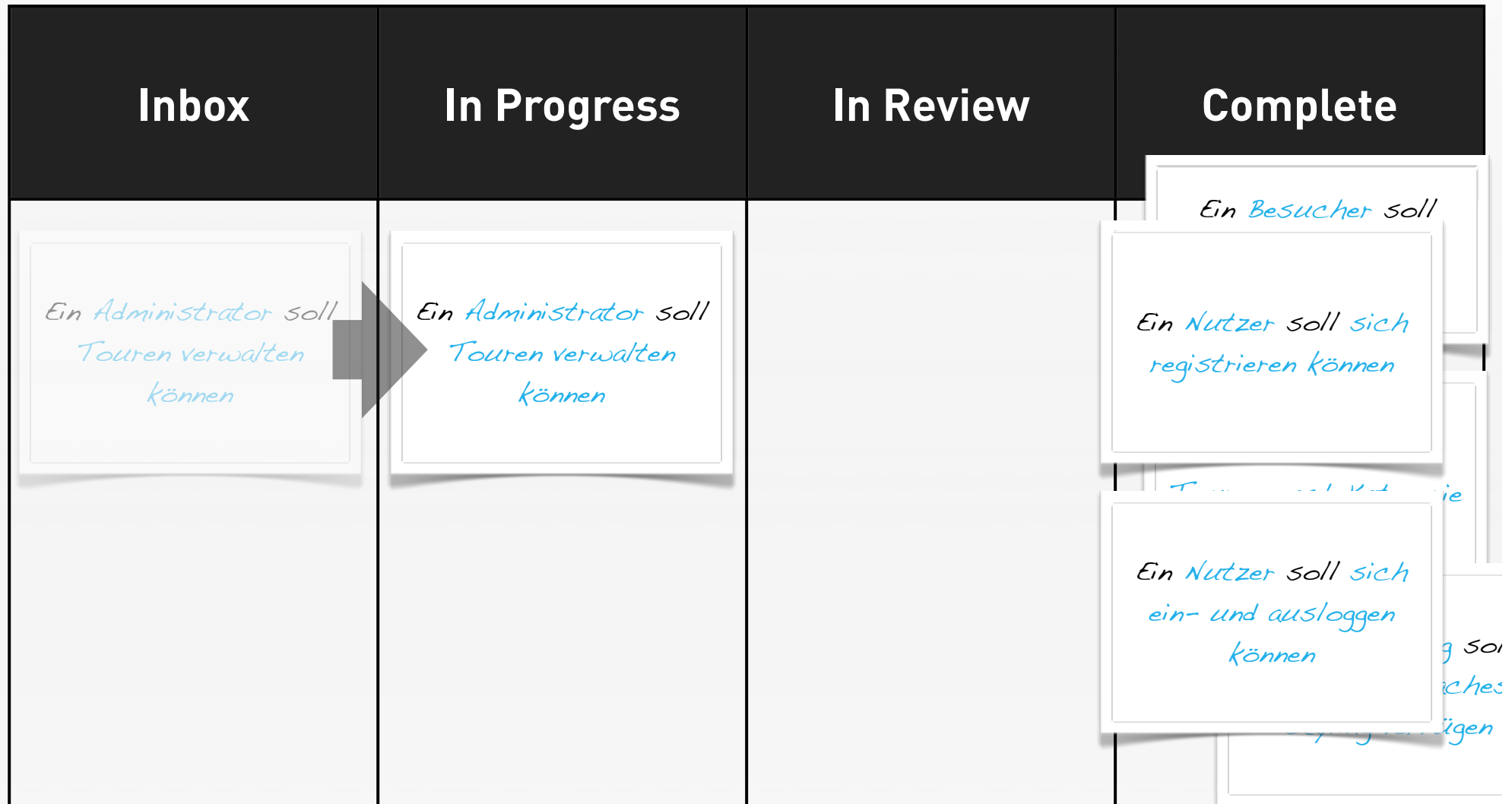




Vorlesung 10

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 Rails

- **Authentifizierung: Devise**
 - Nutzerregistrierung (Passwort, Facebook, Github, ...)
 - Nutzerlogin und -logout
- **Autorisierung: CanCan**
 - Nutzern werden aufgrund bestimmter Eigenschaften (z.B. Rollen) Rechte zugewiesen
 - Nicht berechtigte 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.*

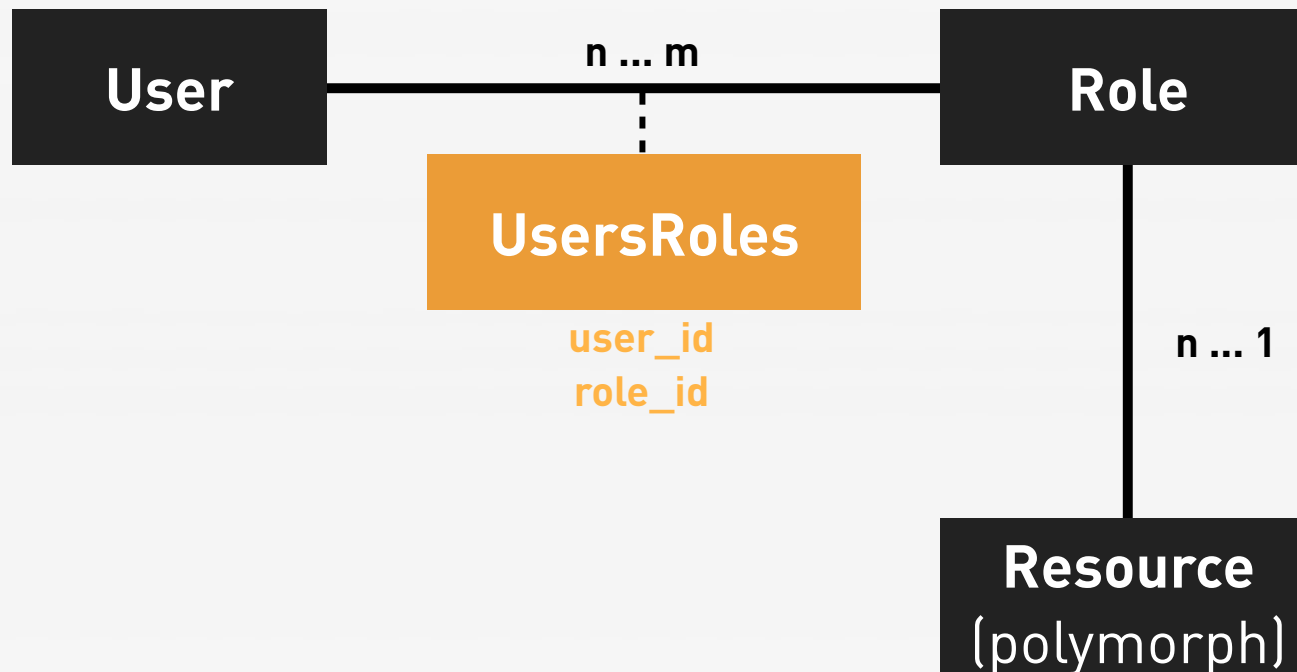
Task:
Rollenverwaltung einbauen

Story 6: rolify

- Rubygem zur einfachen Verwaltung von Rollen
 - <https://github.com/Epp0/rolify>
- Funktionen:
 - Globale Rollen (z.B. Admin für alle Touren)
 - Ressourcenspezifische Rollen (z.B. Admin für Tour #1)
- Nutzung:
 - Gemfile: gem 'rolify'
 - \$ bundle install

Story 6: rolify

- \$ rails generate rolify:role Role User
 - Legt ein neues Role-Model sowie eine Join-Tabelle an



- \$ rake db:migrate

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)`

Story 6: rolify

- Beispiel: Erzeugung eines Admin-Nutzers

```
db/seeds.rb

Category.create name: 'Radtour'
Category.create name: 'Mountainbiketour'
Category.create name: 'Radwandern'

pass = SecureRandom.hex(5)
admin = User.create email: 'admin@example.com', password: pass,
password_confirmation: pass

admin.add_role :admin
puts "Admin password is #{pass}"
```

User Story 6

*Ein Administrator soll Touren
verwalten können.*

Task:
Administrationsoberfläche nutzen

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
 - ...

Story 6: ActiveAdmin

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

Story 6: ActiveAdmin

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

Story 6: ActiveAdmin

- config/initializers/active_admin.rb

```
config/initializers/active_admin.rb

ActiveAdmin.setup do |config|
  config.site_title = "leeze.ms"
  config.authentication_method = :authenticate_admin_user!
  config.current_user_method = :current_user
  config.logout_link_path = :destroy_user_session_path
  config.logout_link_method = :delete
  config.batch_actions = true
end
```

Story 6: ActiveAdmin

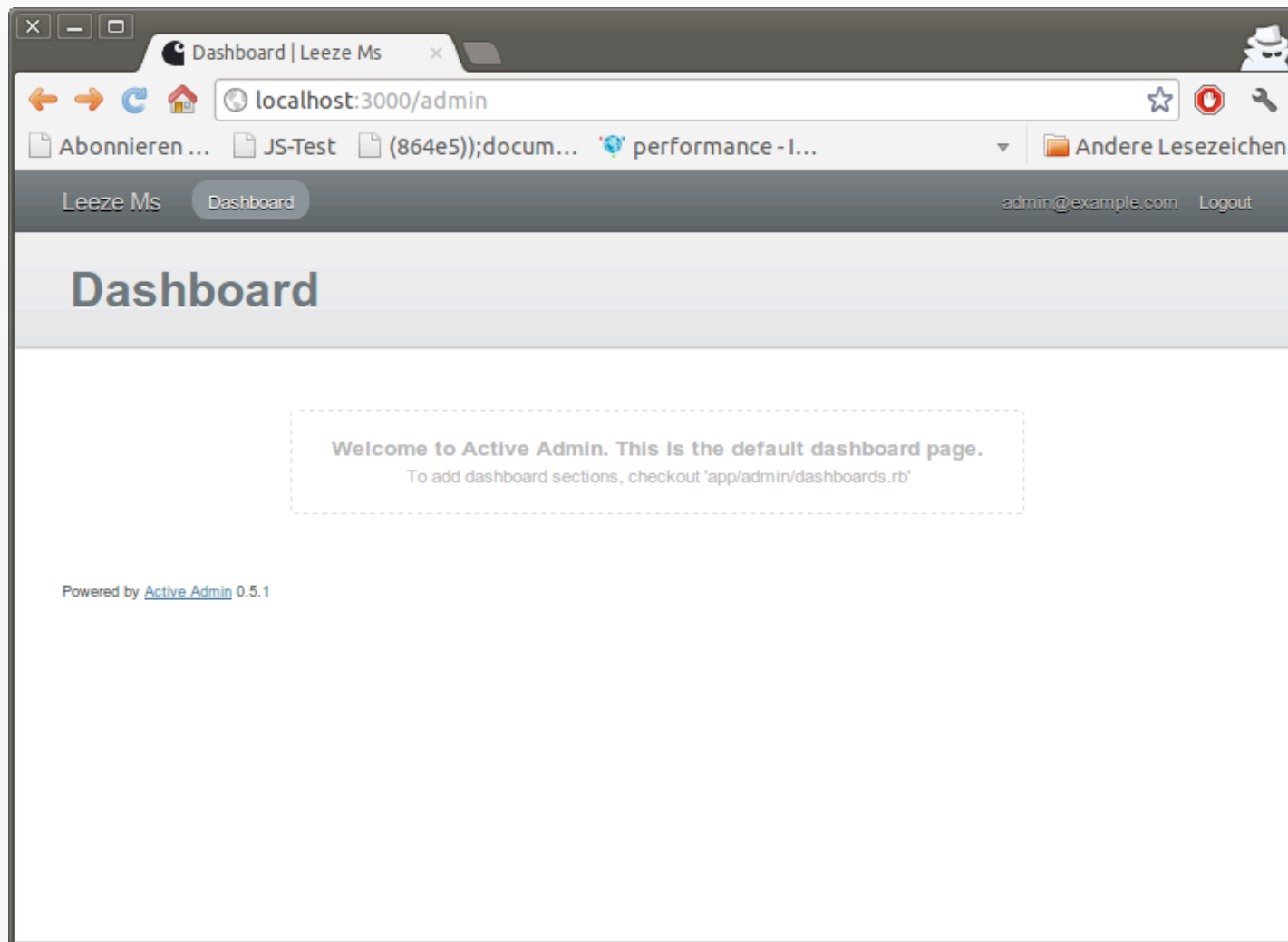
- app/controllers/application_controller.rb:

```
app/controllers/application_controller.rb

class ApplicationController < ActionController::Base
  protect_from_forgery with: :exception

  def authenticate_admin_user!
    redirect_to new_user_session_path unless current_user &&
current_user.has_role?(:admin)
  end
end
```

Story 6: ActiveAdmin



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`

Story 6: ActiveAdmin

- app/admin/tours.rb

```
app/admin/tours.rb

ActiveAdmin.register Tour do
  # See permitted parameters documentation:
  # https://github.com/gregbell/active\_admin/blob/master/docs/2-resource-
  # customization.md#setting-up-strong-parameters
  #
  # permit_params :list, :of, :attributes, :on, :model
  #
  # or
  #
  # permit_params do
  #   permitted = [:permitted, :attributes]
  #   permitted << :other if resource.something?
  #   permitted
  # end
end
```

Story 6: ActiveAdmin

The screenshot shows a web browser window with the URL `localhost:3000/admin/categories`. The page title is "Categories | Leeze Ms". The browser's address bar shows the URL, and the tabs bar shows several open tabs. The page header includes navigation links for "Leeze Ms", "Dashboard", "Categories", and "Tours", along with the user email "admin@example.com" and a "Logout" link. The main content area is titled "ADMIN / Categories" and features a "New Category" button. Below this, there is a "Batch Actions" dropdown and a table of categories. The table has columns for "Id", "Name", "Created At", and "Updated At". It lists three categories: "Radwandern", "Mountainbiketour", and "Radtour". Each row has "View", "Edit", and "Delete" links. To the right of the table is a "Filters" sidebar with search and date range filters. At the bottom, there are download links for "CSV", "XML", and "JSON", and a status message "Displaying all 3 Categories". The footer indicates the application is "Powered by Active Admin 0.5.1".

Categories | Leeze Ms

localhost:3000/admin/categories

Abonnieren ... JS-Test (864e5));docum... performance - I...

Leeze Ms Dashboard Categories Tours admin@example.com Logout

ADMIN / Categories New Category

Batch Actions

	Id	Name	Created At	Updated At	
<input type="checkbox"/>	6	Radwandern	January 10, 2013 13:42	January 10, 2013 13:42	View Edit Delete
<input type="checkbox"/>	5	Mountainbiketour	January 10, 2013 13:42	January 10, 2013 13:42	View Edit Delete
<input type="checkbox"/>	4	Radtour	January 10, 2013 13:42	January 10, 2013 13:42	View Edit Delete

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

Displaying all 3 Categories

Powered by [Active Admin](#) 0.5.1

Filters

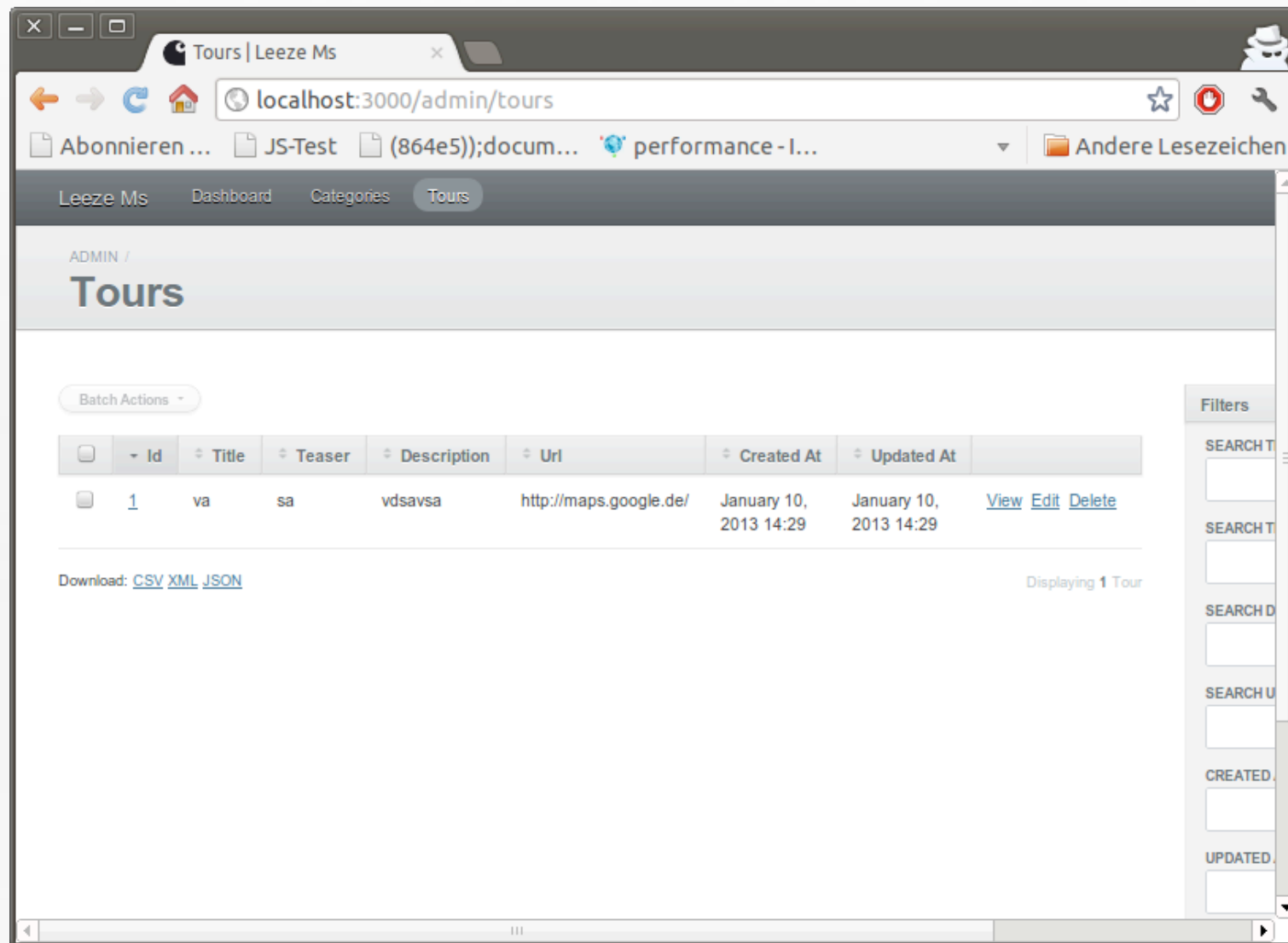
SEARCH NAME

CREATED AT

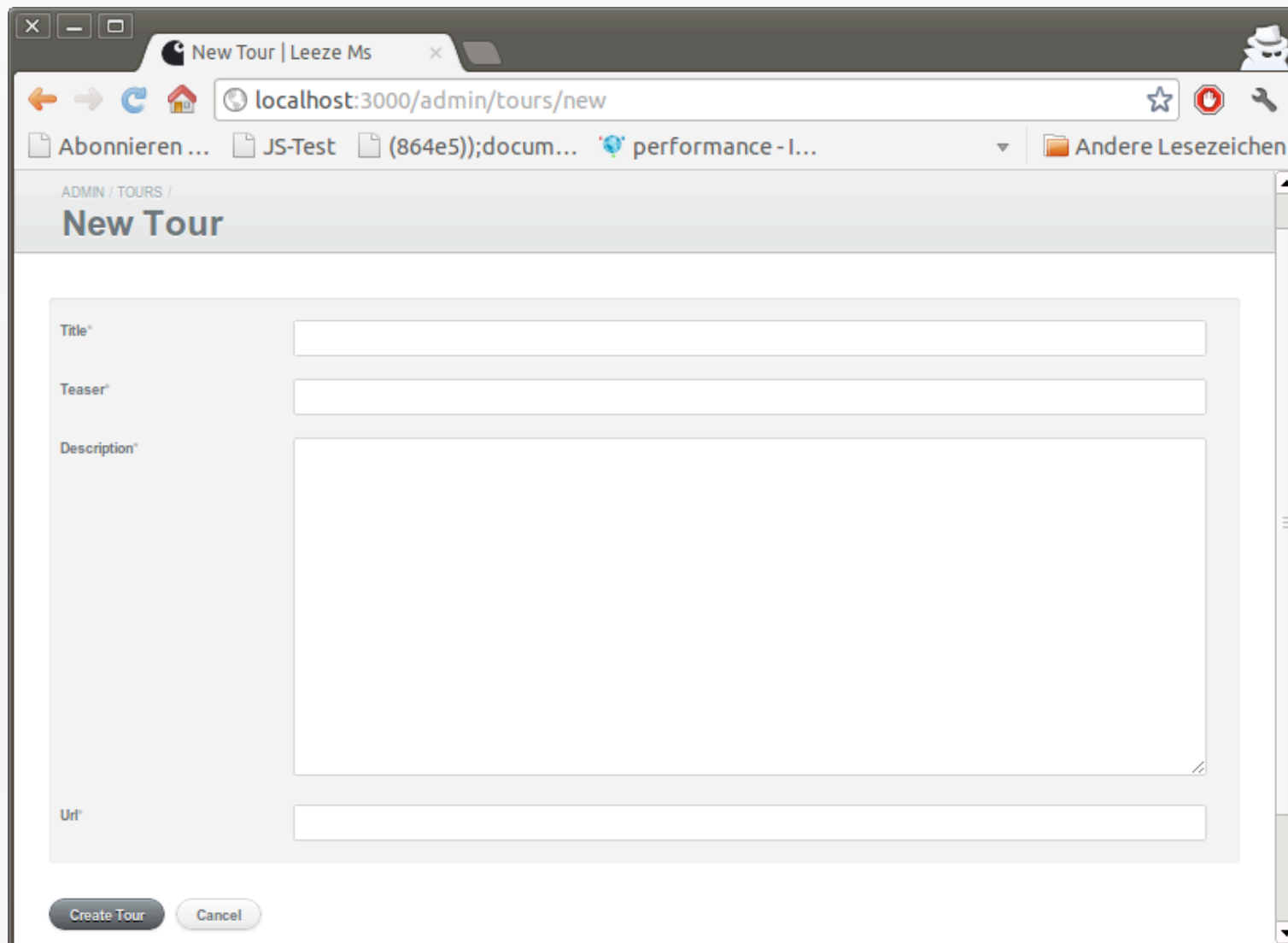
UPDATED AT

Filter Clear Filters

Story 6: ActiveAdmin



Story 6: ActiveAdmin



The screenshot shows a web browser window with the title 'New Tour | Leeze Ms'. The address bar displays 'localhost:3000/admin/tours/new'. The browser's tab bar shows several open tabs: 'Abonnieren ...', 'JS-Test', '(864e5));docum...', and 'performance - I...'. The page content is titled 'ADMIN / TOURS / New Tour'. It features a form with four input fields: 'Title*', 'Teaser*', 'Description*', and 'Url*'. The 'Description*' field is a large text area. At the bottom of the form, there are two buttons: 'Create Tour' and 'Cancel'.

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/tours.rb` vorgenommen

Story 6: ActiveAdmin

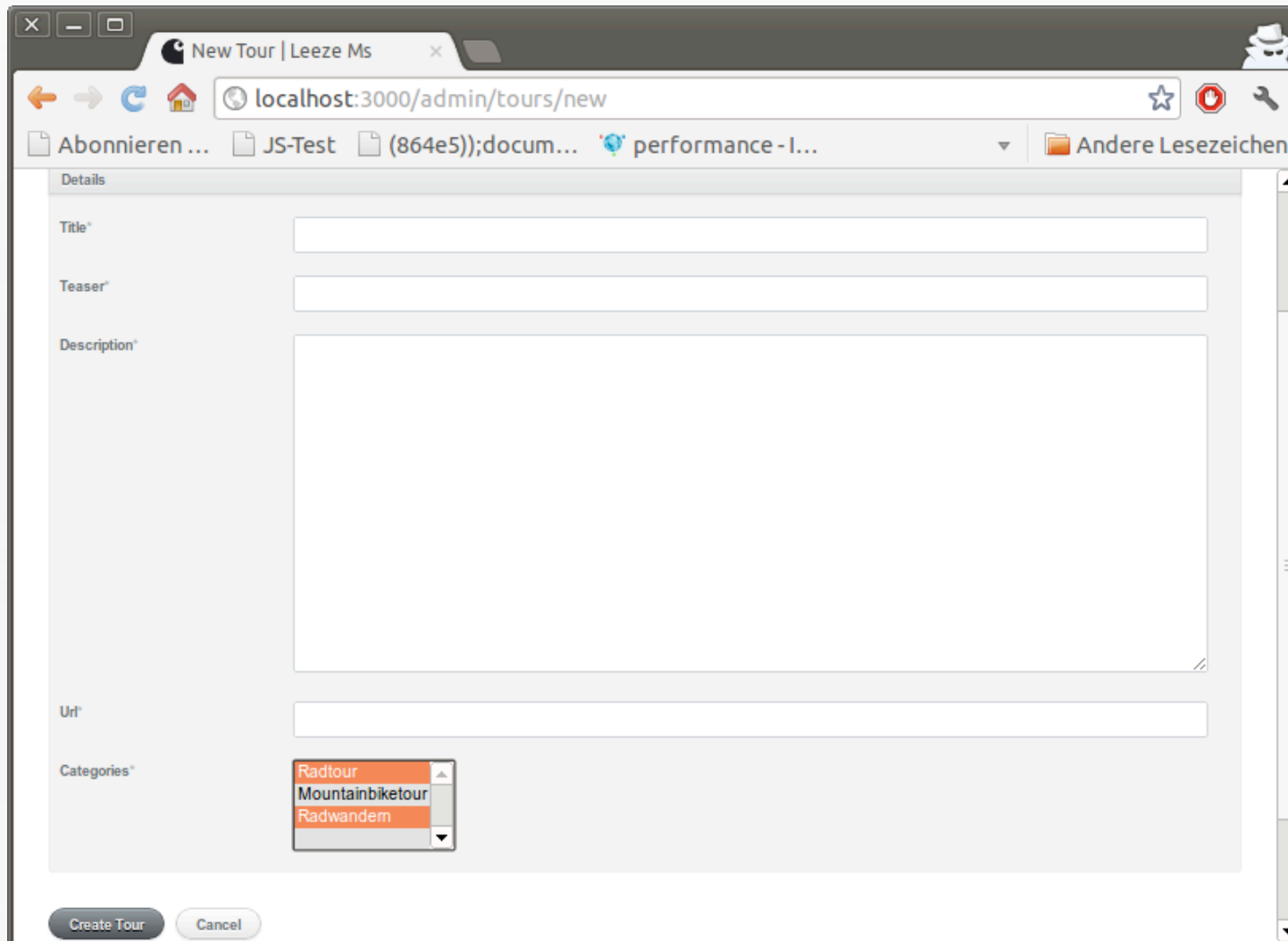
- app/admin/tours.rb:

```
app/admin/tours.rb

ActiveAdmin.register Tour do
  permit_params :title, :teaser, :description, :url, category_ids: []

  form do |f|
    f.inputs "Details" do
      f.input :title
      f.input :teaser
      f.input :description
      f.input :url
      f.input :categories
    end
    f.actions
  end
end
```

Story 6: ActiveAdmin



The screenshot shows a web browser window with the title 'New Tour | Leeze Ms'. The address bar displays 'localhost:3000/admin/tours/new'. The browser's tab bar shows several open tabs: 'Abonnieren ...', 'JS-Test', '(864e5));docum...', and 'performance - I...'. The main content area is a form titled 'Details' for creating a new tour. The form includes the following fields:

- Title***: A text input field.
- Teaser***: A text input field.
- Description***: A large text area for a detailed description.
- Url***: A text input field.
- Categories***: A dropdown menu with three options: 'Radtour' (highlighted in orange), 'Mountainbiketour', and 'Radwandern'.

At the bottom of the form, there are two buttons: 'Create Tour' and 'Cancel'.

User Story 6

*Ein Administrator soll Touren
verwalten können.*

Task:
Autorisierungsframework nutzen

Story 6: Autorisierungsframework

- 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!

Story 6: CanCan

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

Story 6: CanCan

- **Nutzung:**
 - Gemfile: `gem 'cancan'`
 - `$ bundle install`
 - `$ rails generate cancan:ability`

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`

Story 6: CanCan

- Definition von Berechtigungen:

```
app/models/ability.rb

class Ability
  include CanCan::Ability
  def initialize(user)
    if user.present?
      if user.has_role?(:admin)
        # Admins
        can :manage, :all
      else
        # Registered users
        can :read, :all
      end
    else
      # Guest users
      can :read, :all
    end
  end
end
```

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

Story 6: CanCan

- Abfrage von Berechtigungen:

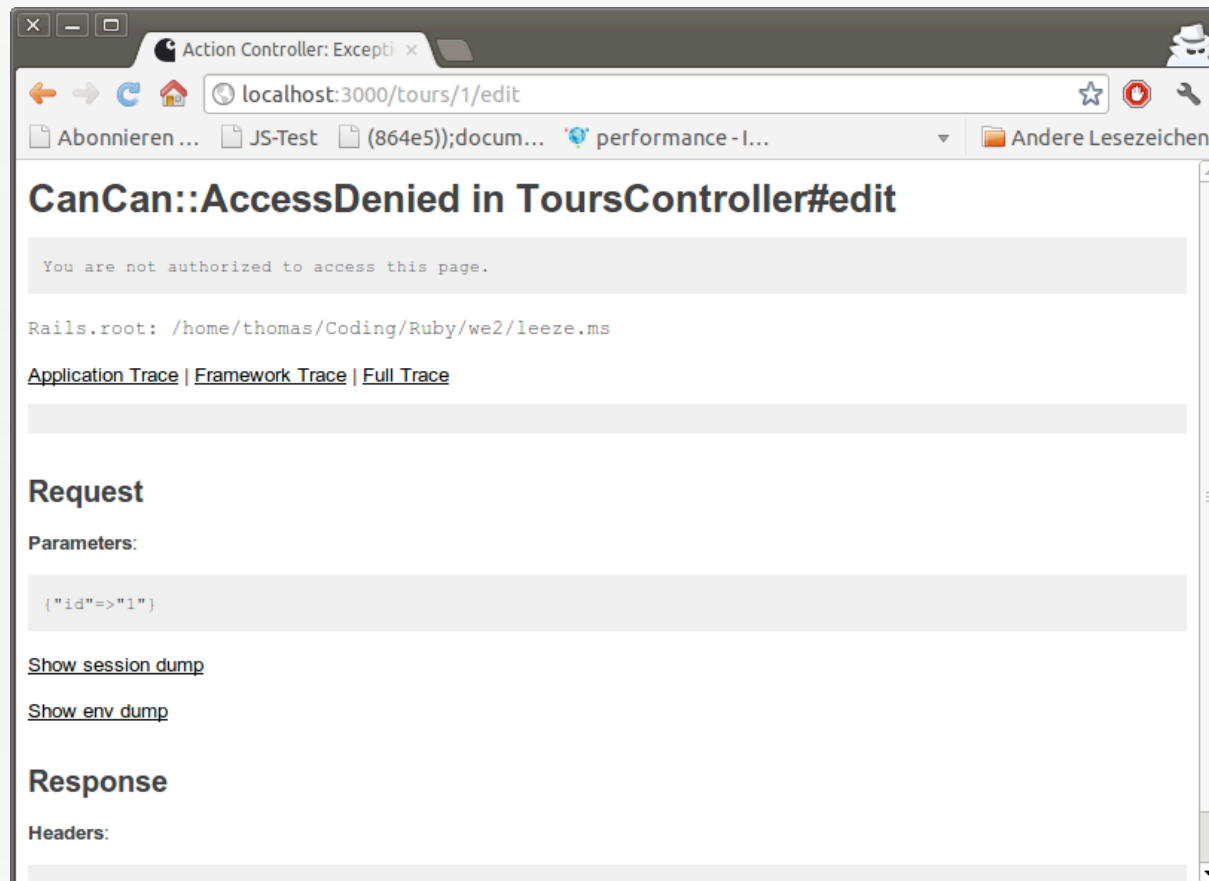
```
app/controllers/tours_controller.rb

class ToursController < ApplicationController
  load_and_authorize_resource

  # ...
end
```

Story 6: CanCan

- Nicht autorisierte Zugriffe werden abgefangen:



Story 6: CanCan

- Sichtbarkeit von Links einschränken:

```
app/views/tours/index.html.erb

<h1>Listing tours</h1>

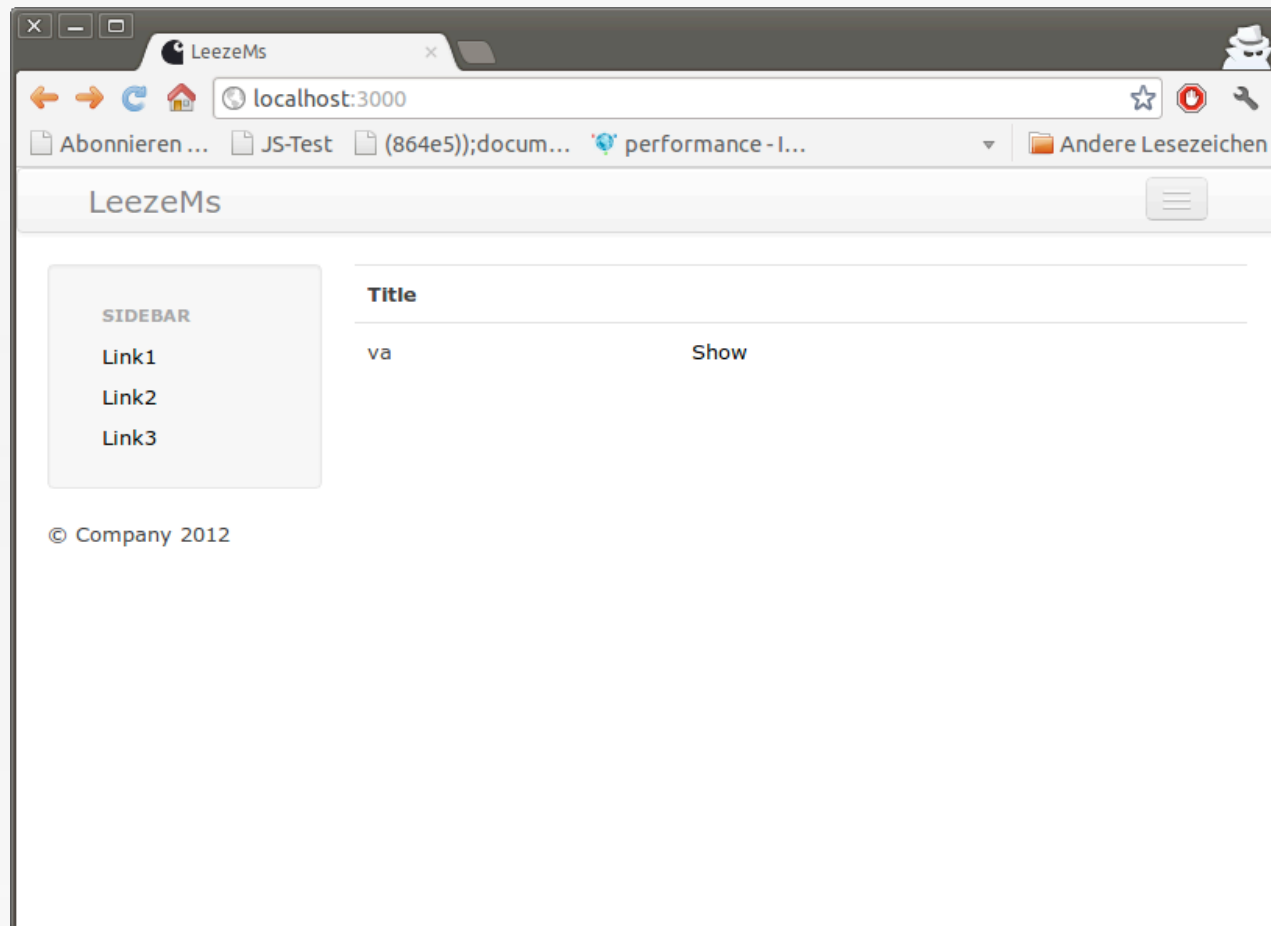
<ul>
  <% @tours.each do |tour| %>
    <li><%= link_to tour.title, tour %></li>
  <% end %>
</ul>

<br>

<% if can?(:create, Tour) %>
  <%= link_to 'New Tour', new_tour_path %>
<% end %>
```

Story 6: CanCan

- Sichtbarkeit von Links einschränken:



Story 6: CanCan

- CanCan für ActiveAdmin-Autorisierung nutzen:

```
app/controllers/application_controller.rb

class ApplicationController < ActionController::Base
  protect_from_forgery

  def authenticate_admin_user!
    redirect_to new_user_session_path unless can?(:manage, :all)
  end
end
```

Taskboard

