BurnOUTEvents

Finale Präsentation

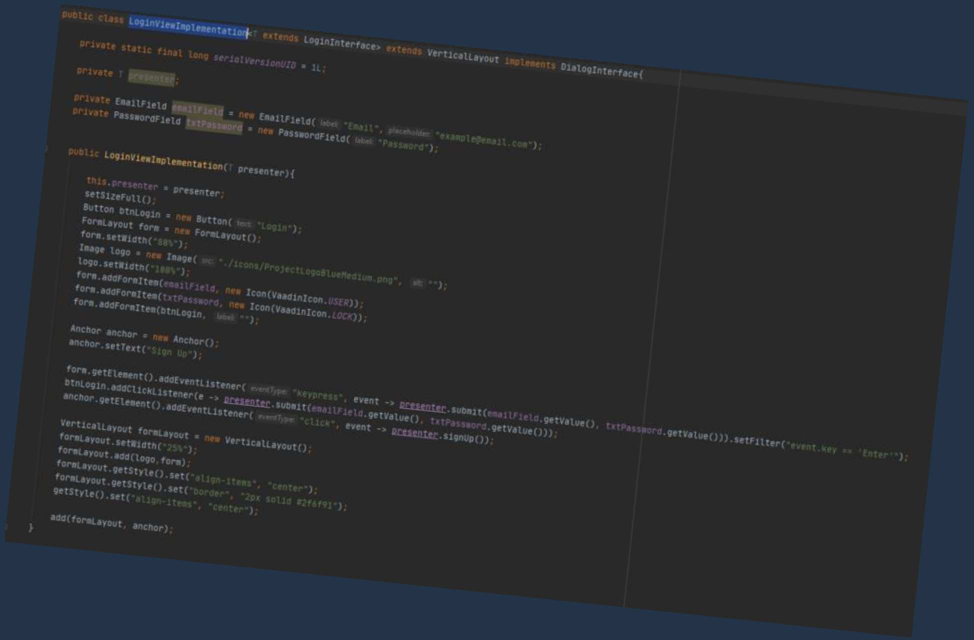
Team Black

DEMO

BurnOUTEvents

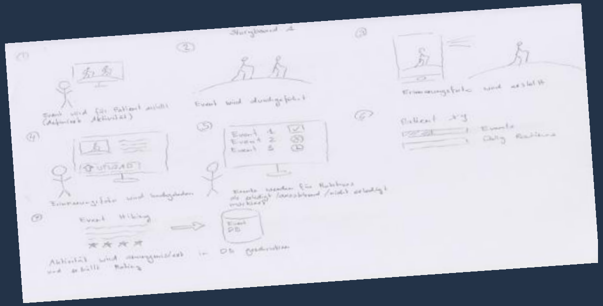
Design Thinking VS Features Implemented

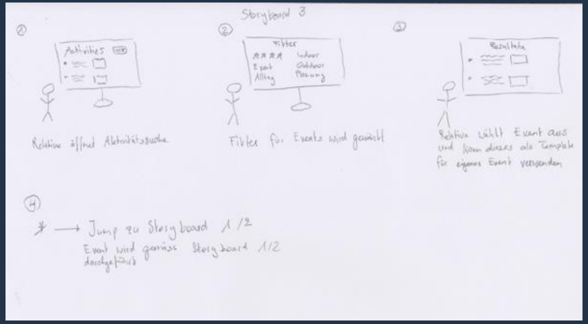
VS

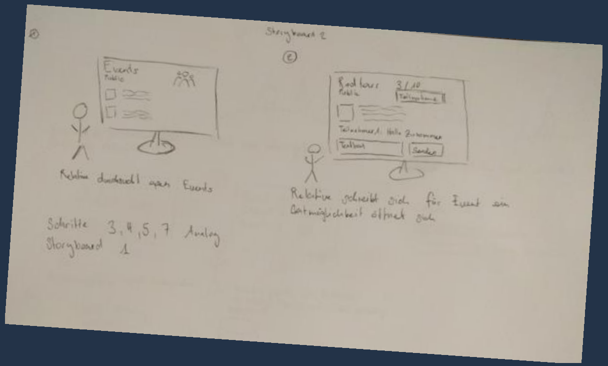


```java
public class LoginViewImplementation<T extends LoginInterface> extends VerticalLayout implements DialogInterface{
    private static final long serialVersionUID = 1L;

    private T presenter;

    private EmailField emailField = new EmailField( label: "Email", placeholder: "example@email.com");
    private PasswordField txtPassword = new PasswordField( label: "Password");

    public LoginViewImplementation(T presenter){
        this.presenter = presenter;
        setSizeFull();
        Button btnLogin = new Button( text: "Login");
        FormLayout form = new FormLayout();
        form.setWidth("80%");
        Image logo = new Image( src: "./icons/ProjectLogoBlueMedium.png", alt: "");
        logo.setWidth("100%");
        form.addFormItem(emailField, new Icon(VaadinIcon.USER));
        form.addFormItem(txtPassword, new Icon(VaadinIcon.LOCK));
        form.addFormItem(btnLogin, label: "");

        Anchor anchor = new Anchor();
        anchor.setText("Sign Up");

        form.getElement().addEventListener( eventType: "keypress", event -> presenter.submit(emailField.getValue(), txtPassword.getValue())).setFilter("event.key == 'Enter'");
        btnLogin.addClickListener(e -> presenter.submit(emailField.getValue(), txtPassword.getValue()));
        anchor.getElement().addEventListener( eventType: "click", event -> presenter.signUp());

        VerticalLayout formLayout = new VerticalLayout();
        formLayout.setWidth("25%");
        formLayout.add(logo,form);
        formLayout.getStyle().set("align-items", "center");
        formLayout.getStyle().set("border", "2px solid #2f6f91");
        getStyle().set("align-items", "center");

        add(formLayout, anchor);
    }
```

Storyboards


Storyboard 1
Event wird für Patient erstellt (Aufwand Aktivität)
Event wird durchgeführt
Erinnerungsfoto wird erstellt
Erinnerungsfoto wird hochgeladen
Event 1
Event 2
Event 3
Events werden für Relatives als erledigt / ausstehend / nicht erledigt markiert
Patient
Events
Daily Routine
UPLOAD
Event Hiking
Event DB
Aktivität wird anonymisiert und erhält Rating
in DB gespeichert

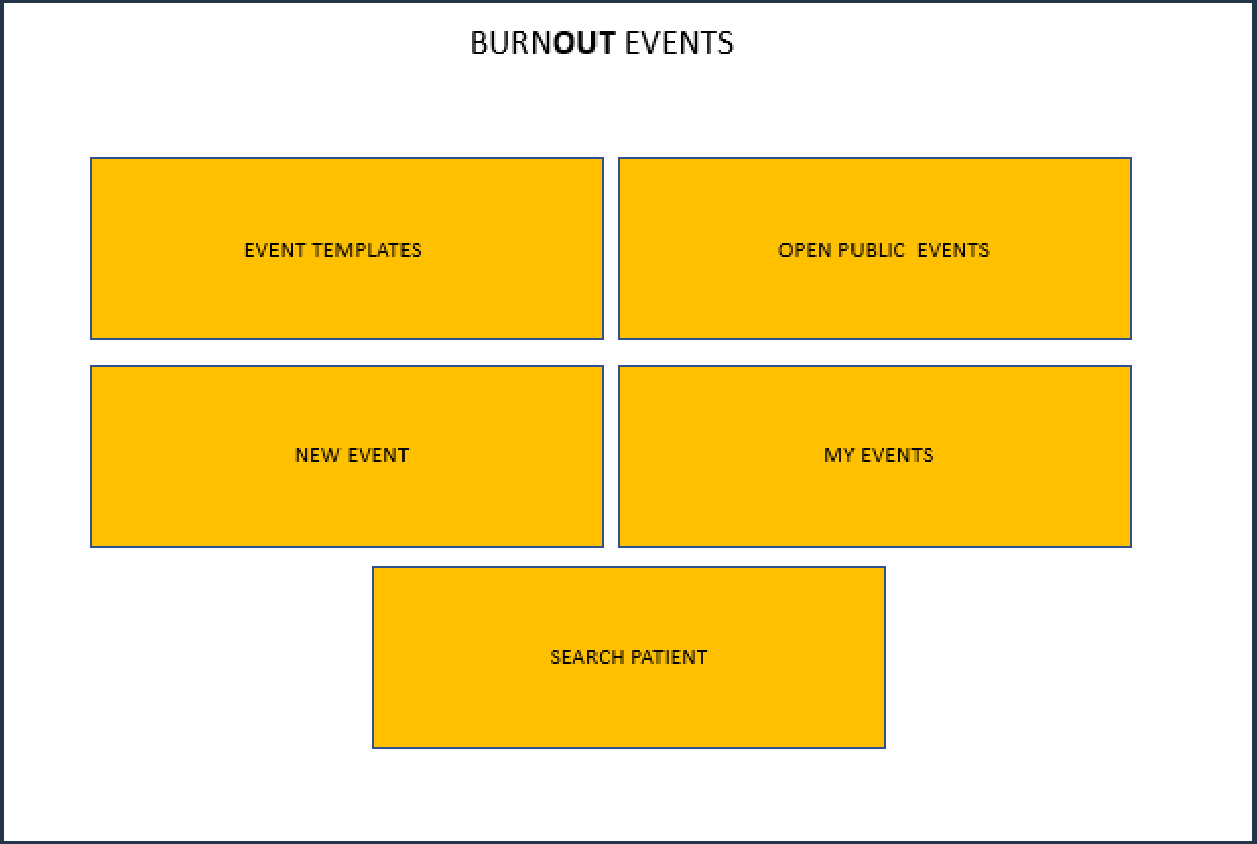

Storyboard 3
Activities
Filter
Event
Alltag
Indoor
Outdoor
Planung
Resultate
Relative öffnet Aktivitätssuche
Filter für Events wird gewählt
Relative wählt Event aus und kann dieses als Template für eigenes Event verwenden
Jump zu Storyboard 1/2
Event wird gemäss Storyboard 1/2 durchgeführt


Storyboard 2
Events
Public
Relative durchsucht open Events
Schritte 3, 4, 5, 7 Analog Storyboard 1
Radtour 3/10
Public
Teilnehmen
Teilnehmer 1: Hallo Zusammen
Textbox
Senden
Relative schreibt sich für Event ein
Chatmöglichkeit öffnet sich

- Öffentliche/private Events ✓
- Erinnerungsfoto ✓
- Events als Templates ✓
- Filter Templates ✓
- Chat/Forum je Event ✓
- Tags hinzufügen ✗

BurnOUTEvents

Prototyp



BURNOUT EVENTS

EVENT TEMPLATES

OPEN PUBLIC EVENTS

NEW EVENT

MY EVENTS

SEARCH PATIENT

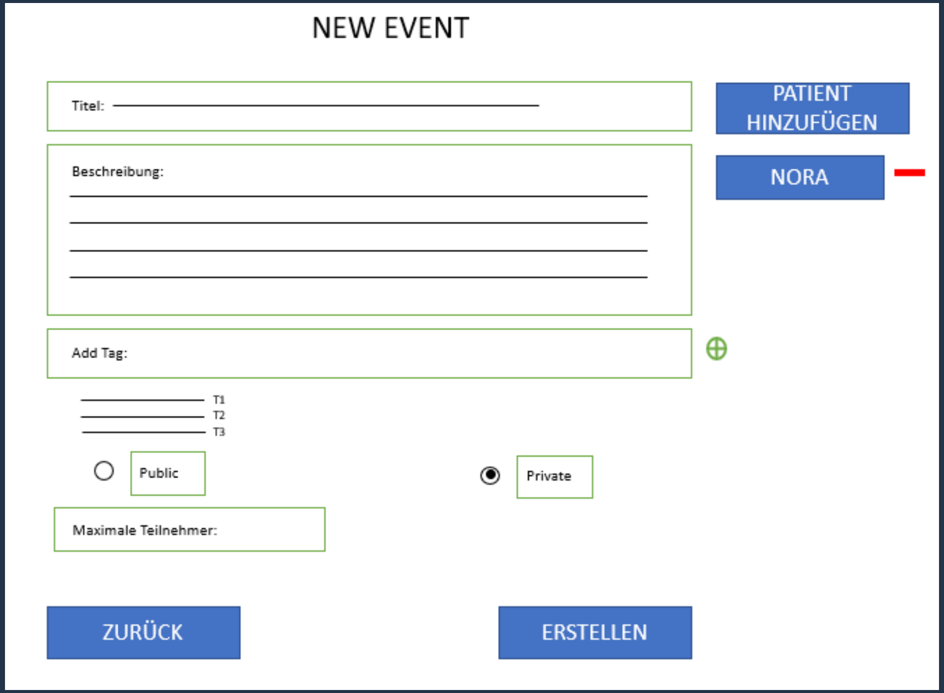
App

CREATE EVENT

JOIN PUBLIC EVENT

MY EVENTS

BurnOUTEvents



NEW EVENT

Titel: ____________

PATIENT HINZUFÜGEN

Beschreibung: ____________

NORA

Add Tag:

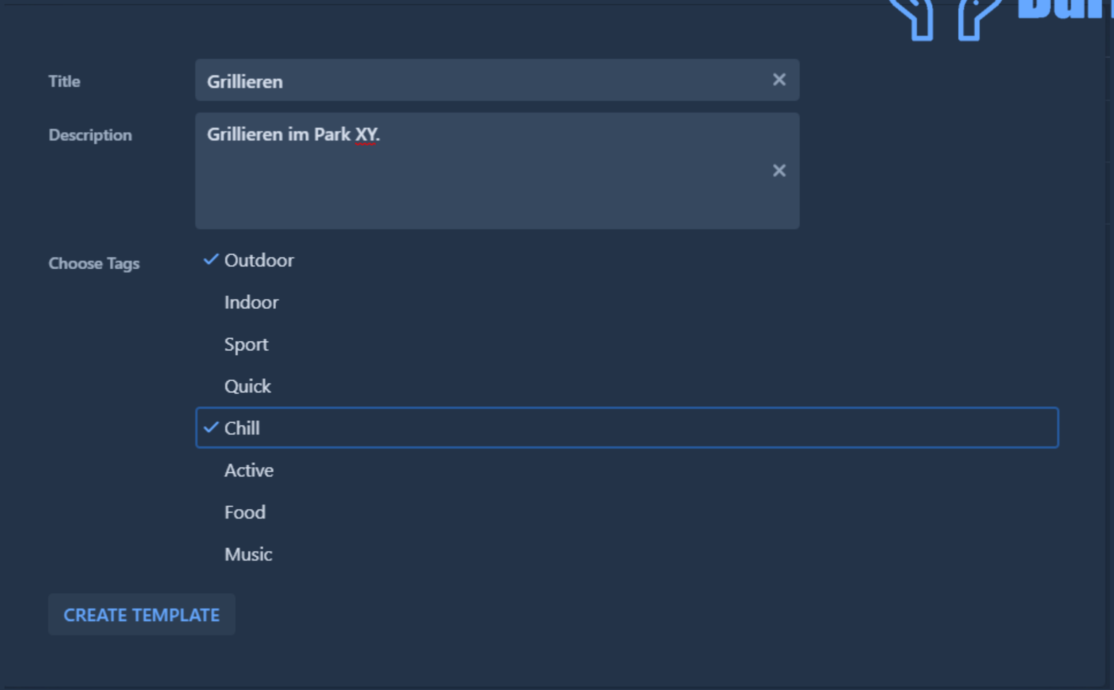
T1
T2
T3

Public [✓ Private]

Maximale Teilnehmer:

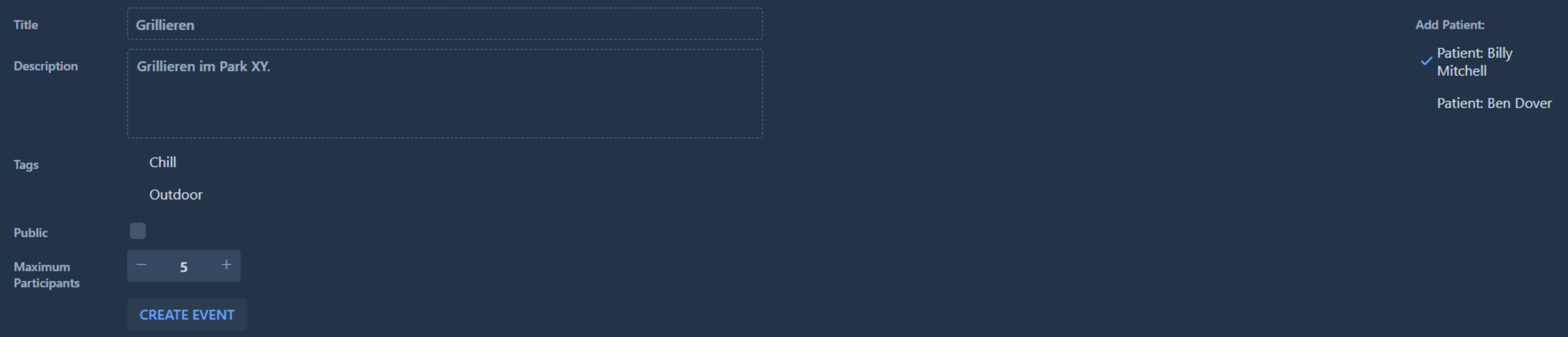
ZURÜCK

ERSTELLEN



Title	Grillieren
Description	Grillieren im Park XY.
Choose Tags	✓ Outdoor
	Indoor
	Sport
	Quick
	✓ Chill
	Active
	Food
	Music

CREATE TEMPLATE



Title	Grillieren
Description	Grillieren im Park XY.
Tags	Chill
	Outdoor
Public	☐
Maximum Participants	5

CREATE EVENT

Add Patient:

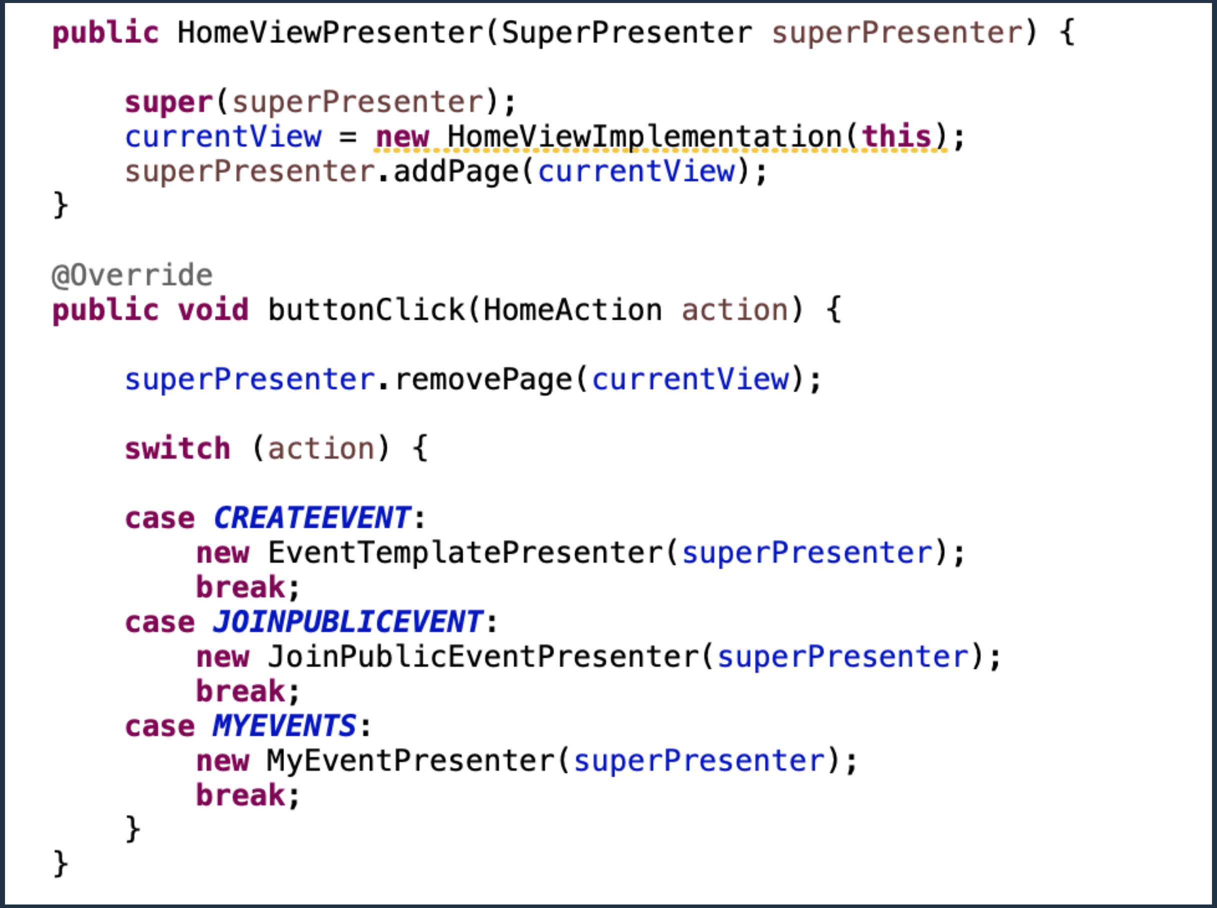
✓ Patient: Billy Mitchell

Patient: Ben Dover

BurnOUTEvents

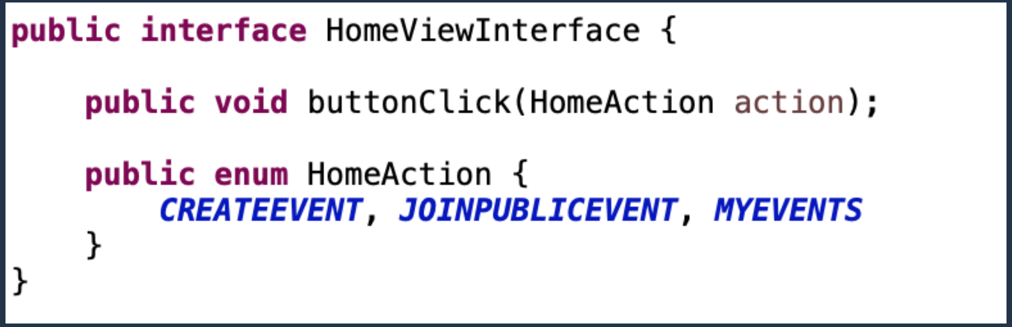
```java
private MainView mainView;
private VerticalLayout header;
private VerticalLayout page;

private Account account;
private StateModel stateModel;

public SuperPresenter(MainView mainView) {

    this.stateModel = null;
    this.mainView = mainView;

    header = new VerticalLayout();
    page = new VerticalLayout();

    mainView.add(header, page);

    if (account == null) {
        new LoginPresenter(this);
    } else {
        new HeaderPresenter(this);
        new HomeViewPresenter(this);
    }
}

public void addHeader(Component component) {
    header.add(component);
}

public void removeHeader(Component currentView) {
    header.remove(currentView);
}

public void addPage(Component component) {
    page.add(component);
}

public void removePage(Component currentView) {
    page.remove(currentView);
}
```

BurnOUTEvents

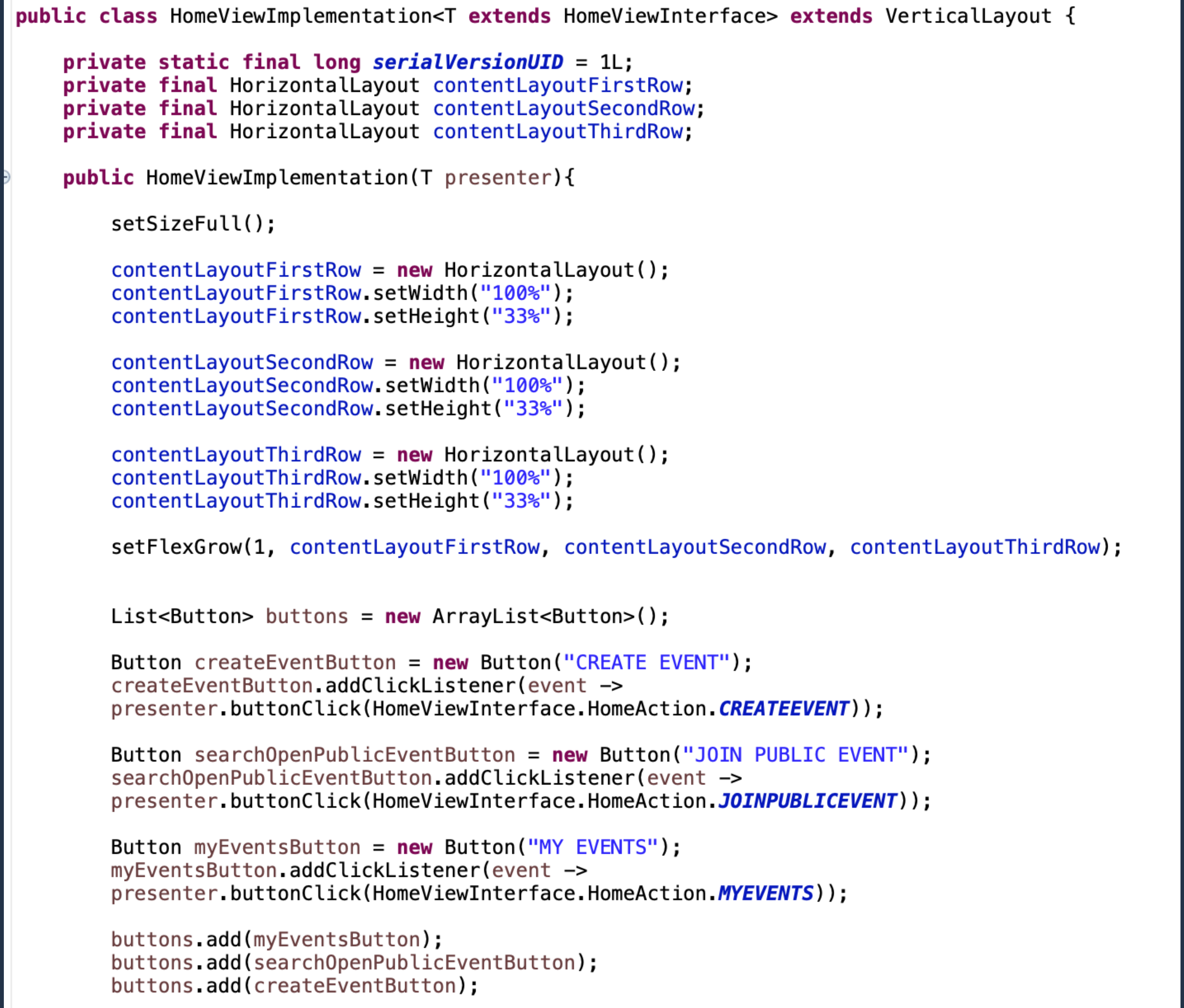


```java
public HomeViewPresenter(SuperPresenter superPresenter) {

    super(superPresenter);
    currentView = new HomeViewImplementation(this);
    superPresenter.addPage(currentView);
}

@Override
public void buttonClick(HomeAction action) {

    superPresenter.removePage(currentView);

    switch (action) {

    case CREATEEVENT:
        new EventTemplatePresenter(superPresenter);
        break;
    case JOINPUBLICEVENT:
        new JoinPublicEventPresenter(superPresenter);
        break;
    case MYEVENTS:
        new MyEventPresenter(superPresenter);
        break;
    }
}
```

BurnOUTEvents



```java
public interface HomeViewInterface {

    public void buttonClick(HomeAction action);

    public enum HomeAction {
        CREATEEVENT, JOINPUBLICEVENT, MYEVENTS
    }
}
```



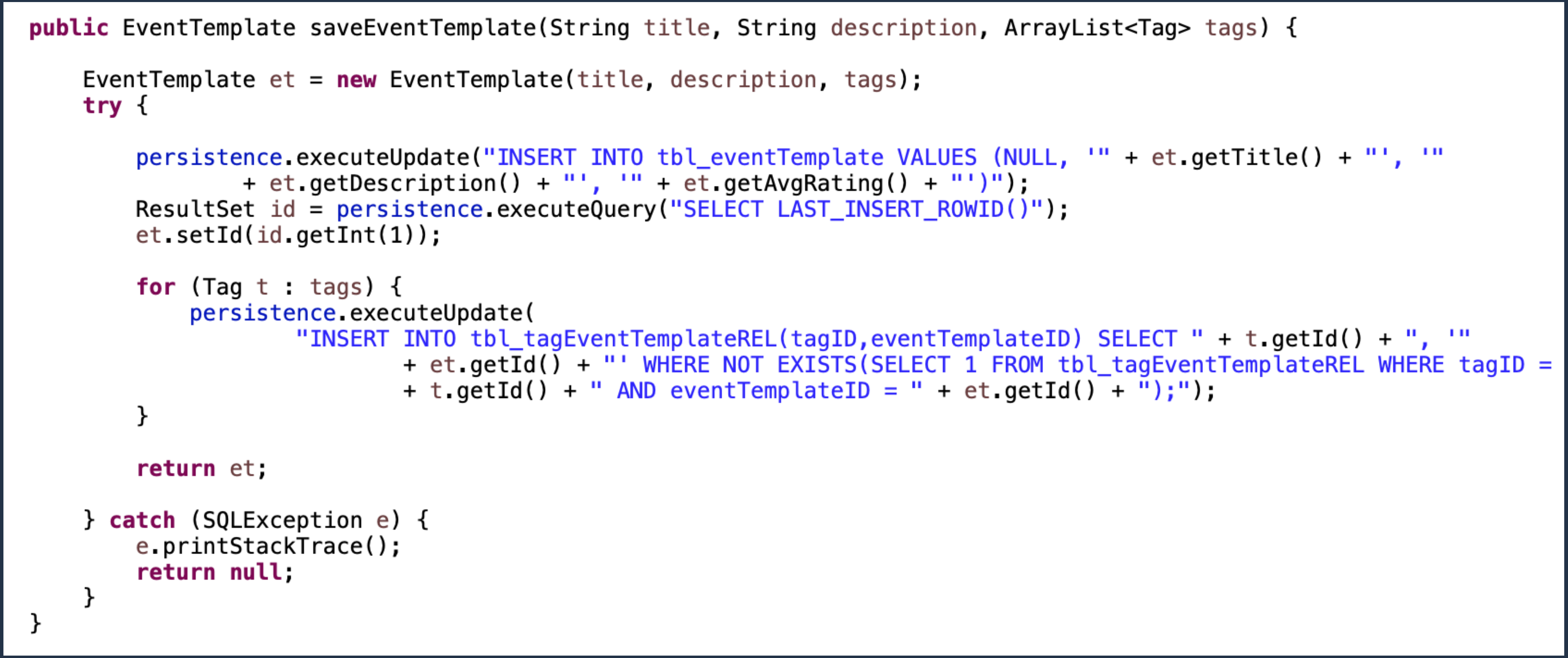
```java
public class HomeViewImplementation<T extends HomeViewInterface> extends VerticalLayout {

    private static final long serialVersionUID = 1L;
    private final HorizontalLayout contentLayoutFirstRow;
    private final HorizontalLayout contentLayoutSecondRow;
    private final HorizontalLayout contentLayoutThirdRow;

    public HomeViewImplementation(T presenter){

        setSizeFull();

        contentLayoutFirstRow = new HorizontalLayout();
        contentLayoutFirstRow.setWidth("100%");
        contentLayoutFirstRow.setHeight("33%");

        contentLayoutSecondRow = new HorizontalLayout();
        contentLayoutSecondRow.setWidth("100%");
        contentLayoutSecondRow.setHeight("33%");

        contentLayoutThirdRow = new HorizontalLayout();
        contentLayoutThirdRow.setWidth("100%");
        contentLayoutThirdRow.setHeight("33%");

        setFlexGrow(1, contentLayoutFirstRow, contentLayoutSecondRow, contentLayoutThirdRow);


        List<Button> buttons = new ArrayList<Button>();

        Button createEventButton = new Button("CREATE EVENT");
        createEventButton.addClickListener(event ->
        presenter.buttonClick(HomeViewInterface.HomeAction.CREATEEVENT));

        Button searchOpenPublicEventButton = new Button("JOIN PUBLIC EVENT");
        searchOpenPublicEventButton.addClickListener(event ->
        presenter.buttonClick(HomeViewInterface.HomeAction.JOINPUBLICEVENT));

        Button myEventsButton = new Button("MY EVENTS");
        myEventsButton.addClickListener(event ->
        presenter.buttonClick(HomeViewInterface.HomeAction.MYEVENTS));

        buttons.add(myEventsButton);
        buttons.add(searchOpenPublicEventButton);
        buttons.add(createEventButton);
```

BurnOUTEvents

BurnOUTEvents

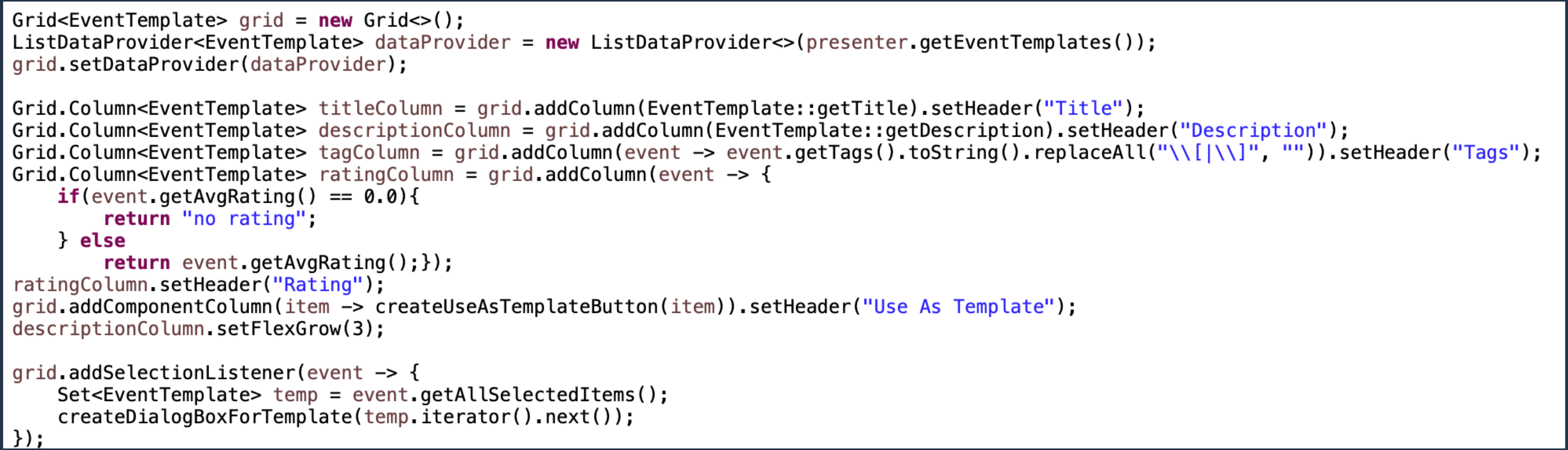
```java
public class CloseEvent extends StateModel {

    // Save image to DB
    public void savePicture(byte[] picture, Event event) throws SerialException, SQLException {

        String sql = "INSERT INTO tbl_image (eventID, image) VALUES (?,?)";

        PreparedStatement stm = persistence.getPreparedStatement(sql);

        stm.setInt(1, event.getId());
        stm.setBytes(2, picture);

        stm.executeUpdate();
        stm.close();
    }

    // Set rating of the done event
    public void setEventRating(Event event) {
        persistence.executeUpdate(
                "UPDATE tbl_event SET rating = " + event.getRating() + " WHERE eventID = " + event.getId());
    }

    // Set status for the done event
    public void setEventStatus(Event event) {
        persistence.executeUpdate(
                "UPDATE tbl_event SET state = '" + event.getStatus() + "' WHERE eventID = " + event.getId());

    }

    // Calculate the average rating of an event template
    public void updateAvgRating(EventTemplate eventTemplate) {
        ArrayList<Event> events = getEventListByTemplate(eventTemplate);

        double count = 0;
        double sum = 0;

        for (Event e : events) {
            if (e.getRating() != 0) {
                sum += e.getRating();
                count++;
            }
        }
        double avgRating = sum/count;

        persistence.executeUpdate(
                "UPDATE tbl_eventTemplate SET rating = '" + avgRating  + "' WHERE eventTemplateID = " + eventTemplate.getId());
```

BurnOUTEvents



```java
public EventTemplate saveEventTemplate(String title, String description, ArrayList<Tag> tags) {

    EventTemplate et = new EventTemplate(title, description, tags);
    try {

        persistence.executeUpdate("INSERT INTO tbl_eventTemplate VALUES (NULL, '" + et.getTitle() + "', '"
                + et.getDescription() + "', '" + et.getAvgRating() + "')");
        ResultSet id = persistence.executeQuery("SELECT LAST_INSERT_ROWID()");
        et.setId(id.getInt(1));

        for (Tag t : tags) {
            persistence.executeUpdate(
                    "INSERT INTO tbl_tagEventTemplateREL(tagID,eventTemplateID) SELECT " + t.getId() + ", '"
                            + et.getId() + "' WHERE NOT EXISTS(SELECT 1 FROM tbl_tagEventTemplateREL WHERE tagID =
                            + t.getId() + " AND eventTemplateID = " + et.getId() + ");");
        }

        return et;

    } catch (SQLException e) {
        e.printStackTrace();
        return null;
    }
}
```

BurnOUTEvents



```java
Grid<EventTemplate> grid = new Grid<>();
ListDataProvider<EventTemplate> dataProvider = new ListDataProvider<>(presenter.getEventTemplates());
grid.setDataProvider(dataProvider);

Grid.Column<EventTemplate> titleColumn = grid.addColumn(EventTemplate::getTitle).setHeader("Title");
Grid.Column<EventTemplate> descriptionColumn = grid.addColumn(EventTemplate::getDescription).setHeader("Description");
Grid.Column<EventTemplate> tagColumn = grid.addColumn(event -> event.getTags().toString().replaceAll("\\[|\\]", "")).setHeader("Tags");
Grid.Column<EventTemplate> ratingColumn = grid.addColumn(event -> {
    if(event.getAvgRating() == 0.0){
        return "no rating";
    } else
        return event.getAvgRating();});
ratingColumn.setHeader("Rating");
grid.addComponentColumn(item -> createUseAsTemplateButton(item)).setHeader("Use As Template");
descriptionColumn.setFlexGrow(3);

grid.addSelectionListener(event -> {
    Set<EventTemplate> temp = event.getAllSelectedItems();
    createDialogBoxForTemplate(temp.iterator().next());
});
```

BurnOUTEvents

Scrum retrospective

- Positive Erfahrungen:
  - ÄNDERUNGEN & Entscheide können schnell umgesetzt werden
  - Jeder weiss immer an was er und das Team ist

- Negative Erfahrungen:
  - Code Qualität leidet teilweise durch schnelle Änderungen und zusammenfügen von Code der verschiedenen Entwickler
  - Merging wird aufwändig wenn mehrere Entwickler an gleichen/voneinander abhängigen Klassen arbeiten

- Learnings:
  - Konsequenter sein, sich mehr treffen und besser absprechen
  - Explizit Zeit für Recherche/Selbststudium einplanen
  - Vieles über GIT und wie man es effizient einsetzt

- Fazit/unsere Meinung:
  - Gewisse Voraussetzungen, dass Scrum funktioniert
  - Task Aufteilung war dadurch beeinflusst