
0 mm
Consumption
Motivation
Max Level Consumption
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30