## Projektbericht

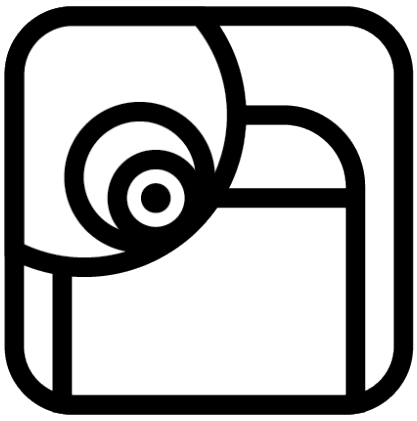
Kurs: Agile Webanwendungen mit Python

Kursleiter: Dipl.-Inf. (FH), Dipl.-Des. Erich Seifert, MA

Wintersememester 22/23 - Hochschule Augsburg

Studiengang: Interaktive Medien

# The Picture Game




**Elias Ginter** Matrikelnummer: 2120054

**Bastian Rögele** Matrikelnummer: 2118370

**Timo Holzmann** Matrikelnummer: 2118987

# Inhaltsverzeichnis

1. Motivation und Anforderungen
2. Planung und Entwurf
3. Entwicklung
4. Inbetriebnahme
5. Fazit und Zukunftsausblick
6. Quellennachweis

# 1. Motivation und Anforderungen

## 1.1 Ideenfindung

Beim Brainstorming über ein umsetzbares Projekt im Modul „Agile Webanwendungen mit Python“ haben wir über verschiedene Ideen nachgedacht und abgewogen. Anfangs wollten wir eine Art WG-Manager mit Finanzplanung und Aufgabenteilung für verschiedene anfallende Tätigkeiten in einer Wohngemeinschaft entwickeln, da eine solche Anwendung definitiv Verwendung in unserem Alltag finden würde. Dann kam uns jedoch die Idee, etwas mit Bildern umzusetzen.

Da wir neben unserem Studienschwerpunkt Informatik in unserer Freizeit gerne gestalterisch tätig sind und unsere Kreativität häufig in der Leidenschaft zur Fotografie ausdrücken, konnten wir uns schnell auf eine Konzeptidee einigen, durch die wir unsere Passion mit anderen teilen und sie mit möglichst einfachen Mitteln an die Fotografie heranführen können. Wir sind der Meinung, Bilder können meist viel mehr erzählen als Worte und wollen Menschen durch unsere Anwendung dazu bringen, ihre Gefühle, Gedanken und die eigene Interpretation zu bestimmten Begriffen möglichst kreativ zu teilen.

## 1.2 Inspiration durch BeReal

Die 2022 erschienene Social Media App BeReal, welche vor allem im letzten Quartal von 2022 in Deutschland immer mehr in den Fokus rückte, basiert auf einem interessanten Konzept, welches in Ansätzen mit unserer Idee verglichen werden kann und einige Ähnlichkeiten aufweist.

BeReal benachrichtigt alle Nutzer einmal pro Tag zu einer wechselnden, vorher nicht bekannten Uhrzeit und fordert sie auf, ein Foto mit dem Smartphone, sowohl mit der Front- als auch der Hauptkamera, aufzunehmen und so den aktuellen Moment möglichst echt und „real“ einzufangen. Sollten Nutzer diesen zweiminütigen Zeitraum verpassen, können die Fotos zu einem späteren Zeitpunkt als „Late-BeReal“, mit der entsprechenden Verspätung gekennzeichnet, nachgereicht werden. Jetzt können die Fotos von Freunden im Feed kommentiert und mit sogenannten „Realmojis“, selbst aufgenommenen, echten Emojis der eigenen Interpretation der typischen Bildschriftzeichen, darauf reagiert werden. So hat man als Nutzer der App nur einmal am Tag die Möglichkeit, seine Inhalte in Form eines Fotos mit der Front- und Hauptkamera hochzuladen und mit anderen zu teilen. Beiträge befreundeter Nutzer können nur eingesehen werden, wenn man selbst ein BeReal hochgeladen hat.

Die App möchte durch fehlende Bearbeitungsoptionen der Bilder und die eingeschränkten Möglichkeiten zum Content-Upload vom typischen, häufig übermäßigen und schädlichen Social Media-Konsumverhalten ablenken, Abhängigkeit reduzieren und den aktuellen Moment möglichst real darstellen.

Die größte Schnittstelle zu BeReal mit unserem Konzept besteht darin, nur einmal am Tag ein Foto aufnehmen und dieses auf die Plattform hochladen zu können, es bestehen allerdings trotzdem deutliche Unterschiede in beiden Ansätzen.

## 1.3 Konzept: The Picture Game

Ähnlich wie bei BeReal sollen die Nutzer unseres Picture Games einmal pro Tag die Möglichkeit haben, ein Foto hochzuladen und in unserer Anwendung mit anderen zu teilen. Hierbei soll jedoch nicht der aktuelle Moment möglichst real eingefangen werden, sondern die eigene Interpretation des Nutzers zu einem vorgegebenen Themenbegriff durch ein Foto mit der Smartphonekamera dargestellt werden.

Ziel der Anwendung ist es, Menschen dazu zu bringen, sich Gedanken über die Umsetzung eines Bildes zum jeweiligen Thema zu machen und diese Idee dann aktiv zu realisieren. Aufgrund der Bewertungsfunktion soll ein kompetitiver Charakter entstehen, der Nutzer motiviert, möglichst gute Bilder aufzunehmen, die sich in Kreativität, Umsetzung und Qualität gegen die anderen Teilnehmer durchsetzen können.

In unserer Anwendung soll es folgende verschiedene Spielphasen geben:

### 1. Einsendungsphase:

Diese Phase soll mit Beginn des Tages immer um 00:00 Uhr starten und mit der Veröffentlichung des Themenbegriffes beginnen. Jetzt haben die Nutzer eine bestimmte Zeitspanne (etwa bis 18:00 Uhr), um ihr jeweiliges Bild zum Thema aufzunehmen und hochzuladen. Es folgt eine Bestätigung über den erfolgreichen Upload des Bildes. Ein Timer soll die verbleibende Zeit bis zum Ablauf der Einsendungsphase visualisieren. Bewusst wollen wir den Nutzern noch keine Einsicht über die eingereichten Fotos der anderen Teilnehmer gewähren, um mögliche Interpretationen oder Ideen zum Thema nicht zu beeinflussen. Hat ein Teilnehmer sein Foto aufgenommen und eingereicht, muss die Zeit bis zur nächsten Spielphase noch vollständig ablaufen.

Die Nutzer sollen jetzt zwischen den Galerien zu früheren Themen und den Profilen anderer Nutzer, auf denen deren Teilnahmen und Einsendungen an früheren Spielen angezeigt werden, navigieren können.

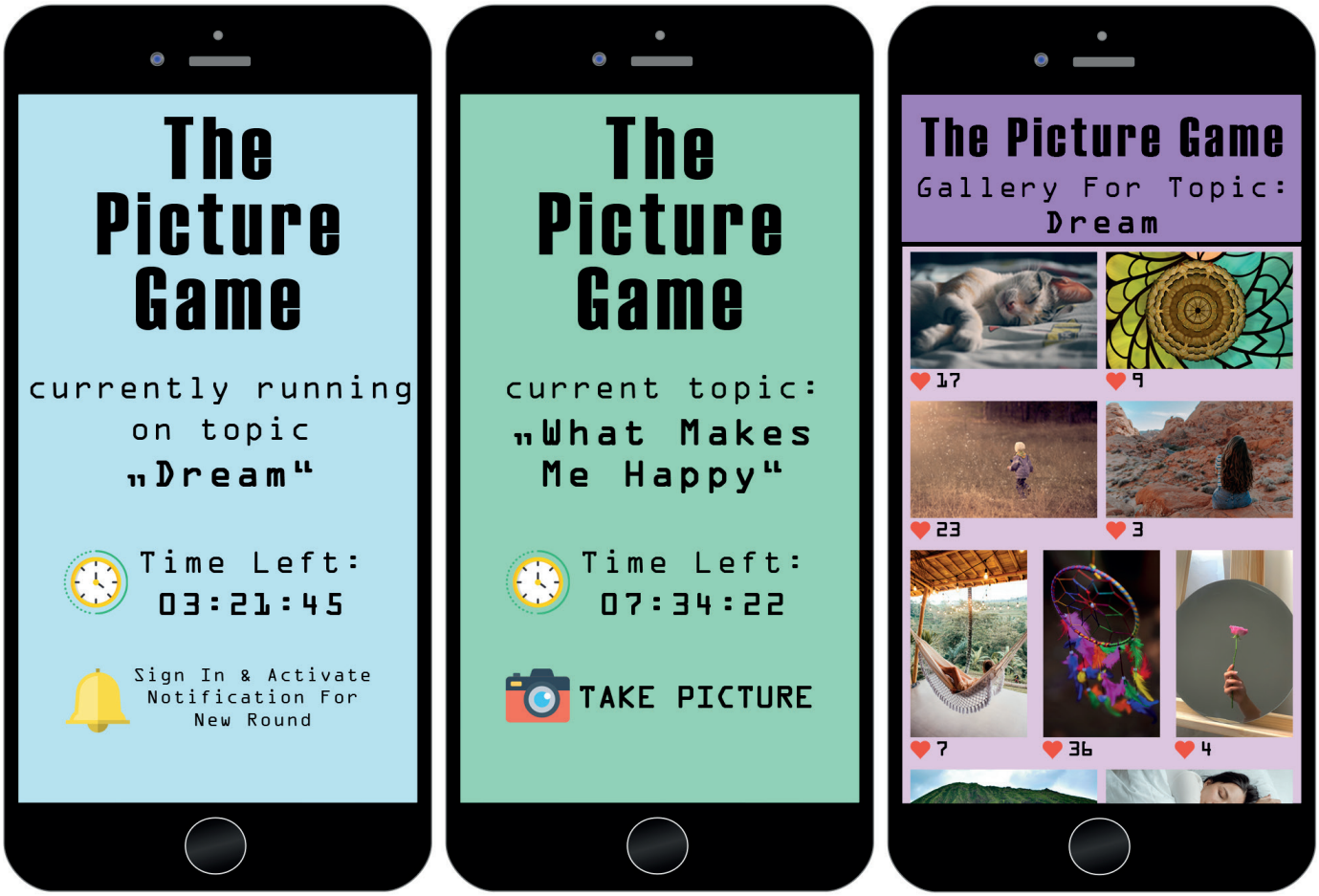
### 2. Bewertungsphase:

Nach Ablauf der Einsendungsphase um 18:00 Uhr folgt die Bewertungsphase bis 21:00 Uhr, in der jeder Teilnehmer eine Galerie zum aktuellen Thema mit den eingereichten Bildern von allen Nutzern sehen kann. Hier soll jeder Nutzer die Möglichkeit haben, ein Bild mit einer Bewertung in Form eines roten Herzens zu bewerten und dementsprechend sein Lieblingsbild wählen zu können.

### 3. Siegerphase:

In der letzten Phase unseres Spiels sollen weder Bilder hochgeladen, noch bewertet werden können. Besucht der Sieger des aktuellen Durchlaufs die Anwendung, wird er über seinen Gewinn informiert und hat die Möglichkeit, aus drei vorgeschlagenen Begriffen aus einem Themenpool ein Thema für das Spiel am nächsten Tag auszuwählen und zusätzlich einen Vorschlag für zukünftige Themen einzureichen. Sollte der Sieger versäumen, die Anwendung nach Ablauf der Bewertungsphase nochmal zu besuchen, wird ein zufälliges Thema für den nächsten Tag und Durchlauf ausgewählt. Alle anderen Benutzer können sich in dieser finalen Phase weiterhin die Galerie des aktuellen Themas anschauen, die Bewertungen einsehen, in früheren Galerien von vorangegangenen Spielrunden und auf den Profilen anderer Nutzer navigieren.

Nachdem wir uns dieses Konzept für unsere Anwendung überlegt haben, folgten die ersten Mockups als grober Entwurf für die einzelnen Spielphasen, um unsere Ansätze erstmals zu visualisieren. Anfangs war es unsere Idee, in eine aktuell laufende Runde als neuer Nutzer nicht einfach einsteigen zu können, weswegen der erste Screen unserer Mockups einen Button zur Anmeldung für eine neue Spielrunde beinhaltete. Aufgrund von Nutzerfreundlichkeit haben wir uns im Laufe der Entwicklung allerdings dafür entschieden, dass man als neuer Nutzer einfach in aktuelle Spielrunden einsteigen kann, ohne sich für die nächste neue Runde anmelden zu müssen.




Erste Screen-Mockups.

Mockup 1: Anzeige für neue Nutzer während einer aktiven Runde.

Mockup 2: Anzeige während Einreichungsphase mit Option zur Bildaufnahme.

Mockup 3: Anzeige der Galerie während der Bewertungsphase.

# 2. Planung und Entwurf

## 2.1 Aufgabenteilung und Organisation

Aufgrund unseres kleinen Entwicklerteams, bestehend aus Elias Ginter, Timo Holzmann und Bastian Rögele sind die Übergänge zwischen den einzelnen Tätigkeiten innerhalb der Gruppe sehr fließend. Es war schwierig, die Aufgabenbereiche konkret aufzuteilen und abzugrenzen und nur bestimmten Personen einen Bereich zuzuweisen. Müssten wir die verschiedenen Aufgaben einzelnen Teammitgliedern zuordnen, so würde die Aufgabenteilung in etwa wie folgt aussehen:

### Elias Ginter

**Main Programmer**

Mit seiner Programmiererfahrung und Kompetenzen in den verschiedensten Bereichen der Softwareentwicklungkonnte Elias in vielen Situationen vernetzt denken und Lösungsansätze für Probleme liefern. Er hatte sofort einen Überblick über unsere Datenstruktur und konnte sich dementsprechend an neue Herausforderungen und Situationen schnell anpassen und adaptieren. Seine Erfahrung und logisches Denkvermögen waren eine große Bereicherung für das gesamte Projekt.

### Timo Holzmann

**Head of Creativity**

Als Photograph hat Timo die meiste Erfahrung im Umgang mit Fotos und Bildern. Da er sich in seinem privaten Umfeld ausgiebig mit der Fotografie beschäftigt, war er für die Grundidee unseres Projektes verantwortlich und maßgeblicher Faktor bei der Planung unseres Konzepts mit konkreten Vorstellungen von den verschiedenen Abläufen und Spielphasen. Seine kreative Ader spiegelt sich in der Gestaltung, unserer Anwendung, die er maßgeblich mitbestimmte, wider.

### Bastian Rögele

**Organizator**

Bastian Rögele war die treibende Kraft bei der Organisation und Verwaltung innerhalb der Gruppe. Zielstrebig plante und organisierte er gemeinsame Treffen und Coding Sessions, während er den Überblick über die noch anstehenden Aufgaben und Probleme bewahrte und diese mit der Gruppe kommunizierte. Hierbei gab er stets Impulse und Vorschläge für Lösungen aus praktischer und technischer Sicht und kombinierte damit seine Kompetenzen im Bereich der Programmierung mit seinen Erfahrungen in der Fotografie.

Da wir durch private Freundschaften regelmäßig im Austausch miteinander sind, fand die Organisation und Planung in vielen Fällen im direkten Dialog bei Treffen in der Hochschule oder bei privaten Zusammenkünften statt. Ansonsten nutzten wir verschiedene Messenger Dienste und trafen uns online über Zoom und Discord.

Wir programmierten immer zusammen und kommunizierten hierbei über Zoom. Ein Gruppenteilnehmer teilte hierfür meist seinen Bildschirm mit den anderen beiden Teammitgliedern und wir konnten so gemeinsam nach Lösungsansätzen suchen und den Code bearbeiten. So konnten zwei Mitglieder durch unterstützende Arbeit und Recherche zu bestimmten Problemen dem Programmierenden helfen und den Coding-Prozess deutlich beschleunigen. Wir hatten den Eindruck, dass wir uns so gegenseitig motivieren konnten, lange an Herausforderungen zu arbeiten, bis alles funktioniert wie gewollt.

## 2.2 Konkreter Entwurf mit Figma-Prototyping

Ursprünglich war es unsere Idee, variierende Zeitspannen für die verschiedenen Themen festzulegen und so Abwechslung in die täglichen Challenges zu bringen. Nach längerem Überlegen und mehrmaligem Überdenken des Konzepts, haben wir uns allerdings dafür entschieden, die Zeiträume für Einreichungsphase, Bewertungsphase und die Siegerphase immer gleich zu wählen und so eine deutlich nutzerfreundlichere Anwendung zu schaffen. So können sich Nutzer an die Abläufe in unserer Anwendung gewöhnen und diese einfacher in ihren Alltag integrieren. Wechselnde Zeiträume von bspw. vier Stunden für ein bestimmtes Thema oder drei Tagen für ein anderes hätten nur zu Verwirrung und Unübersichtlichkeit geführt. Zu lange Einreichungsphasen hätten zudem den Reiz, auf die Bewertungsphase zu warten und dadurch Feedback zum eigenen Foto zu erhalten deutlich reduziert und Nutzer schneller dazu gebracht, das Interesse an unserer Anwendung zu verlieren.

Um unsere Vorstellung von der Anwendung zu konkretisieren, noch besser bildlich zu veranschaulichen und die Abläufe und Navigation vorab einmal durchzuspielen, haben wir einen digitalen Prototyp unserer Anwendung mit Hilfe der Webanwendung „Figma“ erstellt.

Neben den Screen-Mockups für die einzelnen Spielphasen konnten wir hiermit verschiedene Interaktionen durch Buttons mit kleinen Animationen simulieren.

In unserer Figma-Simulation waren somit folgende Screens und Interaktionen möglich:

### 1. Landingpage

mit Login-Funktion und Registrierungsbutton. Hier besteht die Möglichkeit, sich mit einem bestehenden Account einzuloggen und dementsprechend in die aktuelle Spielphase weitergeleitet zu werden oder durch Betätigung des Registrierungsbuttons zu einem Registrierungsformular weitergeleitet zu werden.

### 2. Registrierungsformular

mit Möglichkeit zur Registrierung und anschließender Weiterleitung in aktuelle Spielphase.

### 3. Mockup der Einsendungsphase

Hier kann durch Betätigen des Kamerasymbols auf die Smartphonekamera zugegriffen und anschließend ein Bild zum Upload aufgenommen werden. Durch einen Klick auf den Profil-Button soll das Benutzerprofil des eingeloggten Nutzers aufgerufen werden.

### 4. Simulation des Kamerazugriffs
mit der Möglichkeit, ein Bild durch Betätigen des Kamerabuttons aufzunehmen.

### 5. Screen-Mockup nach erfolgreicher Bildaufnahme
vor dem Upload des Bildes und der Möglichkeit, das Bild hochzuladen.

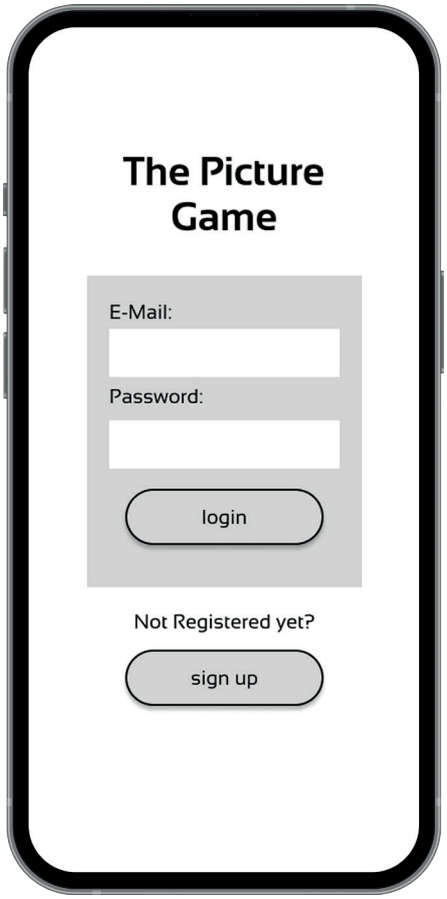
### 6. Bildergalerie für das aktuelle Thema in der Bewertungsphase
mit der Option, ein Bild zu bewerten und durch Klick auf einen Nutzernamen zu seinem Profil weitergeleitet zu werden.

### 7. Profil eines anderen Nutzers
mit Auflistung aller Bilder und Themen, an denen der Nutzer teilgenommen hat. Hier kann durch einen Klick auf ein Thema zu der jeweiligen Themengalerie navigiert werden.

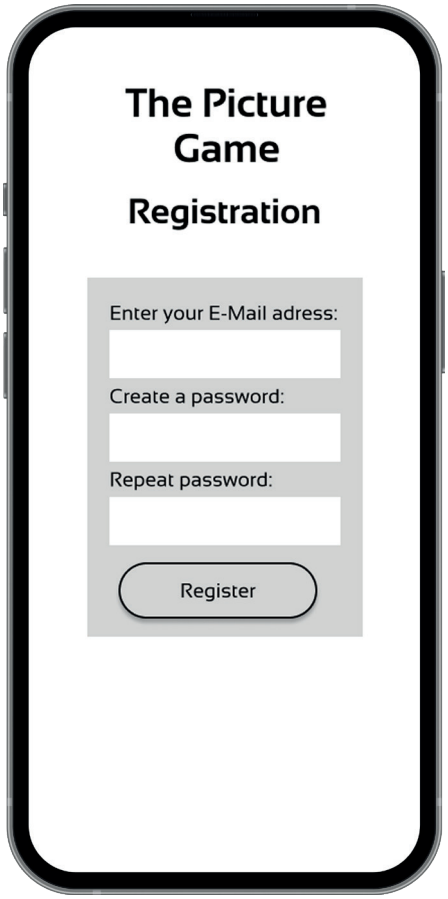
### 8. Profil des angemeldeten Nutzers
in dem durch Klick auf Themenbegriffe zur jeweiligen Galerie navigiert werden kann.

### 9. Themengalerie eines speziellen Themas außerhalb der Bewertungsphase
in der durch Klick auf den Nutzernamen zum jeweiligen Profil oder durch Betätigen des Profilbuttons zum Benutzerprofil des eingeloggten Benutzers navigiert werden kann.

## Figma-Mockups:




1. Figma-Mockup.

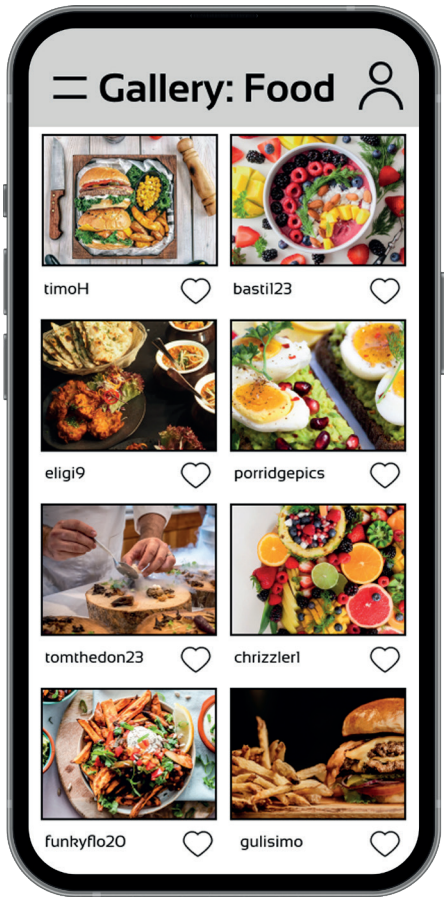



2. Figma-Mockup.

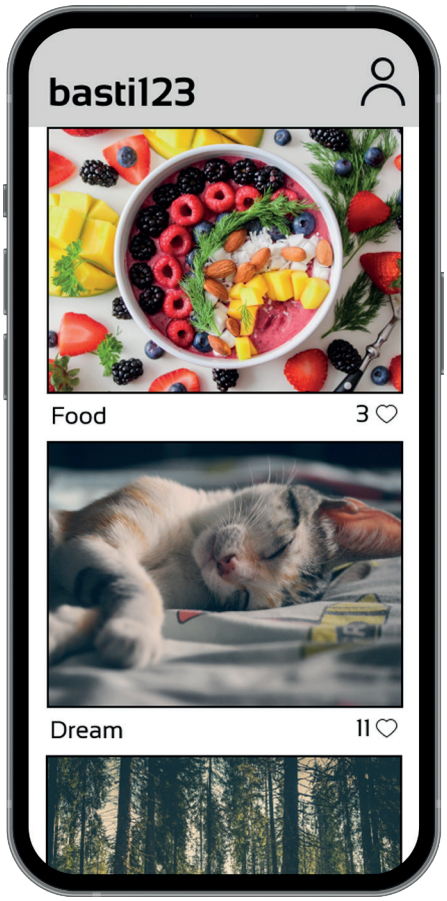

3. Figma-Mockup.


4. Figma-Mockup.

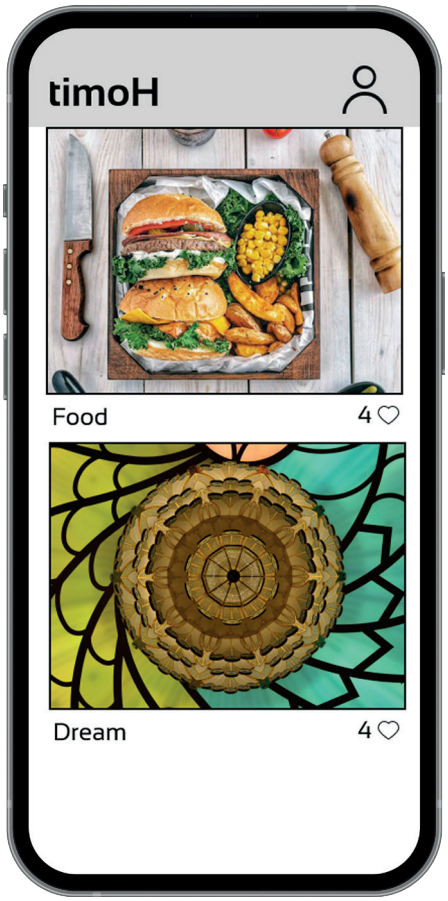

5. Figma-Mockup.



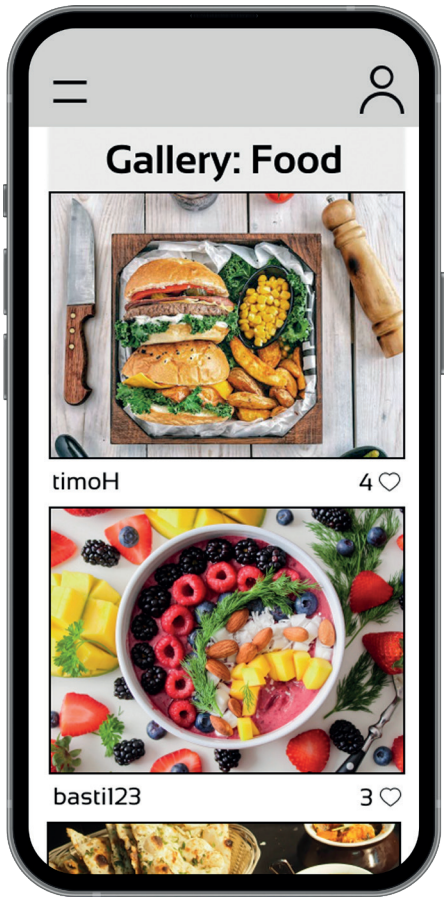

6. Figma-Mockup.




7. Figma-Mockup.




8. Figma-Mockup.




9. Figma-Mockup.

Dieser einfache Prototyp hat uns sehr geholfen, unsere Vorstellungen gegenseitig auszutauschen, uns auf ein realistisches Ergebnis zu einigen, Abläufe genau zu planen und so auf ein gemeinsames Ziel hinzuarbeiten. Im Laufe der Entwicklung haben sich Kleinigkeiten geändert oder kamen hinzu, die Grundstruktur mit den einzelnen Spielphasen und Übergängen zwischen ihnen blieb jedoch größtenteils unverändert.

## 2.3 Entwicklungsumgebung

### PyCharm


Da wir in der Vorlesung die Entwicklungsumgebung „PyCharm“ vom Hersteller JetBrains kennengelernt haben, war es sehr naheliegend diese IDE für unser Python-Projekt zu verwenden.

Zügig wurden wir im Umgang mit PyCharm immer vertrauter und konnten viele Vorzüge der Entwicklungsumgebung genießen. Neben intelligenter Codebearbeitung und -formatierung konnten wir schnell und einfach im Projekt navigieren, mit Hilfe einfacher Funktionen refaktorieren und uns durch die Kompatibilität mit Python, JavaScript, HTML und CSS für alle grundlegenden Funktionen und Anforderungen auf eine Entwicklungssoftware beschränken. Bibliotheken und Python-Webframeworks können durch PyCharm bequem für das Projekt installiert werden und auch das integrierte Datenbanktool erleichterte uns die Arbeit maßgeblich. Nach einmaliger Verknüpfung des Projektes in der IDE mit dem Versionsverwaltungssystem Git konnten wir unsere Versionen einfach organisieren und Änderungen im Team teilen. Die fortlaufende verständliche Benennung unserer Commits erlaubt es uns, auch im Nachhinein noch Überblick über die Versionen unserer Anwendung zu behalten und Änderungen genau nachzuvollziehen.

## 2.4 Bibliotheken

### Django


Für die Backend-Entwicklung haben wir uns für Django entschieden, da das Framework sehr viele administrative Prozesse erleichtert und durch seine Funktionen eine einfache, grundlegende Struktur vorgibt.

Die Erstellung verschiedener Apps ermöglichte es uns, zusammengehörige Elemente zu gruppieren und Funktionen, die den jeweiligen Bereich betreffen innerhalb der einzelnen Apps zu bearbeiten.

Unser Projekt wurde hierfür in die Apps „thepicturegame“, „gallery“ und „accounts“ unterteilt. Thepicturegame ist in unserer Anwendung eine Art übergeordnete Instanz, welche die allgemeine Pfadstruktur festlegt und die untergeordneten Apps gallery und accounts in das Gesamtprojekt mit einbindet.

In gallery werden die Hauptfunktionen unseres Picture Games verwaltet. Alle Elemente und Optionen, die die verschiedenen Spielphasen betreffen, wie der Bilderupload oder die Navigation und Anzeige für verschiedener Nutzer- und Themengalerien wurden hier definiert.

Die accounts-App behandelt alle für die Nutzerverwaltung relevanten Elemente wie Login, Logout und Registrierung.

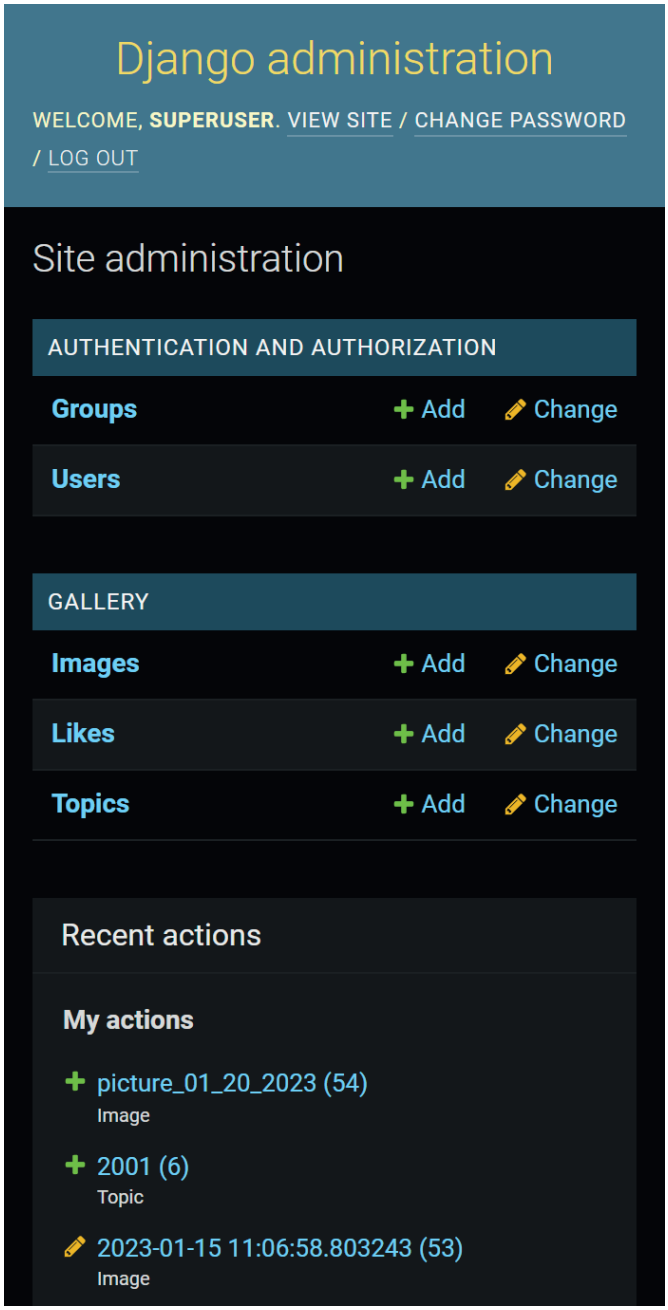
Mit Views bietet Django eine weitere Möglichkeit der Modularisierung, indem Bereiche innerhalb einer App separat definiert werden können. So konnten wir in der gallery-App verschiedene Views für einzelne Spielphasen, wie die Einreichungsphase mit Upload-Funktion und den dazugehörigen, notwendigen Daten festlegen. Die Bereiche für Nutzerprofile mit Bildergalerien des jeweiligen Nutzers und Themengalerien zu einzelnen Themenbegriffen konnten wir klar voneinander abgrenzen und in unserer Haupt-View namens „home“ durch Fallunterscheidungen zur korrekten View, abhängig von der aktuellen Spielphase und anderen Input-Attributen weiterleiten.

Die Views in der accounts-App definieren das Registrierungs- und Loginformular.

Die Weiterleitung von View zu View und damit von HTML-Seite zu HTML-Seite ermöglicht Django mit wiederum separat abgegrenzten URL-Routing-Pages für jede App, was uns eine flexible Navigation durch die einzelnen Programmelemente gewährleistet.

Durch integriertes Datenbankmanagement und Django-Models konnten wir unsere Datenbankobjekte ebenfalls mit Hilfe des Frameworks erstellen und verwalten. Die automatisch generierte API für den Datenbankzugriff ermöglichte uns die Verwaltung und Organisation der Daten innerhalb des Projektes.

Eine weitere Möglichkeit, auf die Daten unseres Spiels zuzugreifen, bietet das automatische `Django-Admin-Interface`. Hiermit konnten wir nach Autorisierung als Administrator in der laufenden Webanwendung auf die Daten in unserer Datenbank zugreifen, diese verwalten und modifizieren.




`Django-Admin-Interface` mit Benutzer- und Datenverwaltung.

Mit vorgefertigten Forms erleichtert Django die Eingabe von Benutzerdaten, es besteht allerdings die Möglichkeiten die Forms individuell anzupassen, wodurch beispielsweise bei unserer Upload-Form alle initialen Input-Felder durch ein Kamerasymbol ersetzt wurden. Beim Benutzerlogin und der Registrierung griffen wir ebenfalls auf Django-Forms zurück.

Django bietet ein integriertes User-Authentication-System, das spezifischen Nutzern bzw. Nutzergruppen Zugriff zu bestimmten Handlungen oder Anwendungsbereichen gewährt oder verbietet. So konnten wir als

Administratoren bestimmte Bereiche wie das Admin-Interface einsehen und mit Hilfe des login_required-Decorators Funktionen nur für authentifizierte, eingeloggte Benutzer zugänglich machen.

## Pillow


Da Bilder und damit Bildobjekte Hauptbestandteil unserer Daten sind, haben wir uns für Pillow als Bibliothek für den Umgang mit den Bildobjekten entschieden.

Unsere Bildobjekte sind vom Django-Dateifeld „ImageField“, wodurch beim Upload der Bilder wirklich nur Bilddateien zugelassen werden. Verwendet man dieses Datenfeld, muss Pillow in der Anwendung installiert sein.

Pillow bietet mit zahlreichen Funktionen und Anpassungsoptionen viele Möglichkeiten, die Bildobjekte zu manipulieren und zu bearbeiten.

Mit Hilfe der Image.crop()-Methode von Pillow konnten wir ein einheitliches Bildformat für alle hochgeladenen Bilder realisieren. Ohne diese Manipulation der Bilder hätten verschiedene Smartphones von unterschiedlichen Herstellern durch die interne Kamera Fotos in den verschiedensten Formaten generiert und so eine unübersichtliche Bildergalerie zur Folge gehabt. Wir haben uns dafür entschieden, die Bilder unmittelbar auf ein quadratisches Format zu beschneiden und so einheitliche Bildergalerien erzeugen zu können.

# 3. Entwicklung

## 3.1 Datenstruktur

Für unsere Anwendung mussten wir uns für die einzelnen Datenobjekte Attribute und Beziehungen überlegen. Unsere Datenbank besteht aus den Datenmodellen „topic“, „user“, „image“ und „like“. Die einzelnen Datenobjekte haben hierbei folgende Eigenschaften:

**`topic:`**
**`name:`** Name des Themas
**`date:`** Datum, an dem das Thema für unser Spiel verwendet wird
**`validated:`** gibt an, ob das Thema bereits durch einen Administrator validiert wurde.

**`image:`**
**`name:`** Name des Bildes
**`file:`** Dateityp des Bildes
**`score:`** Anzahl an Bewertungen des Bildes
**`user:`** Benutzer, der das Bild erstellt hat
**`topic:`** Thema, zu dem das Bild gehört
**`winner:`** gibt an, ob das Bild der Gewinner der Spielrunde ist

Ein image wird eindeutig durch user und topic definiert.

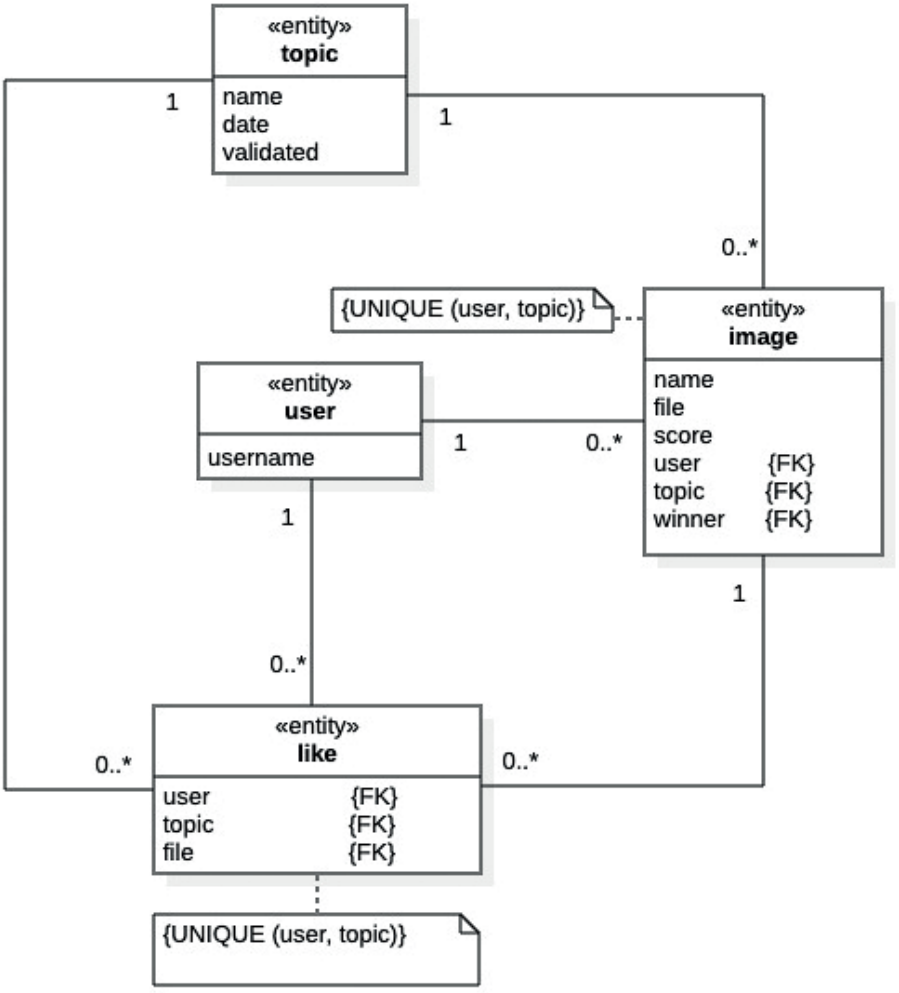
**`like:`**
**`user:`** Benutzer, der die Bewertung abgegeben hat
**`topic:`** Thema des Like-Objekts
**`file:`** Das Bild-Objekt, dem das Like-Objekt zugeordnet ist

Ein like wird eindeutig durch user und topic definiert.

**`user:`**
Hier verwenden wir das von Django bereitgestellte User-Datenmodell

**`username:`** Benutzername des Benutzers

Die Beziehungen und Vielfachheiten zwischen den Datenmodellen können der folgenden Grafik entnommen werden.

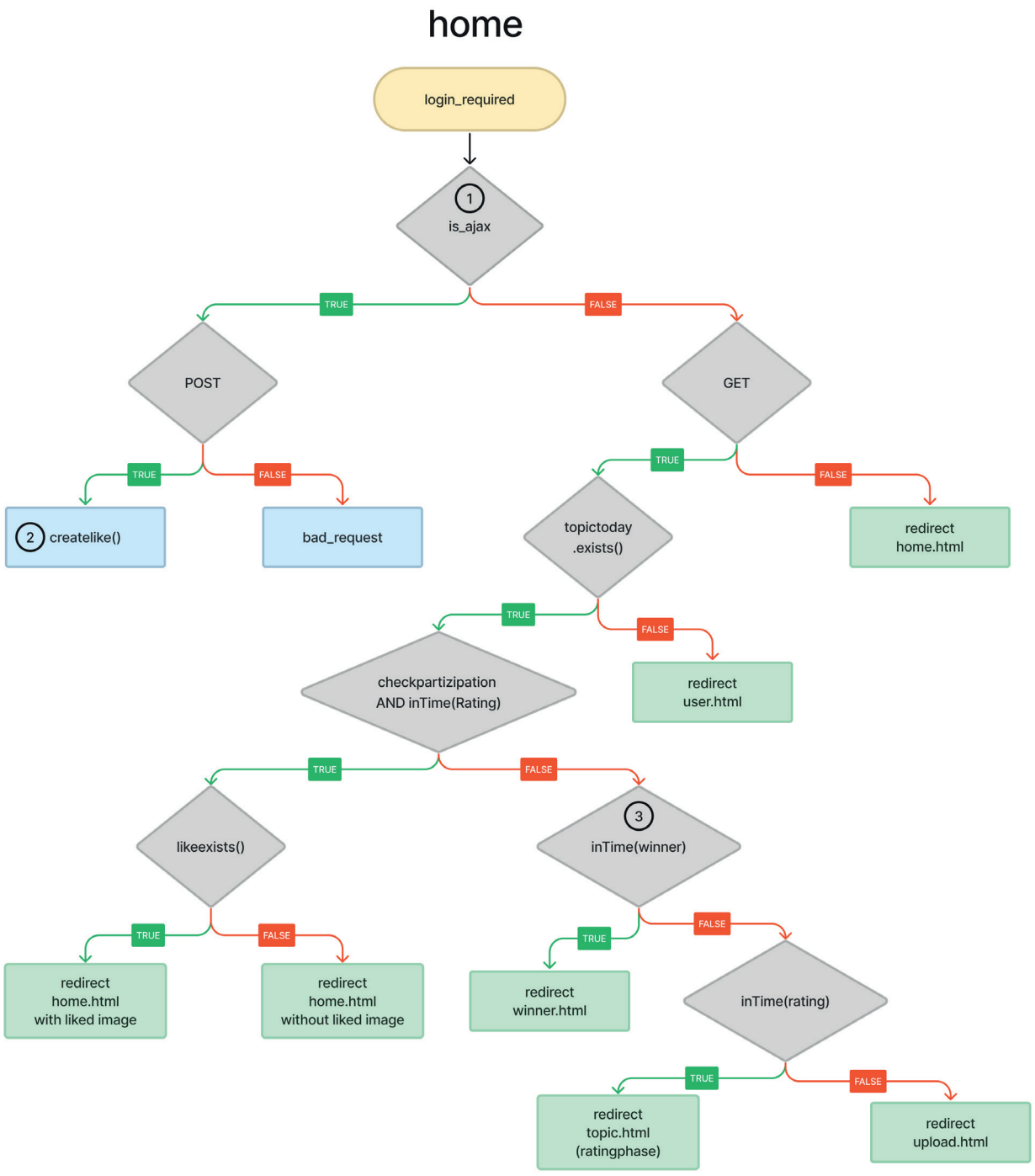



Datenmodelle und Beziehungen unseres Picture Games

## 3.2 Views

Die Views stellten in unserem Projekt eine große Herausforderung dar, da wir uns Lösungen für komplexe Abfragen überlegen mussten, um korrekte Informationen für die richtige Anzeige mitzugeben. Abhängig von der Spielphase und anderen Input-Attributen, werden in unserer Anwendung verschiedene Elemente und Funktionen bereitgestellt und der Nutzer dementsprechend zur jeweils richtigen html-Seite weitergeleitet. Unsere hauptsächlich verwendeten Views sind „home“, „upload“ und „winner“. Zur besseren Übersichtlichkeit haben wir für diese Views Flowcharts erstellt, die im Folgenden dargestellt werden. Anhand dieser Flowcharts werden einige, komplexere Funktionen erläutert. Diese sind in den Flowcharts durch eine Zahl gekennzeichnet.

### Home




Flowchart der Home-View

## 1. is_ajax

Likeobjekte werden durch Herzen visuell umgesetzt.

Die HTML-Bildelemente der Herzen enthalten folgende Attribute:

**imageusername**:
Name des Users, der das Bild erstellt hat

**topicname**:
Name des Themas, dessen Bilder angezeigt werden.

**username:**
Name des Users, der momentan eingeloggt ist.

**url:**
Die url, an die bei Klick der Post Request erfolgt.

Bei Klick auf ein Herz wird die Callbackfunktion des Eventlisteners, welcher auf dem Objekt initialisiert wurde, ausgelöst. Diese sendet einen POST AJAX Request an die Home-View.

Mittels Fetch-Methode wird ein Form-Data Objekt, welches die Attribute des Event-Targets(Herz) enthält an die Home-View gesendet, wo die Datenbank angepasst wird.

Dank Ajax kann dies im Hintergrund passieren, ohne dass die Seite dafür neu geladen werden muss.

## 2. create_like (user_name, topic_name, image_user_name)

**user_name:**
Name des Users, der eingeloggt ist.

**topic_name:**
Thema des Bildes für das ein Like erstellt werden soll

**image_user_name:**
Name des Benutzers, der das Bild erstellt hat, für das ein Like erstellt wird

Zunächst werden die mit den Parametern verankerten Objekte aus der Datenbank geholt und überprüft, ob diese überhaupt existieren.

Das Image-Objekt, das bewertet wurde, ist durch den zugehörigen User und das Topic eindeutig definiert. Dafür werden der image_user_name und der topic_name abgefragt.

Nun gibt es zwei Fälle:

1. Es existiert bereits ein Like-Objekt von dem ausführenden User zum gegebenen Topic.

In diesem Fall wird das bestehende Like-Objekt verändert, indem das Image-Attribut des Like-Objekts an das bewertete Image angepasst wird.

2. Es existiert noch kein Like-Objekt von dem ausführenden User zum gegebenen Topic.

Es wird ein neues Like Objekt erstellt.

Anschließend wird das Attribut „score“ des bewerteten Bildes angepasst. Hierfür wird für beide Fälle die Funktion **set_score()** aufgerufen, welche die Anzahl der Like-Objekte des übergebenen Bildes in der Datenbank zählt und dann das Attribut „score“ schreibt.

Die Funktion gibt True zurück, wenn ein Like-Objekt erfolgreich initialisiert oder geändert wurde, ansonsten wird False ausgegeben.

### `set_score(image)`

**`Image:`** übergebenes Bild-Objekt, dessen score gesetzt werden soll.

Die Methode nimmt alle Like-Objekte aus der Datenbank, die für das übergebene Bild-Objekt existieren. Anschließend wird die Anzahl der Like-Objekte im score-Attribut des übergebenen Bildes gespeichert.

```python
def create_like(user_name, topic_name, image_user_name):
    if check_user_exists(user_name) and check_topic_exists(topic_name) and check_user_exists(image_user_name):
        current_user = User.objects.filter(username=user_name)
        current_topic = Topic.objects.filter(name=topic_name)
        image_user = User.objects.filter(username=image_user_name)

        # check whether the liked image exists
        if check_image_exists(current_topic[0], image_user[0]):
            liked_image = Image.objects.get(user=image_user[0], topic=current_topic[0])

            # If there exists a like-object already, the image attribute will be adapted,
            # else a new like object will be created
            if check_like_exists(current_topic[0], current_user[0]):
                current_like = Like.objects.get(topic=current_topic[0], user=current_user[0])
                before_liked_image = current_like.image
                current_like.image = liked_image

                # saving modified like object to the db
                current_like.save()
                # setScore new Image
                set_score(before_liked_image)
                set_score(liked_image)

                return True
            else:
                # incrementing score of the liked-image and create Like object in db
                Like.objects.create(user=current_user[0], topic=current_topic[0], image=liked_image)
                set_score(liked_image)
                return True
        else:
            return False
    else:
        return False
```

Screenshot der create_like()-Methode

```python
def set_score(image):
    image_likes = Like.objects.filter(image=image)
    if image_likes.exists():
        image.score = len(image_likes)
        image.save()
    else:
        image.score = 0
        image.save()
```

Screenshot der set_score()-Methode

### 3. in_time(start_time, end_time)

**start_time:** übergebener Startzeitpunkt

**end_time:** übergebener Endzeitpunkt

Es wird zuerst der jetzige Zeitpunkt in der Variable „now“ gespeichert.

Danach wird ein Start- und Enddatum gesetzt. Hierfür wird der jetzige Zeitpunkt jeweils mit dem übergebenen Start- und Endzeitpunkt kombiniert.

In der Variable „delta_start“ wird der Abstand vom Startzeitpunkt zum aktuellen Zeitpunkt berechnet.

In der Variable „delta_end“ wird der Abstand vom jetzigen Zeitpunkt zum Endzeitpunkt berechnet.

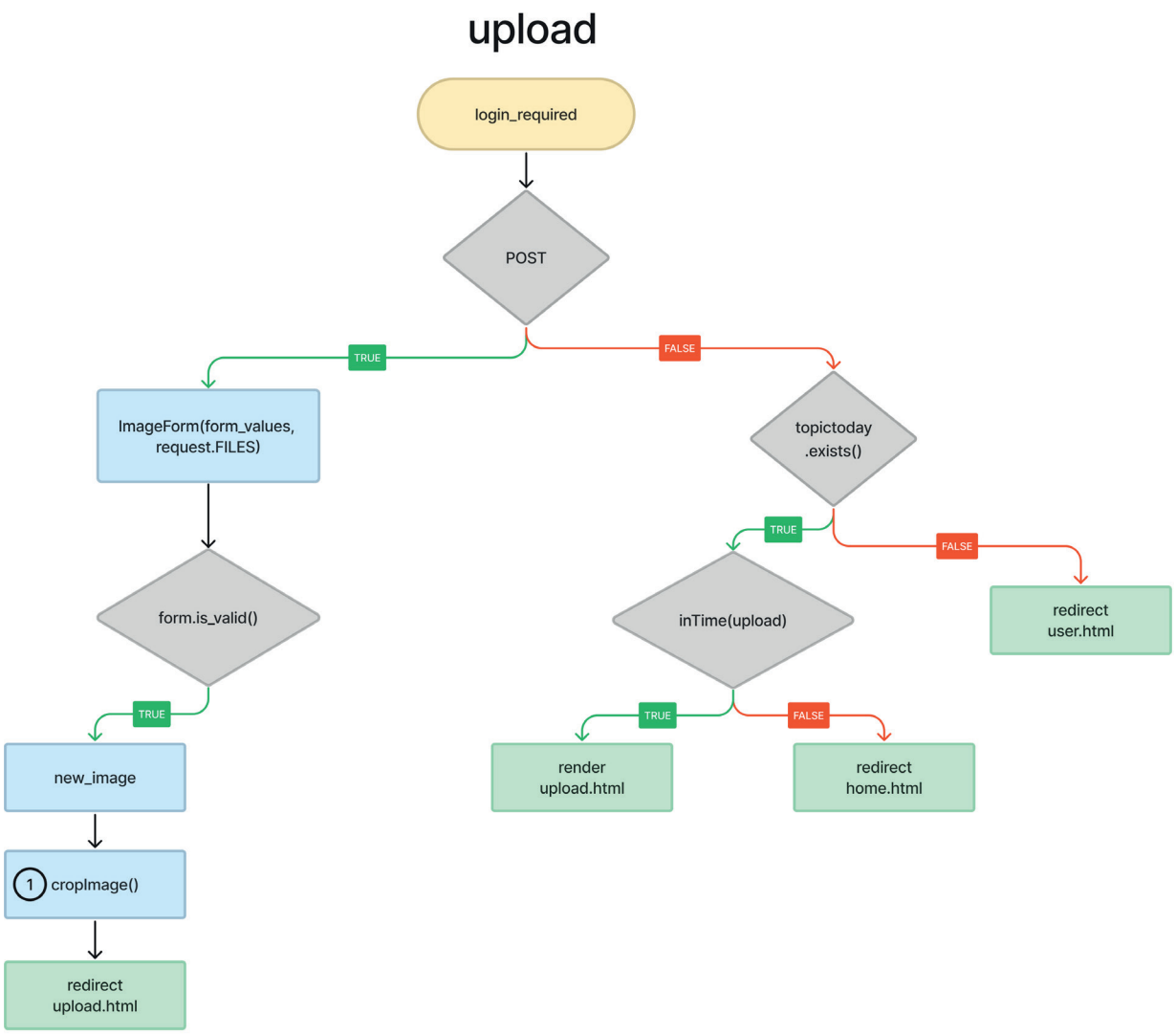
Der aktuelle Zeitpunkt befindet sich genau dann im übergebenen Zeitintervall, wenn beide Abstände > 0 sind.

In diesem Fall gibt die Funktion True zurück. Andernfalls wird False zurückgegeben.

```
def in_time(start_time, end_time):
    now = datetime.now()
    start_date = datetime.combine(now, start_time)
    end_date = datetime.combine(now, end_time)
    delta_start = now - start_date
    delta_end = end_date - now
    return delta_start.total_seconds() > 0 and delta_end.total_seconds() > 0
```

Screenshot der in_time()-Methode

# Upload




Flowchart der Upload-View

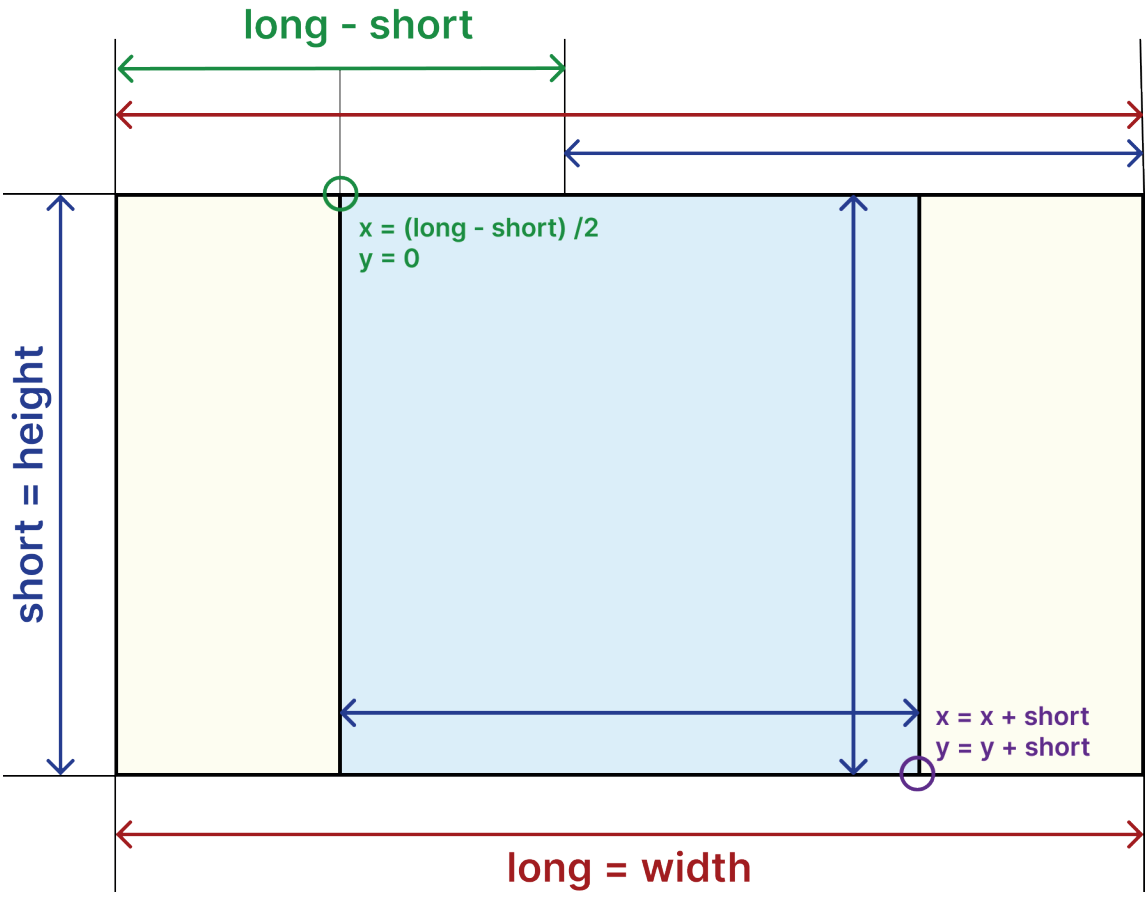
## 1. crop_image(image_file)

**image_file:** übergebenes Bildobjekt, das beschnitten werden soll.

Das übergebene Bildobjekt wird mittels der Python Imaging Library (PIL) zuerst in der Variable „image“ geöffnet.
Wenn das übergebene Bildobjekt im RGBA-Format übergeben wird, wird es zu RGB angepasst.
Es wird die Methode „exif_transpose“ aufgerufen, die das exif-Tag des Bildes erkennt und die Ausrichtung des Bildes entsprechend setzt.
In der Variable „short_side“ wird die kürzere Seite des Bildes gespeichert.
Je nachdem, welche Seite kürzer ist, wird der linke obere Eckpunkt der längeren Seite, welcher den Beschnitt festlegt, berechnet.
Das übergebene Bildobjekt wird mittels der image.crop()-Funktion beschnitten. Der rechte untere Eckpunkt ergibt sich hierbei aus den Koordinaten des linken oberen Eckpunktes + der kurzen Seite.Das Bildobjekt wird mittels byte_stream in ein InMemoryUploadedFile konvertiert und kann so als „ImageField“ abgespeichert werden. Die Funktion gibt dieses InMemoryUploadedFile zurück.

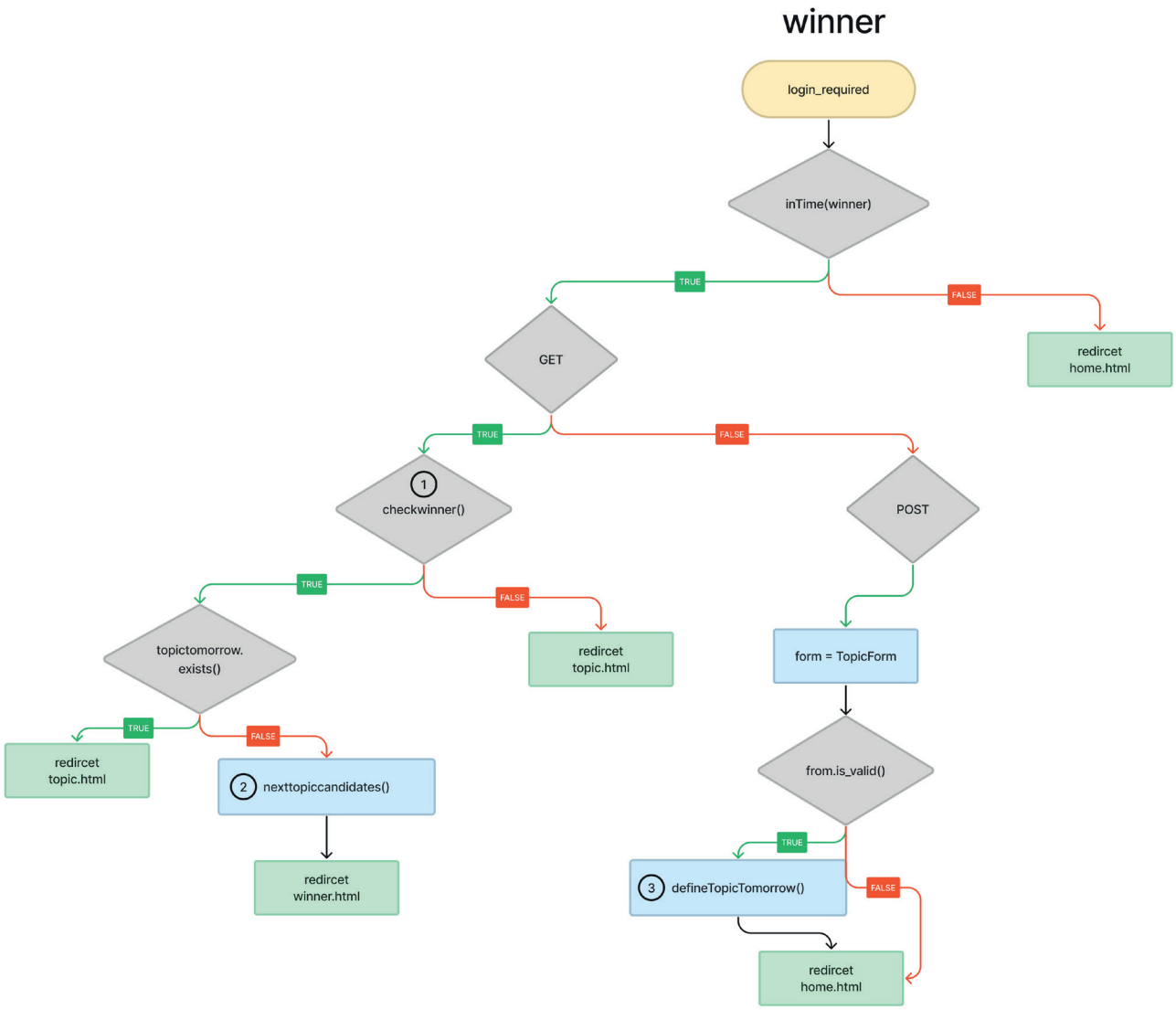
```python
def crop_image(image_file):
    image = PIL.Image.open(image_file)

    if image.mode == 'RGBA':
        image = image.convert('RGB')

    # detect exif tag of image and sets orientation
    image = ImageOps.exif_transpose(image)
    width, height = image.size
    # detect shorter side which defines length of square
    short_side = {True: width, False: height}[width < height]
    # detect format of the uploaded image to calculate left upper corner point of square
    if short_side == width:
        long = height
        y = (long - short_side)/2
        x = 0
    else:
        long = width
        x = (long - short_side)/2
        y = 0

    # convert Pillow Image Object into MemoryUploadedFile by using a byte stream
    byte_stream = BytesIO()
    image.crop((x, y, x + short_side, y + short_side)).save(byte_stream, format='JPEG')
    byte_stream.seek(0)
    django_image_file = InMemoryUploadedFile(byte_stream, 'ImageField', image_file.name, 'image/jpeg',
                                             sys.getsizeof(byte_stream), None)
    return django_image_file
```

Screenshot der in_time()-Methode




Bildliche Veranschaulichung zur Berechnung des Beschnitts

# Winner




Flowchart der Winner-View

## 1. check_winner(user, topic)

**user:** Benutzer, für den geprüft werden soll, ob er der Gewinner ist

**topic:** Thema, für das der Gewinner überprüft werden soll

Die Funktion check_winner überprüft, ob es sich beim übergebenen user um den Gewinner des übergebenen topics handelt. Existieren keine Bilder zu diesem Thema, wird False ausgegeben. Andernfalls wird überprüft, ob bereits ein Gewinner existiert und es sich beim übergebenen user um den Gewinner handelt.

Falls kein Gewinner existiert, wird mit der Funktion **set_winner** ein Gewinner bestimmt.

Die Funktion führt sich rekursiv erneut aus, um zu überprüfen, ob es sich beim übergebenen Nutzer um den Gewinner handelt.

```python
def check_winner(user, topic):
    images = Image.objects.filter(topic=topic)
    if images.exists():
        winner = Image.objects.filter(topic=topic, winner=True)

        # if no winner exists, set winner for current topic
        if winner.exists():
            return winner[0].user == user
        else:
            set_winner(topic)
            return check_winner(user, topic)
    else:
        return False
```

Screenshot der check_winner()-Methode

## set_winner(topic)

**topic:** übergebenes Thema, für das der Gewinner gesetzt werden soll.

Es werden alle Bilder vom übergebenen Thema mit Hilfe der reduce-Funktion aus dem „functools“-Modul verglichen. Sie iteriert über die Liste aller Bilder des Themas und reduziert sie anhand der Höhe des Scores. Übrig bleibt ein Bildobjekt mit maximalem Score.

Auf dessen Basis werden alle weiteren Bilder des Themas mit dem gleichen Score aus der Datenbank geholt.

Aus dieser Auswahl wird ein Gewinner zufällig bestimmt.

Das winner-Attribut des entsprechenden Bildes wird auf True gesetzt.

```python
def set_winner(topic):
    images = Image.objects.filter(topic=topic)

    # reduce image set to candidate with the highest score
    candidate_img = reduce(lambda candidate, image: image if image.score > candidate.score else candidate, images)

    # if there are more than one image with the highest score, randomly pick one as winner
    winner_list = Image.objects.filter(topic=topic, score=candidate_img.score)
    winner = winner_list[randint(0, winner_list.count() - 1)]
    winner.winner = True
    winner.save()
```

Screenshot der set_winner()-Methode

## 2. next_topic_candidates()

Die Funktion next_topic_candidates gibt eine Liste mit drei Themen zurück.

Für die Auswahl der Kandidaten werden die Themen bevorzugt, welche bisher noch kein Datum enthalten aber schon durch einen Admin validiert wurden.

Diese werden aus der Datenbank gefilter und in der Variable undated_topics hinterlegt.

In der Variable dated_topics sind die Themen hinterlegt, welche bereits ein Datum enthalten.

Es gibt **3 Fälle:**

1. Die Anzahl der undated_topics ist >= 3:
In diesem Fall werden für die Kandidaten nur Elemente aus der undated_topics Liste

gewählt.

2. Die Anzahl der undated_topics ist > 0 und < 3:
In diesem Fall werden die undated_topics verwendet, die existieren.

Für die verbleibenden Kandidaten wird aus den dated_topics gewählt.

3. Es existieren keine undated topics:
Die Auswahl der Kandidaten erfolgt ausschließlich aus den undated Topics.

Die explizite Erstellung der Liste erfolgt mittels der Methode: **get_random_topic_subset(length, set)**.

Diese erstellt eine randomisierte und duplikatlose Auswahl einer übergebenen Menge mit der Länge „Anzahl“.

## 2. get_random_topic_subset (length, set)

Die Funktion wählt aus der Liste an übergebenen Themen ein zufälliges Thema aus und fügt es dem subset hinzu, bis das Subset die übergebene Länge hat. Das zufällig ausgewählte Thema wird hierbei in jedem Schritt aus der Liste an übergebenen Themen entfernt.

Die Funktion gibt am Ende das subset zurück.

```
next_topic_candidates():

undated_topics = Topic.objects.filter(date=None, validated=True)
dated_topics = Topic.objects.filter(date__isnull=False, validated=True)
if undated_topics.exists():
    if undated_topics.count() > 2:
        return get_random_topic_subset(3, undated_topics)
    else:
        candidates = list(undated_topics)
        candidates.extend(get_random_topic_subset(3 - undated_topics.count(), dated_topics))
        return candidates
else:
    return get_random_topic_subset(3, dated_topics)
```

Screenshot der next_topic_candidates()-Methode

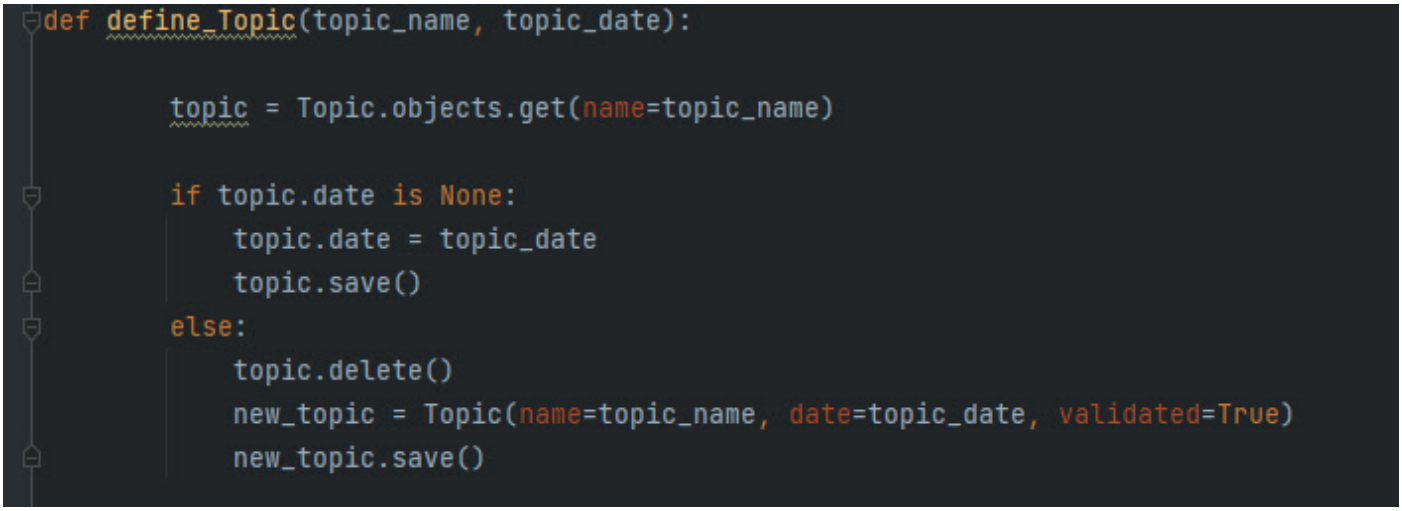
### 3. define_topic(topic_name, topic_date)

Die Methode setzt das Datum für das Thema mit dem übergebenen topic_name.

Existiert bereits ein Datum für das Thema, wird es aus der Datenbank entfernt.

Hierbei werden alle Bild-Objekte und Like-Objekte, die für das Thema angelegt wurden, gelöscht. Anschließend wird ein neues Topic-Objekt mit dem übergebenen topic_name und topic_date erstellt.

Existiert kein Datum für das Thema, wird das date-Attribut auf topic_date gesetzt.



```
def define_Topic(topic_name, topic_date):

        topic = Topic.objects.get(name=topic_name)

        if topic.date is None:
            topic.date = topic_date
            topic.save()
        else:
            topic.delete()
            new_topic = Topic(name=topic_name, date=topic_date, validated=True)
            new_topic.save()
```

Screenshot der define_Topic(topic_name, topic_date)

# 4. Inbetriebnahme

Nachdem das Projekt auf einem Rechner lokal installiert wurde, müssen die Bibliotheken Django und Pillow installiert werden.

Jetzt kann mittels **`cd thepicturegame`** in den Ordner des Picture Games navigiert werden. Mittels **`python manage.py makemigrations`** und **`python manage.py migrate`** muss die Datenbank migriert werden.

Jetzt kann der Server mit dem Befehl **`python manage.py runserver 0.0.0.0:8000`** gestartet werden.

Wenn sich das Smartphone und der hostende Computer im selben Netzwerk befinden, kann die Seite im Browser des Smartphones durch folgende URL aufgerufen werden:

**`[IPv4-Adresse des Computers]:8000`**

Jetzt können alle im nächsten Abschnitt beschriebenen Funktionen verwendet werden.

Die Anwendung kann natürlich auch im Browser auf dem Computer durch den selben Link aufgerufen werden, sie wurde allerdings für Smartphones gestyled. Wenn man die Website am Computer ausprobieren möchte, sollte man darum in den Entwicklertools auf eine Smartphone-Ansicht wechseln.

Um die Admin-View aufzurufen und die Datenbank anzupassen, muss folgende URL eingegeben werden: **`[IPv4-Adresse des Computers]:8000/admin`**

Um sich hier einloggen zu können, können die Nutzerdaten unseres Superusers verwendet werden:

Username: superuser

Passwort: thepicturegameadmin

Möchte man die einzelnen Spielphasen testen, müssen folgende globale Variablen direkt am Anfang der Datei views.py anders eingestellt werden:

RATING_START_TIME = time(HH, MM, SS)

RATING_END_TIME = time(HH, MM, SS)

### Link zum Projekt:

https://gitlab.informatik.hs-augsburg.de/eligi9/thepicturegamews2022

# 5. Fazit und Zukunftsausblick

## 5.1 Fazit und erreichte Ziele

Wir haben alle unserer ursprünglich festgelegten Anforderungen in unserer finalen Anwendung realisieren können. Die Anwendung ist als Handyanwendung ausgelegt und wurde dementsprechend für die Verwendung am Smartphone gestyled. Wir konnten die Logik und den Spielfluss unseren Vorstellungen entsprechend implementieren.

Im Picture Game haben die Nutzer dementsprechend folgende Möglichkeiten:

### 1. Landingpage

Hier können sich Benutzer einloggen und werden dann in die aktuelle Spielphase weitergeleitet, oder registrieren, sofern noch kein Account vorhanden ist.

### 2. Registrierungsformular

Unser Registrierungsformular bietet die Möglichkeit, sich mit Benutzernamen und Passwort für die Anwendung zu registrieren.

### 3. Einreichungsphase

In der Einreichungsphase können die Nutzer das aktuelle Thema und einen Timer, der die verbleibende Zeit zum Upload des Bildes anzeigt, sehen. Das Kamerasymbol bietet die Möglichkeit, die Smartphonekamera aufzurufen und so ein Foto zum aktuellen Thema einzureichen.

### 4. Kamerazugriff

Durch den Kamerazugriff auf die Smartphonekamera, kann ein Bild aufgenommen und für den Upload ausgewählt werden.

### 5. Bestätigungsansicht

Bestätigungsansicht nach erfolgreichem Upload des Bildes in der Einreichungsphase mit verbleibender Zeit bis zur Bewertungsphase.

### 6. Bewertungsphase

Nach Ablauf des Timers in der Einreichungsphase erfolgt eine automatische Weiterleitung zur Bewertungsphase, in der alle eingereichten Bilder von allen Nutzern angezeigt werden. Hier kann durch Auswahl des Herzens unter einem Bild das Favoritenbild bewertet und durch Tap auf einen Benutzernamen zum Benutzerprofil des jeweiligen Nutzers navigiert werden.

### 7. Nutzerprofil

Im Profil des Nutzers, sowie in den Profilen der anderen Nutzer, werden alle Bilder mit den dazugehörigen Themen aufgelistet, an denen der Nutzer bisher teilgenommen hat. Klickt man hier auf einen Themenbegriff, wird man zur Themengalerie des Begriffs weitergeleitet und kann die Bilder aller Nutzer, die dazu eingereicht wurden, einsehen.

### 8. Themengalerie

Die Themengalerie funktioniert ähnlich wie die Galerie in der Bewertungsphase. Hier werden alle eingereichten Bilder zum Thema inklusive des jeweiligen Benutzers angezeigt. Ein Tap auf den Benutzernamen navigiert zur Benutzergalerie des Nutzers. Die Bewertung der Bilder ist hier, außerhalb der Bewertungsphase, nicht mehr möglich.

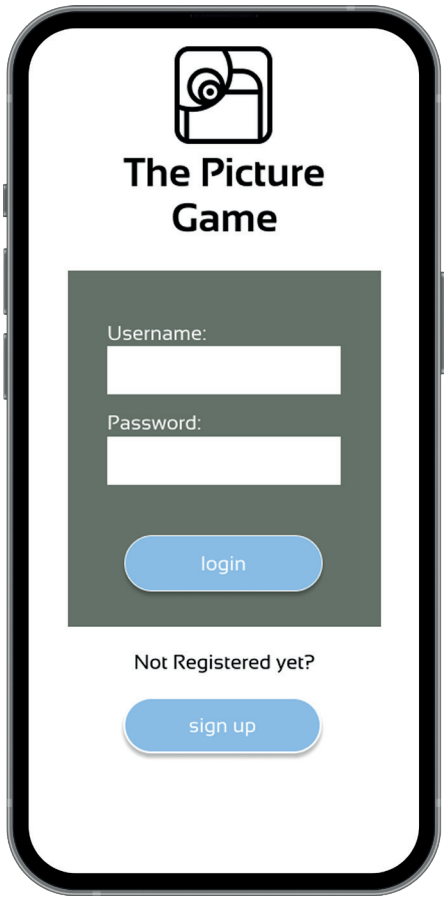
### 9. Gewinneransicht

Nach Ablauf der Bewertungsphase gibt es für einen speziellen Nutzer, den Gewinner der aktuellen Spielrunde, eine besondere Gewinneransicht. Hier kann der Nutzer aus drei Vorschlägen das Thema für den nächsten Tag festlegen und selbst einen Vorschlag für zukünftige Themen einreichen. Dieser Vorschlag muss von einem Administrator noch verifiziert und validiert werden, damit keine unangebrachten Vorschläge in der Themenauswahl landen.

Nach erfolgreicher Auswahl durch den Gewinner wird dieser, genau wie alle anderen Benutzer auf die Galerie des aktuellen Themas weitergeleitet. Bis zum nächsten Thema am nächsten Tag besteht jetzt noch die Möglichkeit, zwischen alten Bildergalerien, Nutzerprofilen und dem eigenen Profil hin und her zu navigieren.

Durch Klick auf unser Logo im linken oberen Bereich des Bildschirms gelangt der Nutzer immer zur aktuellen Spielphase. Ein Klick auf den Profilbutten am rechten oberen Bildrand navigiert zum Profil des eingeloggten Nutzers.

## Screenshots aus der finalen Anwendung:



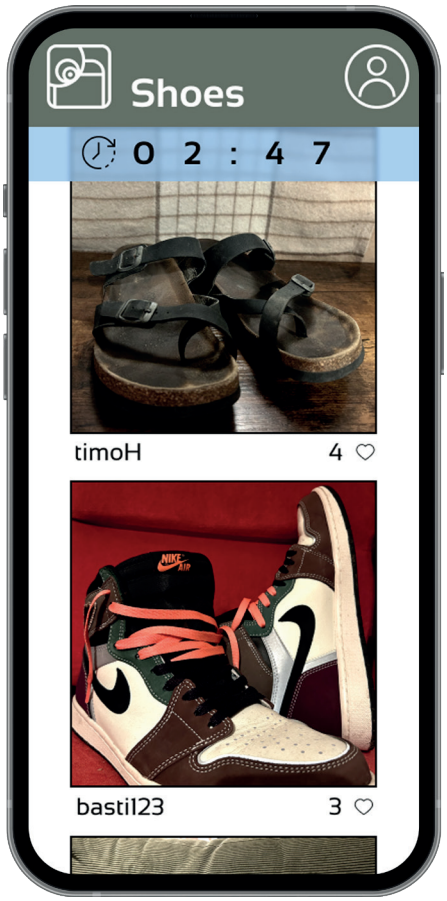

1. Landingpage


2. Registrierungs-formular

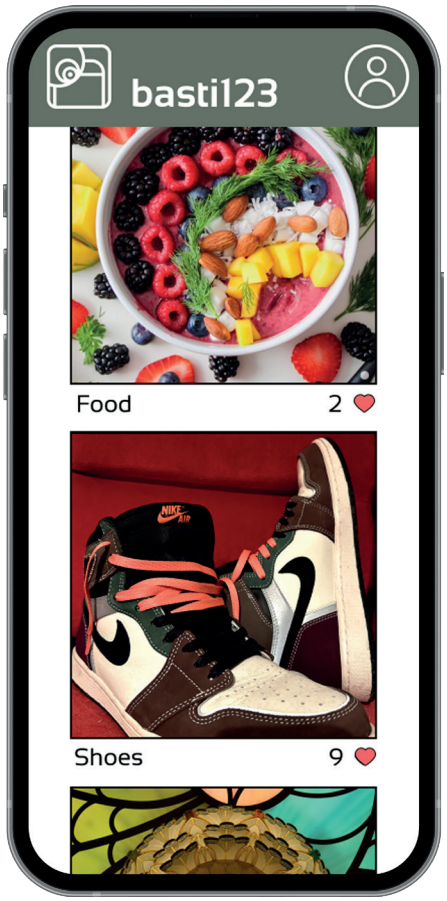

3. Einreichungsphase


4. Kamerazugriff

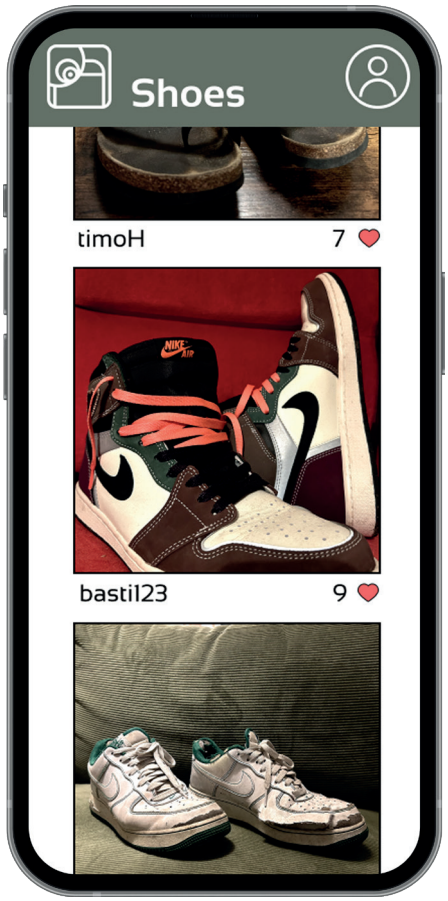

5. Bestätigungsansicht



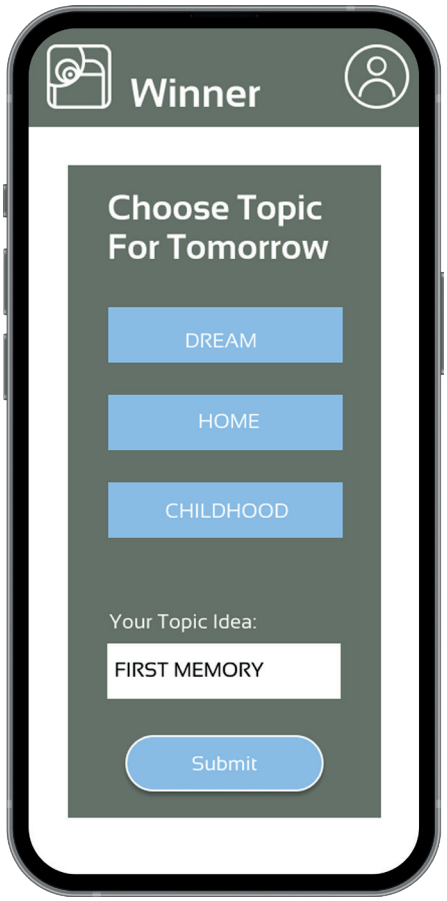

6. Bewertungsphase




7. Nutzerprofil




8. Themengalerie




9. Gewinneransicht

Unser persönlicher Eindruck der Anwendung ist größtenteils positiv. Natürlich gibt es immer Verbesserungsmöglichkeiten und Schönheitsreparaturen, wir sind allerdings sehr erfreut, dass wir uns realistische Ziele setzen und diese auch umsetzen konnten.

Im Laufe der Entwicklung der Anwendung konnten wir eine Menge über die Konzeption und die Abläufe bei der Entwicklung einer Webanwendung lernen. Neben Programmierwissen und dem Erlernen der Programmiersprache Python, konnten wir unser Wissen im Bereich des Prototypings mit Figma erweitern und damit gestalterische, sowie interface- und interaktionsspezifische Kompetenzen erlernen.

## 5.1 Zukunftsausblick und Erweiterungsmöglichkeiten

Neben den bisher implementierten Funktionen haben wir noch weitere Ideen für mögliche Erweiterungen und zusätzliche Funktionen, die in Zukunft umgesetzt werden können. Diese werden im Folgenden erläutert:

- Eine Kommentar-Funktion innerhalb der Anwendung, um neben den Bewertungen eine weitere Möglichkeit für Feedback zu den Bildern zu haben.
- Ein integriertes Freunde-System, um bestimmte Benutzer in der eigenen Freundesliste hinzuzufügen und so einfacher zu deren Profil zu navigieren und über deren eingereichte Bilder Überblick zu halten.
- Eine Suchfunktion für frühere Themen, um einfacher zu bestimmten Galerien zu gelangen.
- Die Möglichkeit, die quadratische Beschneidung des Bildes selbst festzulegen. Aktuell werden die Ränder der längeren Bildseite abgeschnitten, ohne dass der Nutzer darauf direkten Einfluss nehmen kann. In einer Art verschiebbaren Maske könnten die Nutzer so den Beschnitt ihrer Bilder selbst festlegen.
- Neben den täglichen Themenbegriffen auch längerfristige Challenges mit längeren Zeiträumen, zu denen man individuell beitreten kann.
- Ein Bearbeitungstool, um Bildeinstellungen zu individualisieren und die Bildeigenschaften anzupassen. So könnte man den Bildern seinen persönlichen Look mitgeben.
- Aktuell ist die Anwendung wegen der Zeitintervalle für die einzelnen Spielphasen nur für den europäischen Raum ausgelegt. Eine Anpassung des Picture Games für andere Zeitzonen würde Internationalität gewährleisten.
- Langfristig wäre es wahrscheinlich sinnvoll, die Anwendung, anstatt einer Browseranwendung, zu einer im Appstore des Smartphones herunterladbaren Smartphone-App zu machen. So könnte man mit Push-Benachrichtigungen arbeiten, um die Nutzer an die einzelnen Spielphasen zu erinnern. Das Benutzen einer separierten App mit Icon auf dem Startbildschirm des Smartphones ist zudem für die meisten Nutzer intuitiver als jedes Mal die Website über den Browser aufrufen zu müssen.

Diese Erweiterungsmöglichkeiten würden unsere Anwendung noch benutzerfreundlicher machen, wir denken allerdings, dass wir unsere Idee und unser Konzept gut umsetzen konnten und sind im Allgemeinen mit dem Endprodukt sehr zufrieden

# 5. Quellennachweis

Django: https://www.djangoproject.com/

Pillow: https://pillow.readthedocs.io/en/stable/

Figma: https://www.figma.com/files/recent?fuid=1182949022825363310