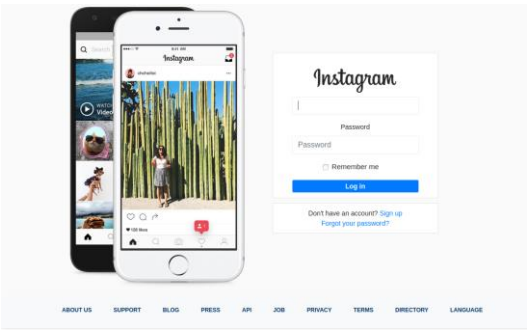
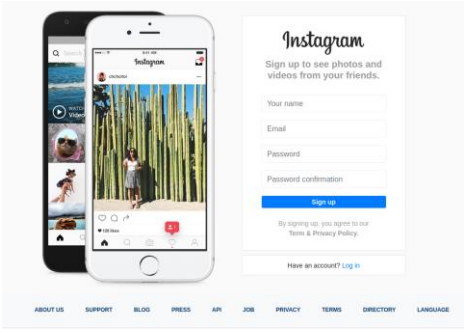
Quicknote 1 AB1-AB3 Luca Frei

Verwendete Technologien

Devise

Beschreibung: Devise ist eine Authentifizierungskomponente, welche automatisch Views generiert. Diese Views können dann beliebig zum eigenen Nutzen angepasst werden. Beim Instagram baut unsere gesamte Authentifizierung auf dem Gem «Devise»
Vorteile: Durch Devise ist es simpel eine kleine Authentifizerung einzubinden ohne gross Code schreiben zu müssen. Man kann zusätzlich eigene Daten bei der Registrierung hinzufügen wie zum Beispiel beim Instagram einen Benutzernamen.
Nachteile: Ein Nachteil von Devise ist, dass man sich dem Gem anpassen muss und man nicht genau bestimmen kann wo und wie die Daten abgespeichert werden und auch gehasht werden.





Bootstrap

Beschreibung: Bootstrap wird zum style und strukturieren von HTML Komponenten verwendet. Beim Instagram wird zum Beispiel der Carousel Effekt beim Login und bei der Registrierung durch Bootstrap möglich.
Vorteile: Bootstrap hat einige sehr nützliche Styling-Features, welche zudem einfach eingebunden werden können.
Nachteile: Bei komplexen Funktionen muss man bei Bootstrap etwas nachdenken um die gewünschte Funktion zu realisieren.

Partialview

Beschreibung: Durch Partialviews ist es möglich mehrfach benötigten Code einfach wieder zu verwenden. Eine Partialview wird mit einem «_» vor dem Dateinamen definiert. Den benötigten Code kann dann einfach in das File geschrieben werden und in den Views gerendert werden, welche diese Komponente benötigen. In unserem Instagram haben wir zum Beispiel den Footer und das Dummy-Phone als Partialview deklariert, da diese Elemente in mehreren Views verwendet werden
Vorteile: Durch Partialviews wird viel redundanter Code eingespart. Wenn Änderungen an der Partalview vorgenommen werden wird die Änderung in allen Views aktualisiert, welche diese Partialview verwenden.
Nachteile: Im Beispiel unseres Dummy-Phones könnte es ein Nachteil sein, wenn man nur in einer

bestimmten View eine kleine Änderung vornehmen möchte. Da für alle Views die gleiche Partialview verwendet wird ist es nur beschränkt möglich Änderungen spezifisch für eine View zu machen.

.core-sprite

Beschreibung: Core-sprite ist ein Gem, welches es möglich macht Spritesheets zu verwenden. Durch core-sprite können alle benötigten Icons auf einem einzigen Bild gespeichert werden. Um ein Bild auszuwählen müssen im CSS Koordinaten angegeben werden. Beispielsweise das Instagram Signet wird beim Login und beim Registrieren verwendet.
Vorteile: Es muss nur ein Bild gerendert werden. Das Sprite bleibt dann im Cache und muss nicht immer wieder geladen werden, wenn ein anderes Bild benötigt wird.
Nachteile: CSS Koordinaten müssen angepasst werden, wenn das Sprite abgeändert wird und sich die Bilder nicht mehr an der gleichen Stelle befinden.

Gitlab

Beschreibung: Gitlab basiert wie schon im Namen enthalten auf Git. Gitlab wird verwendet um Projekte sowie unser Instagram abzuspeichern. Gitlab ermöglicht es mehreren Entwicklern gleichzeitig am gleichen Projekt zu arbeiten was sonst nur beschränkt möglich ist.
Vorteile: Der Code ist von überall erreichbar. Alle Arbeitsschritte sind sichtbar.
Nachteile: Es kann zu mühsamen Merge-konflikten kommen, wenn alle auf demselben Branch arbeiten.

Validierung

Beschreibung: Bei Rails kann man auf verschieden Arten Daten validieren. In unserem Instagram haben wird eine Validierung beim Model verwendet. Als zweites Beispiel könnte man auch auf der View direkt validieren. Validieren lassen sich eigentlich alle Daten.
Vorteile: Daten können simpel überprüft werden, dadurch können nicht erlaubte Werte und Daten abgefangen werden.
Nachteile: Validierung an sich hat nur einen Nachteil: Performance. Wenn man alle Daten validiert kann es zu eine bemerkbaren Performanceverlust kommen. Zu dem kommt es noch auf die Art der Validierung an.

```
validates :name, presence: true
validates :name, length: { maximum: 50}
```

Namenskonventionen

Beschreibung: Namenskonventionen werden verwendet um eine Applikation einheitlich zu gestalten. Zudem kann es einfacher sein sich im Code zurecht zu finden. Bei Rails werden zum Beispiel Controller immer Gross geschrieben. Views sollten im CamelCase erfasst werden.
Vorteile: Einheitliche Gestaltung der Applikation.
Nachteile: Wenn die Namenskonvektionen nicht eingehalten werden kann es zu Fehlern kommen. Beim Generieren von Controllern usw. könnten Dateien andere Namen als erwartet haben, wenn die Konvektionen nicht eingehalten werden.

Selbstreflexion

Was habe ich gelernt?

In den Arbeitsblättern habe ich gelernt wie einfach es ist gems zu installieren. Zudem war es sehr informativ mit Bootstrap zu arbeiten. Funktionen wie die Carousel-Funktion wusste ich zum Beispiel nicht wie man das implementieren kann. Auch die Validierung war für mich neu. Insgesamt war viel neues und Interessantes in den drei Arbeitsblättern.

Wie bin ich vorgegangen beim Lernen bzw. Ausführen des Auftrages?

Beim Ausfüllen der Arbeitsblätter habe ich mir einige Notizen zu den Aufgaben gemacht, sowie zu den verwendeten Technologien (siehe Notes Ordner). Ich habe versucht mich genau mit den Dingen zu beschäftigen, welche häufiger vorgekommen sind oder mir Probleme bereitet haben.

Was waren die Schwierigkeiten, wie konnte ich diese lösen?

Während dem Bearbeiten der Arbeitsblätter hatte ich eigentlich keine grossen Probleme. Ich musste lediglich einmal die Rails-Applikation einmal neustarten um einen Fehler zu beheben. Sonst hatte ich nur einige Tippfehler.

Was habe ich nicht verstanden bzw. was konnte ich nicht lösen?

Ich konnte alle Aufgaben ohne grössere Probleme lösen. Manchmal musste ich jedoch die Aufgaben zweimal lesen um genau zu verstehen was ich machen musste.

Was kann ich nächstes Mal besser machen?

Beim nächsten Mal möchte ich meine Notizen etwas genauer machen. Und auch bei jeder Aufgabe, welche ich nicht 100% verstehe genau beschreiben was ich nicht verstanden habe. Zudem möchte ich früher mit der Quicknote beginnen um die Qualität etwas zu steigern.

Fazit

Insgesamt kann ich sagen, dass ich viel Neues kennengelernt habe was ich auch in meiner täglichen Arbeit verwenden kann. Lediglich Bootstrap habe ich schon vorher gekannt. Devise und die Möglichkeit Partialviews zu erstellen war ganz neu für mich. Partialviews finde ich besonders spannend. Ich habe gar nicht gewusst, dass so etwas möglich ist bei Webseiten.

Lösungen Aufgabenblätter

AB1

Aufgabe 1

Beantworten Sie folgende Fragen unter Berücksichtigung der Namenskonventionen:

1. Was bedeutet `2.4.2 :001 > Book.connection ?`
2. Was bedeutet `2.4.2 :002 > Book ?`
3. Was bedeutet `2.4.2 :003 > Book.all ?`

Book.connection überprüft ob eine Verbindung zum Book besteht.

Book zeigt den Aufbau eines Book-Objekts.

Book.all gibt alle Books aus.

Aufgabe 3

Erstellen Sie folgendes Book-Objekt Book mit der create-Methode:
`#<Book title: "Linux Server", price: 13.0, subject_id: 2, description: "Linux is faster">`

```
Book.create(:title =>'Linux Server', :price => 13.0, :subject_id => 2,
:description => 'Linux is faster')
```

Erstellen Sie folgendes Book-Objekt mit der new- und save-Methode:
`#<Book title: "Ruby Book", price: 21.00, subject_id: 8, description: "Simple as that">`

```
b = Book.new()
b.title = "Ruby Book"
b.price = 21.00
b.subject_id = 8
b.description = "Simple as that"
```

Aufgabe 2

Vervollständigen Sie folgende Tabelle. Beachten Sie, dass alle Modell- und Klassennamen in Englisch geschrieben sind.

Aufgabe	Modell / Klasse	Tabellenname / Schema
	`Budget`	budgets
	BudgetPositions	`budget_positions`
	Person	`people`
	`Mouse`	mice
	FootballTeam	`football_teams`
	`City`	cities
	Berry	`berries`
	`ClockOfChurch`	clock_of_chruch
	Fox	`foxes`
	`Knife`	knifes
	Spoof	`spoofs`
	`Scarf`	scarves
	Neurose	`neuroses`
	`Analysis`	analysises
	Crise	`crises`
	`Zero`	zeros

AB2

1. Erklären Sie die 1. Zeile. Verwenden dazu den Befehl `rails routes` im Terminal (zeigt alle Routen der Applikation).
2. Wohin geht der Link?

Die Session des eingelogten users wird durch diese Zeile beendet und leitet ihn zur Seite /users/sign_out weiter.

1. Was ist das Gemfile und wieso müssen wir diesen Eintrag erstellen?
2. Wie installiert man ein Gem?

Im Gemfile werden features für die Applikation mittels dependencies eingebunden. Mit bundle install kann man alle gems installationen durchführen und mit rails g gemname:install kann die installation abgeschlossen werden.

Kontrollieren Sie die Tabelle `users` in der Konsole oder mit dem Sqliteman und notieren Sie die Attribute und weitere Beobachtungen:

In der Datenbank gibt es folgende Attribute: User id, email, created_at, updated_at, jedoch kein Attribut für das Passwort

Erklären Sie die Funktionsweise dieses Bildes anhand der CSS-Klassen `core-sprite`, `navbar-brand` und `hide-text`:

Mit der Klasse core-sprite wird definiert, dass dieses Objekt eine Sprite-Grafik verwenden. In den beiden anderen Klassen wird durch koordinaten ein Bild definiert, welches dann im Objekt angezeigt wird. Durch diese Technologie müssen Bilder nur einmal geladen werden.

AB3

Erstellen Sie die Migration und führen Sie diese aus. Notieren Sie die notwendigen Kommandos:

Migration erstellen: rails g migration AddNameToUser
Wenn alle benötigten Elemente in der Migration angegeben sind mit Rails db:migrate migration durchführen.

Bauen Sie die Abfrage in der richtigen View am richtigen Ort ein. Notieren Sie Pfad und Namen der View und die geänderte Codezeile:

In der Datei /app/views/layouts/application.html.erb:

```
<% if current_user %>
  <%=render 'shared/navbar' %>
<%end%>
```

Was fällt Ihnen auf? Stichwort SCSS, CSS und «nesting»?

```
.landing-left {
  .dummy-phone {
  ...
  }
  .screen-shot {
  ...
    .item {
    ...
    }
  }
}
```

1. In welcher Datei haben Sie den Code für den Footer als Partial-View erfasst?
2. Wo wird die Datei mit dem Footer gerendert?
3. Wie lautet der Befehl zum Rendern?

Der Footer wurde in der Datei /shared/_footer.html.erb erfasst und in der new-view von devise mit dem code <%= render „shared/footer" %> geredner.

Im SCSS können Klassen direkt innerhalb einer anderen Klasse angegeben werden und müssen nicht wie beim normalen CSS in einem separaten Tag erfasst werden.

1. In welcher Datei haben Sie den Code für das Dummy-Phone erfasst?
2. Wo wird die Datei mit dem Dummy-Phone für die Views «sign_in» und «sign_up» gerendert?
3. Wie lautet der Befehl zum rendern dieser Datei?

Das Dummyphone wurde in der Datei /devise/shared/_dummy_phone.html.erb erfasst. Das Dummyphone wird im left-landing Div mit dem Befehl
<%= render "devise/shared/dummy_phone" %> gerendert

Notieren Sie die Validierung:

```
validates :name, presence: true
validates :name, length: { maximum: 50}
```