

LEEZE TEIL 4

ADMIN STUFF

TASKBOARD



USER STORY 6

Ein Administrator soll Touren
verwalten können.

USER STORY 6

- ▶ Rollenverwaltung einbauen (rolify)
- ▶ Administrationsoberfläche nutzen (ActiveAdmin)
- ▶ Autorisierungsframework nutzen (CanCan)

AUTHENTIFIZIERUNG, AUTORISIERUNG, ROLLEN

- ▶ **Authentifizierung**: Nachweis, dass ein Nutzer derjenige ist, für den er sich ausgibt.
- ▶ **Autorisierung**: Einräumung bestimmter Rechte aufgrund von Eigenschaften eines Nutzers.
- ▶ **Rollen**: Eigenschaft eines Nutzers, die etwa bei der Autorisierung genutzt wird

BEISPIEL

- ▶ Besuch des Wahlbüros bei der Bundestagswahl
 - ▶ **Authentifizierung:** Vorzeigen des Personalausweises, Wahlhelfer prüft Foto.
 - ▶ **Autorisierung:** Wahlhelfer prüft mit Hilfe des Wählerverzeichnis, ob in diesem Wahlbüro gewählt werden darf.
- ▶ **Rollen:** Wahlberechtigter im aktuellen Wahlbüro, Wahlberechtigter in anderem Wahlbüro, Minderjähriger, ausländischer Staatsbürger, ...

ANWENDUNG IN RUBY ON RAILS

- ▶ Authentifizierung: Devise
 - ▶ Nutzerregistrierung (Passwort, Facebook, Github, ...)
 - ▶ Nutzerlogin und -logout
- ▶ Autorisierung: CanCan
 - ▶ Nutzern werden aufgrund bestimmter Eigenschaften (z.B. Rollen) Rechte zugewiesen
 - ▶ Nicht berechnigte Nutzer werden abgefangen
- ▶ Rollenmanagement: rolify
 - ▶ Verwaltet Rollen als Eigenschaften eines Nutzers
 - ▶ Ein Nutzer hat mehrere Rollen, eine Rolle darf mehrere Nutzer haben

USER STORY 6

*Ein Administrator soll Touren
verwalten können.*

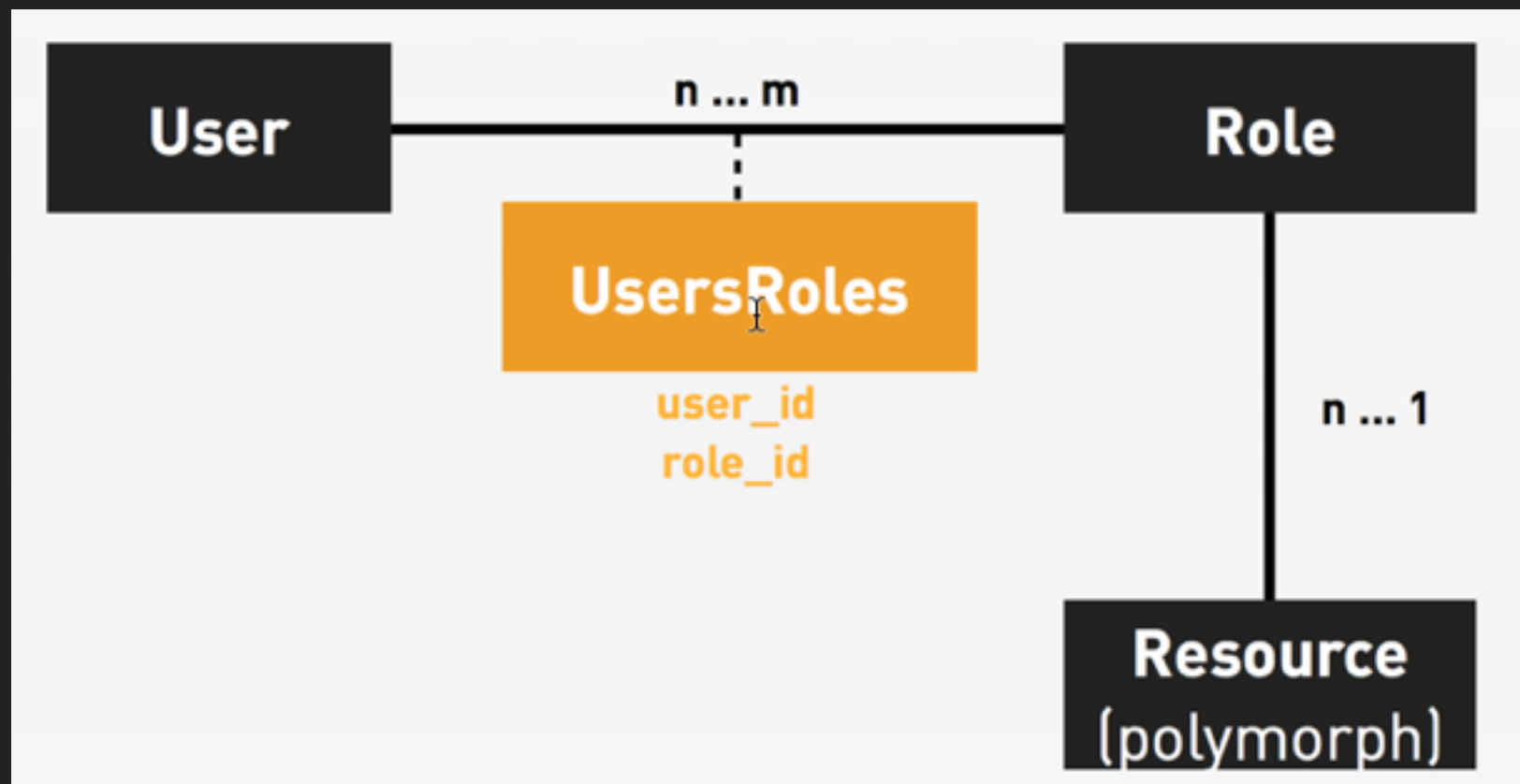
ROLLENVERWALTUNG EINBAUEN

USER STORY 6: ROLIFY

- ▶ Rubygem zur einfachen Verwaltung von Rollen
 - ▶ <https://github.com/EppO/rolify>
- ▶ Funktionen:
 - ▶ Globale Rollen (z.B. Admin für alle Touren)
 - ▶ Ressourcenspezifische Rollen (z.B. Admin für Tour #1)
- ▶ Nutzung:
 - ▶ Gemfile: `gem 'rolify'`
 - ▶ `$ bundle install`

USER STORY 6: ROLIFY

- ▶ `$ bundle exec rails generate rolify Role User`
- ▶ Legt ein neues Role-Model sowie eine Join-Tabelle an
- ▶ `$ bundle exec rake db:migrate`



USER STORY 6: ROLIFY

- ▶ Globale Rolle vergeben
 - ▶ `User.first.add_role(:admin)`
- ▶ Ressourcenspezifische Rolle vergeben
 - ▶ `User.first.add_role(:admin, Tour.first)`
- ▶ Rollen abfragen
 - ▶ `user.has_role?(:admin)`
 - ▶ `user.has_role?(:admin, Tour.first)`

BEISPIEL: ERZEUGUNG EINES ADMIN NUTZERS

```
1  Category.create(name: 'Radtour')
2  Category.create(name: 'Mountainbiketour')
3  Category.create(name: 'Radwandern')
4
5  pass = SecureRandom.hex(5)
6  admin = User.create email: 'admin@example.com',
7                password: pass,
8                password_confirmation: pass
9  admin.add_role :admin
10 puts "Admin password is #{pass}"
```

USER STORY 6

*Ein Administrator soll Touren
verwalten können.*

ADMIN TOOL NUTZEN

USER STORY 6: ACTIVEADMIN

- ▶ Generische Administrationsoberfläche für Models
 - ▶ <http://www.activeadmin.info/>
 - ▶ Railscast: <http://railscasts.com/episodes/284-active-admin>
- ▶ Funktionen:
 - ▶ Navigation über Ressourcen/Models
 - ▶ Übersichtsseiten
 - ▶ Suchformular, Filter
 - ▶ CRUD-Funktionen

USER STORY 6: ACTIVEADMIN

► Nutzung:

- Gemfile: `gem 'activeadmin', github: 'activeadmin'`
- `$ bundle install`
- `$ bundle exec rails generate active_admin:install --skip-users`
- `$ bundle exec rake db:migrate`
- <http://localhost:3000/admin>

USER STORY 6: ACTIVEADMIN

- ▶ Konfiguration:
 - ▶ ActiveAdmin nutzt von Haus aus eine eigene Authentifizierung
 - ▶ Wir haben bereits Devise
- ▶ --> Devise und ActiveAdmin kombinieren

USER STORY 6: ACTIVEADMIN

► config/initializers/active_admin.rb

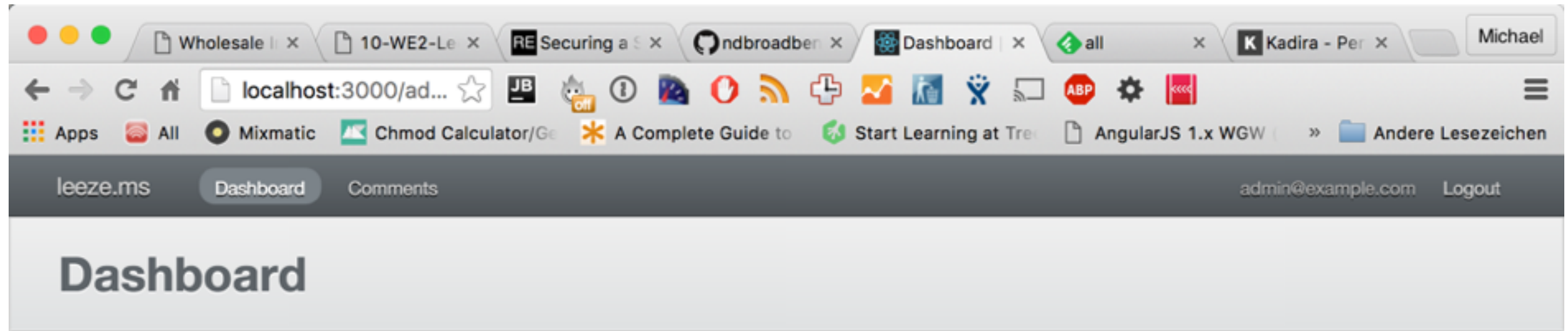
```
1 ActiveAdmin.setup do |config|  
2   config.site_title = "leeze.ms"  
3   config.authentication_method = :authenticate_admin_user!  
4   config.current_user_method = :current_user  
5   config.logout_link_path = :destroy_user_session_path  
6   config.logout_link_method = :delete  
7   config.batch_actions = true
```

USER STORY 6: ACTIVEADMIN

► app/controllers/application_controller.rb

```
1 class ApplicationController < ActionController::Base
2   # Prevent CSRF attacks by raising an exception.
3   # For APIs, you may want to use :null_session instead.
4   protect_from_forgery with: :exception
5
6   def authenticate_admin_user!
7     redirect_to new_user_session_path unless current_user &&
8       current_user.has_role?(:admin)
9   end
10 end
11
```

USER STORY 6: ACTIVEADMIN



Welcome to Active Admin. This is the default dashboard page.
To add dashboard sections, checkout 'app/admin/dashboard.rb'

USER STORY 6: ACTIVEADMIN

- ▶ Ressourcen / Models müssen explizit zu ActiveAdmin hinzugefügt werden.
- ▶ Verwaltbare Ressourcen werden in `app/admin` abgelegt
- ▶ Beispiel:
 - ▶ `$ rails generate active_admin:resource Tour`
 - ▶ `$ rails generate active_admin:resource Category`

▶ app/admin/tour.rb

```
1  ActiveAdmin.register Tour do
2    # See permitted parameters documentation:
3    # https://github.com/activeadmin/activeadmin/blob \
4    # /master/docs/2-resource-customization.md#setting-up-strong-parameters
5    #
6    # permit_params :list, :of, :attributes, :on, :model
7    #
8    # or
9    #
10   # permit_params do
11     #   permitted = [:permitted, :attributes]
12     #   permitted << :other if resource.something?
13     #   permitted
14   # end
15 end
```

Michael

Categories

localhost:3000...

Apps All Mixmatic Chmod Calculator/Ge A Complete Guide to Start Learning at Tre

Andere Lesezeichen

leeze.ms Dashboard Categories Comments Tours admin@example.com Logout

ADMIN /

Categories

New Category

Batch Actions

<input type="checkbox"/>	Id	Name	Created At	Updated At	
<input type="checkbox"/>	3	Radwandern	January 25, 2016 09:24	January 25, 2016 09:24	View Edit Delete
<input type="checkbox"/>	2	Mountainbiketour	January 25, 2016 09:24	January 25, 2016 09:24	View Edit Delete
<input type="checkbox"/>	1	Radtour	January 25, 2016 09:24	January 25, 2016 09:24	View Edit Delete

Download: [CSV](#) [XML](#) [JSON](#)

Displaying all 3 Categories

Filters

TOUR CATEGORIES

Any

TOURS

Any

NAME

Contains

CREATED AT

-

UPDATED AT

-

Filter

Clear Filters

Powered by [Active Admin](#) 1.0.0.pre2

USER STORY 6: ACTIVEADMIN

The screenshot displays the 'New Tour' form within the Leeze Admin interface. The top navigation bar includes the site name 'leeze.ms' and links to 'Dashboard', 'Categories', 'Comments', and 'Tours'. The user is logged in as 'admin@example.com' with a 'Logout' option. The breadcrumb trail shows 'ADMIN / TOURS /'. The form itself is titled 'New Tour' and contains four input fields: 'Title*', 'Teaser*', 'Description*', and 'Url*'. The 'Description*' field is a large text area, while the others are standard text inputs. At the bottom of the form, there are two buttons: 'Create Tour' and 'Cancel'. A mouse cursor is visible over the right side of the form area.

leeze.ms Dashboard Categories Comments **Tours** admin@example.com Logout

ADMIN / TOURS /

New Tour

Title*

Teaser*

Description*

Url*

Create Tour Cancel

USER STORY 6: ACTIVEADMIN

- ▶ **Problem:** Touren können keiner Kategorie zugeordnet werden, da das Formularfeld fehlt.
- ▶ **Lösung:** ActiveAdmin-Formulare lassen sich ähnlich wie Views anpassen
 - ▶ Formular-Anpassungen werden in `app/admin/tour.rb` vorgenommen

USER STORY 6: ACTIVEADMIN

► app/admin/tour.rb

```
1  ActiveAdmin.register Tour do
2    permit_params :title, :teaser, :description, :url, category_ids: []
3
4    form do |f|
5      f.inputs 'Details' do
6        f.input :title
7        f.input :teaser
8        f.input :description
9        f.input :url, as: :string
10       f.input :categories
11     end
12     f.actions
13   end
14 end
15
```

leeze.ms

Dashboard

Categories

Tours

admin@example.com

Logout

ADMIN / TOURS /

New Tour

Details

Title*

Teaser*

Description*

Url*

Categories*

Radtour

Mountainbiketour

Radwandern

Create Tour

Cancel

Powered by [Active Admin](#) 1.0.0.pre2

USER STORY 6

*Ein Administrator soll Touren
verwalten können.*

AUTORISIERUNGS-FRAMEWORK NUTZEN

USER STORY 6: AUTORISIERUNGS-FRAMEWORK NUTZEN

▶ Problem:

- ▶ Bisher haben wir nur eine Rolle (Admin), die alles darf
- ▶ Es wird mehrere Rollen geben (Gast, registrierter Nutzer, Admin)
- ▶ Diese Rollen haben alle unterschiedliche Berechtigungen
- ▶ --> Regelung, wer was darf, wird unübersichtlich
- ▶ Lösung: Autorisierungsframework nutzen!

USER STORY 6: CANCAN

- ▶ Framework zur Definition und Abfrage von Rechten
 - ▶ <http://github.com/ryanb/cancan>
 - ▶ Railscast: <http://railscasts.com/episodes/192-authorization-withcancan>
- ▶ Funktionen:
 - ▶ Zentrale Definition von Berechtigungen (durch DSL)
 - ▶ Einfache Abfrage von Berechtigungen
 - ▶ Automatisches Laden und Autorisieren von Ressourcen

USER STORY 6: CANCAN

- ▶ Nutzung:

- ▶ Gemfile: `gem 'cancan'`

- ▶ `$ bundle install`

- ▶ `$ rails generate cancan:ability`

USER STORY 6: CANCAN

- ▶ Definition von Berechtigungen
 - ▶ Zentraler Ort: `app/models/ability.rb`
 - ▶ Syntax: `can <Berechtigung>, <Scope>`
 - ▶ Beispiel:
 - ▶ `can :manage, Tour`
 - ▶ `can :create, Tour`
 - ▶ `can :delete, Category`

USER STORY 6: CANCAN

► Definition von Berechtigungen

```
1  class Ability
2      include CanCan::Ability
3
4      def initialize(user)
5          if user.present?
6              if user.has_role?(:admin)
7                  # Admins
8                  can :manage, :all
9              else
10                 # Registered users
11                 can :read, :all
12             end
13         else
14             # Guest users
15             can :read, :all
16         end
17     end
18 end
```


USER STORY 6: CANCAN

- ▶ Abfrage von Berechtigungen
 - ▶ Syntax: `can? <Berechtigung>, <Scope>`
 - ▶ Beispiel:
 - ▶ `can? :manage, Tour`
 - ▶ `can? :create, Tour`
 - ▶ `can? :delete, Category`
- ▶ Autorisierung von Controller-Actions:
 - ▶ `load_and_authorize_resource`

USER STORY 6: CANSAN

```
1 class ToursController < ApplicationController
2   before_action :set_tour, only: [:show, :edit, :update, :destroy]
3   load_and_authorize_resource
4
5   # GET /tours
6   # GET /tours.json
7   def index
8     @tours = if params[:category_id].present?
9               Category.find(params[:category_id]).tours
10            else
11              Tour.all
12            end
13   end
14
```

USER STORY 6: CANCAN

- ▶ Nicht autorisierte Zugriffe werden abgefangen

CanCan::AccessDenied in ToursController#edit

You are not authorized to access this page.

Extracted source (around line #208):

```
206     if cannot?(action, subject, *args)
207       message ||= unauthorized_message(action, subject)
208       raise AccessDenied.new(message, action, subject)
209     end
210     subject
211   end
```

Rails.root: /Users/mjohann/projects/fhms/ws2015/leeze.ms

[Application Trace](#) | [Framework Trace](#) | [Full Trace](#)

```
cancan (1.6.10) lib/cancan/ability.rb:208:in `authorize!'
cancan (1.6.10) lib/cancan/controller_additions.rb:228:in `authorize!'
```

USER STORY 6: CANCEL

- Sichtbarkeit von Links einschränken

```
19      <td><%= link_to 'Show', tour %></td>
20      <td><%= link_to 'Edit', edit_tour_path(tour) %></td>
21      <td><%= link_to 'Destroy', tour, method: :delete, data: { method: :delete, confirm: true } %></td>
22    </tr>
23  <% end %>
24 </table>
25 <br>
26
27 <% if can?(:create, Tour) %>
28   <%= link_to 'New Tour', new_tour_path %>
29 <% end %>
```

USER STORY 6: CANCEL

[LeezeMs](#)[Radtour](#)[Mountainbiketour](#)[Radwandern](#)[Welcome, mjohann@rails-experts.com](#)

Listing Tours

- [Radtour](#)
- [Mountainbiketour](#)
- [Radwandern](#)

Title			
Geführte Tour von der FH zu "Burg Erking"	Show	Edit	Destroy

Sidebar



Sidebar

[Link1](#)

[Link2](#)

[Link3](#)

© Company 2015



USER STORY 6: CANGAN

► CanCan für ActiveAdmin nutzen

```
1 class ApplicationController < ActionController::Base
2   # Prevent CSRF attacks by raising an exception.
3   # For APIs, you may want to use :null_session instead.
4   protect_from_forgery with: :exception
5
6   def authenticate_admin_user!
7     redirect_to new_user_session_path unless can?(:manage, :all)
8   end
9 end
10
```

TASKBOARD

