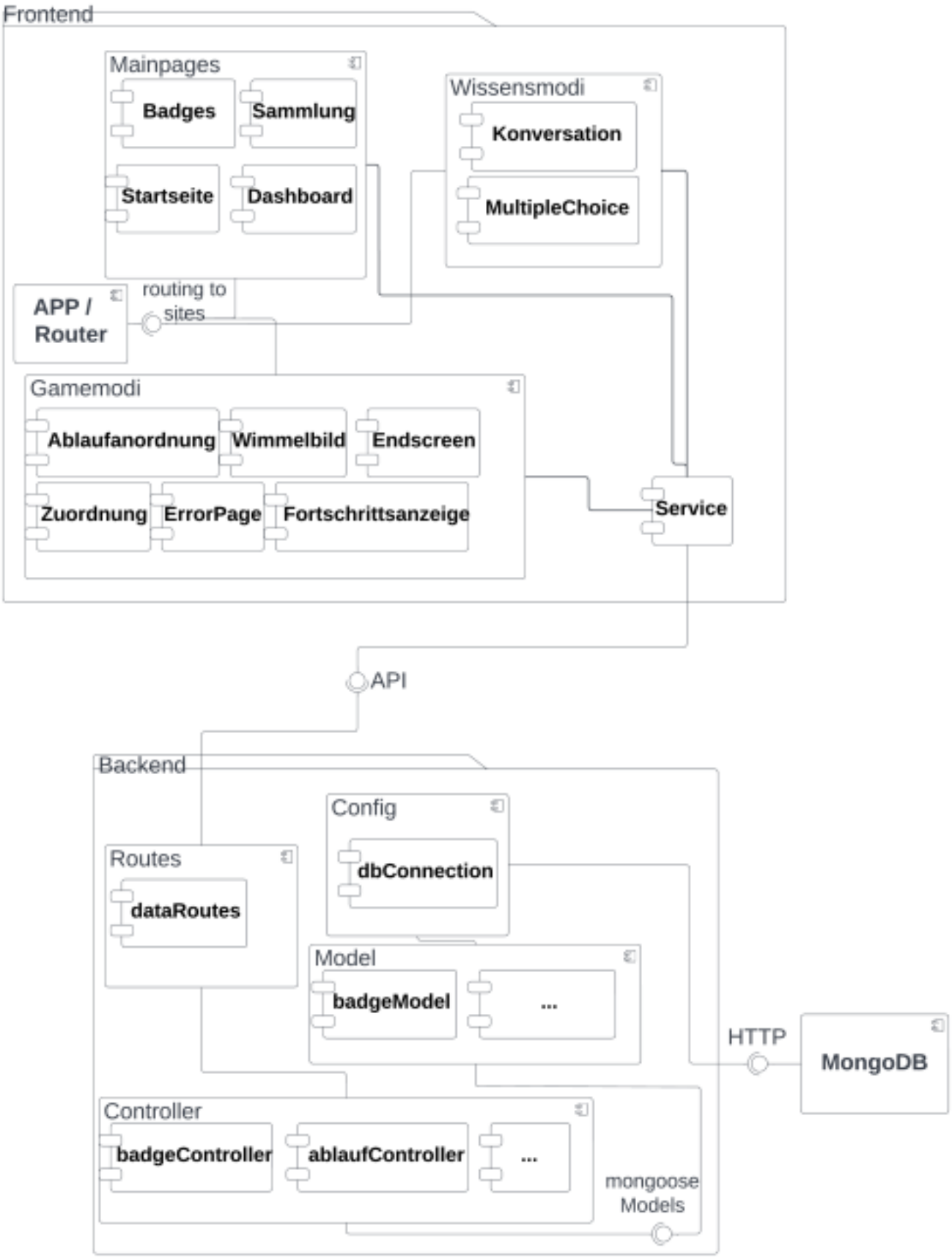
Systemarchitektur

QWERTZ

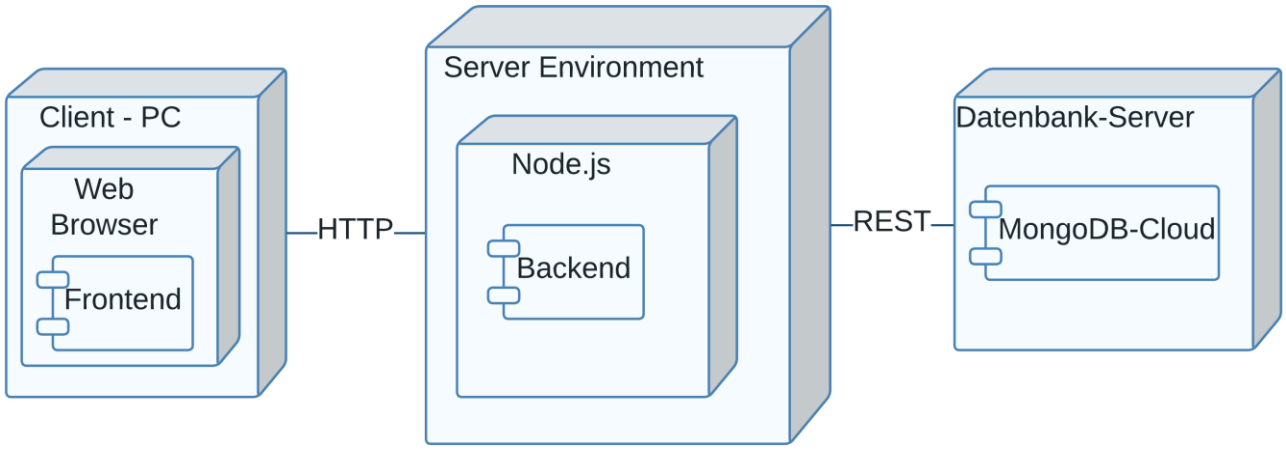
Komponentendiagramm



Die Komponenten sind entsprechend in die drei Schichten Frontend, Backend und Datenbank aufgeteilt. Im Frontend befindet sich die Startmethode APP, welche zudem als Router für die einzelnen Komponenten des Frontend dient. So steuert dieser die Mainpages und die interaktiven Game-/Wissensmodi an. Damit diese Klassen mit dynamischen Daten versorgt werden können, gibt es eine Service-Klasse, welche als Schnittstelle zur API gilt.

Im Backend befindet sich dann eine Routing-Klasse, welche die Anfrage an eine entsprechende Controller-Klasse übergibt. Damit der Controller auf die Daten aus der MongoDB zugreifen kann, werden diese im Code in Form von Mongoose-Models dargestellt. Des Weiteren wird sichergestellt, dass die Datenbankverbindung funktioniert. Zuletzt bietet Mongoose dann entsprechende Schnittstellen, mit der die Daten angefordert werden können und die Daten als JSON zurückgibt.

Verteilungsdiagramm



Das Backend läuft getrennt vom Frontend auf einem eigenen Server. Die Datenhaltung wird durch eine MongoDB realisiert, die durch den MongoDB Cloud-Dienst Atlas bereitgestellt wird.

Schnittstellen

Komponente	Daten	Methoden
APP / Router	Eingang: 2. URL Ausgang: - React-Komponente	- hat keine Schnittstelle nach außen
Badges	Eingang: - JSON mit Badge-Data	- hat keine Schnittstelle nach außen
Startseite	Eingang: - JSON Zitate-Data	- hat keine Schnittstelle nach außen

Dashboard	Eingang: - Alle Modi	- hat keine Schnittstelle nach außen
Sammlung	Eingang: - Bild aus Datenbank	- hat keine Schnittstelle nach außen
Game-/Wissen Modi	Eingang: - JSON Modi-Data	- hat keine Schnittstelle nach außen
Endscreen	Eingang: - Bilder aus der DB	- hat keine Schnittstelle nach außen
ErrorPage	Eingang: - Name eines Modi	- hat keine Schnittstelle nach außen
Service	Ausgang: - JSON-Daten - PDF (als Base64) - Image (als Base64) - Gif (als Base64)	- get{ModiName}(badgeID, modiID) - getPdf(badgeID) - getImage(imageName) - getGif(gifName)
dataRoutes	Eingang: - URL Ausgang: - JSON-Daten - PDF (als Base64) - Image (als Base64) - Gif (als Base64)	- get{ModiName}/badgeID/modiID - getPdf/badgeID - getImage/imageName - getGif/gifName

Beschreibung Datenmodell

Die Daten sind in einer MongoDB gespeichert und liegen daher alle im JSON – Format vor. So wird im folgende ein Beispiel gezeigt bezogen auf den Gamemodi Ablaufanordnung:

_id	6299f6de3e03c315dce5dd75
badgeID	1
modiID	1

data	[{
	"text": "Nicht in Panik geraten, das infizierte Gerät nicht ausschalten oder das Netzteil ziehen, es wird ein laufendes System benötigt"
	},
	. . .
	{
	"text": "Ist das infizierte Endgerät in Quarantäne und das restliche Netzwerk gesichert versuchen sie sämtliche Daten zu sichern auf dem Betroffenen Gerät für eine spätere Analyse"
	}]

So handelt sich hierbei um ein Objekt in der Collection "ablaufs" welche sich demnach dann anhand ihres Contents, also dem "data" Feld, unterscheiden. Anhand der badgeID wird unterschieden zu welchem Badge der Modus gehört und anhand der "modiID" wir unterschieden welche Modi es ist.

Alle weiteren Collections folgen diesem Schema lediglich unterscheiden diese sich anhand ihrer Attribute. Relevant ist dabei aber besonders das jeweils immer die "badgeID" angepasst werden muss für den entsprechenden Modi sowie der Content(data).

Fundamental Modeling Concepts Diagramm