Dokument vertraulich
Darf nicht an dritte weitergegeben werden

iTweet Release2014a
Mobile

Date	Author	Remarks
20140704	uk	initial
20140715	uk	update
20140717	uk	update
20140729	uk	update
20140804	uk	update
20140806	uk	update
20140814	uk	update
20140918	uk	update

W.B.S.S - Web Based Software Solutions - Uraniastrasse 26 - 8001 Zürich - +41 44 431 71 56 - info@wbss.ch

Key issues

Login

- Wie mit Dani besprochen wird nach erfolgreichem Login ein loginToken {token} zurückgeliefert welcher in der App abgelegt wird.
- Dieser loginToken wir bei jedem Request mitgeliefert und validiert
- Der loginToken wir beim setzten des Passworts erstellt resp. beim ändern des Passworts
- Passwörter können nur über das Webinterface geändert werden
- Wann ein loginToken seine Gültigkeit verliert wir auf der Applikations-Seite festgelegt
- Ist der loginToken ungültig so wird der Login-Screen gezeigt
- Der User kann sich abmelden und kommt danach auf die Landing-Page (Google maps) —> **Der loginToken wird auf null gesetzt und “showContext” auf “true”**
- Ist das Login erfolgreich werden die Kategorien zu dem loginToken persistent geladen
- Ist das Login erfolgreich wird der Brand zu dem loginToken persistent geladen
- Ist das Login erfolgreich wird der Kontext zu dem loginToken persistent geladen
- Ist das Login erfolgreich wird erfährt die persistente Variable “showContext” ein update

Nicht gesendete Meldungen managen

- Nicht gesendete Meldungen werden in einer List gezeigt und können “von Hand” verschickt werden
- Man sollte eine Icon positionieren so dass er sieht, dass er nicht gesendete Meldungen hat (ev. mit Anzahl)
- Wenn er eine Meldung nicht schickt und eine neue erstellen will, so wird er gefragt ob er die nicht geschickte Meldungen ablegen will
- User soll nicht gesendete Nachrichten löschen können

Meldungen ansehen

- Hier wird der HTML5 Container aus einem Native-Link heraus erstellt, welcher die Meldungsliste im Mobile-Interface aufruft
- Danach ist der HTML5 Container autonom
- Zurück kommt man mit einer von smooh definierten URL

App Kontext

- 1. Hat die Variable “showContext” den Wert “false” wird der Kategorien-Screen geladen
- 2. Sonst wird die Kontext-List aus dem JSON-File in den Kontext-Screen geladen auch wenn das JSON-File leer ist (Abgelegtes json-File geliefert vom Server)
- 3. Der Request auf den Server wird abgesetzt und der Loader wird angezeigt

App Kontext: appId = "itweet"

- 4a. Kommt eine Kontext-Liste zurück, so wird die auf dem Kontext-Screen angezeigt

App Kontext: appId != "itweet" (Kontext-Screen wird nie angezeigt)

- 4b. Kommt eine Kontext-Liste zurück (es kommt genau ein Kontext zurück), so wird der Kontext nicht angezeigt sondern der nächste Screen
- 5. Kommt keine Verbindung zum Server zustande, so wird die vom File geladene Kontext-List angezeigt
- 6. Ist kein Kontext vorhanden kann keine Meldung erfasst werden

Key issues

Neuer Erfassungs-Prozess 1.1

- **Fall Variable “showContext” ist “true”:**
- 1. Der Kontext wird ausgewählt und damit der Token gesetzt
- 2. Ist das Kategorien-File mit diesem Token vorhanden, so wird die Kategorien-Liste in den Kategorien-Screen geladen
- 3. Ist keine Kategorien-Liste vorhanden so wird eine Leer-Liste im Kategorien-Screen angezeigt
- 4. Der Request auf den Server abgesetzt, der Loader angezeigt und die im Response erhalten Kategorien angezeigt und als JSON-File abgelegt
- 5. Ist keine Verbindung zum Server möglich so wird die Kategorien-Liste vom JSON-File angezeigt

- **Fall Variable “showContext” ist “false”:**
- Es wird kein Kontext-Screen angezeigt sondern der nächste Screen geladen

App branding dynamisch

- 1. Ist das JSON-File mit diesem Token vorhanden, so wird der Brand vom JSON-File geladen
- 4a. Der Request auf den Server abgesetzt, der Loader angezeigt und den im Response erhalten Brand angezeigt und als JSON-File abgelegt
- 4b. Ist das vom Server erhalten File leer so wird es abgelegt und der aktuelle Brand beibehalten
- 5. Ist keine Verbindung zum Server möglich so wird der aktuelle Brand beibehalten

Grundsätzliches

- Alle Fehlermeldungen werden als http-Status mit Status-Message zurückgegeben.
- Smooh liest die Status-Message aus (Ohne den http-Status zu berücksichtigen)

Caching

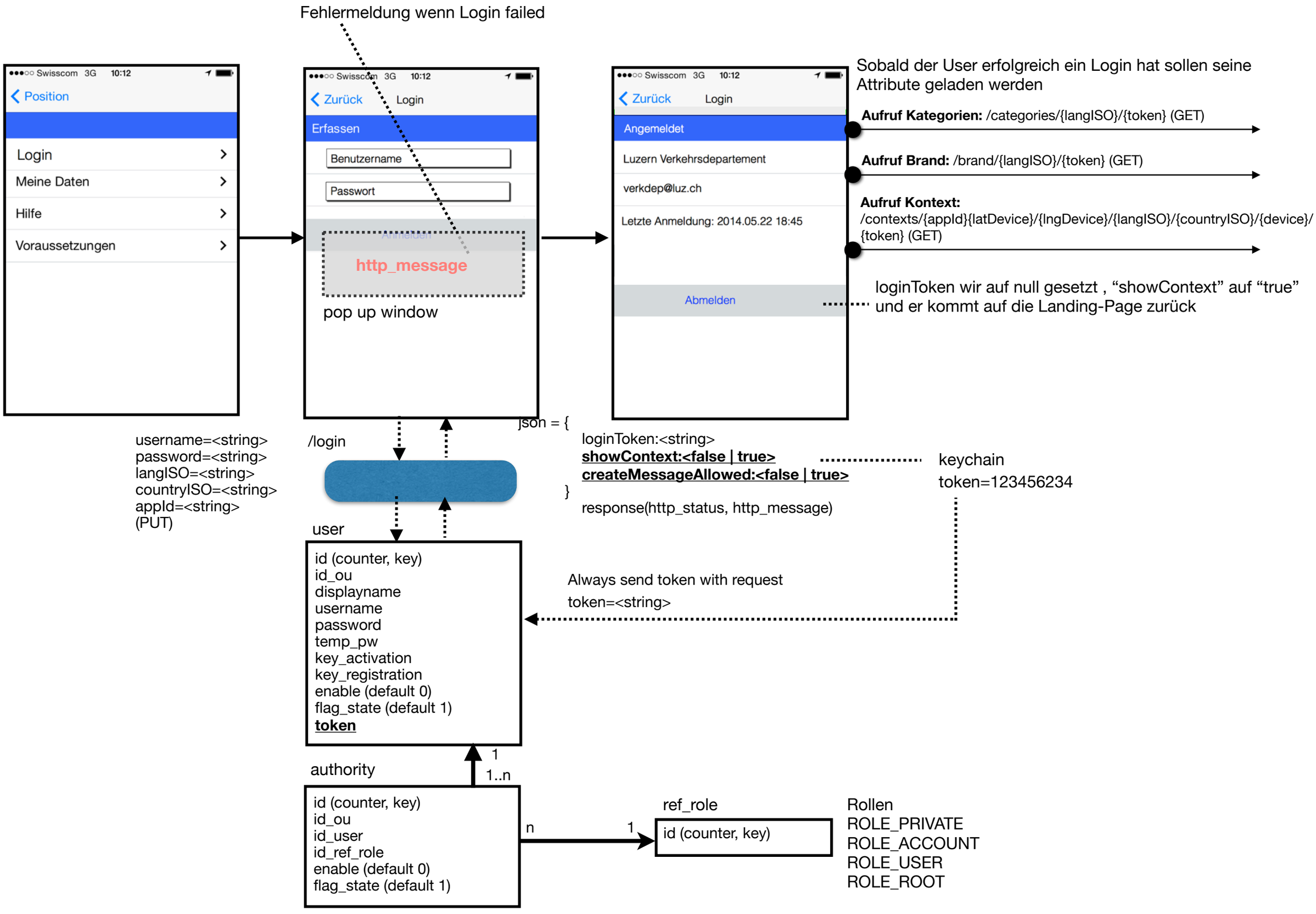
- Vom Konzept her ist es möglich die JSON-Files Kategorien / Brand zu cachen
- Soll man das tun? Wie kann man den Cash wieder löschen?

Darf keine Meldunge erfassen

- Die Variable **createMessageAllowed: false | true** wird beim Login mitgeliefert
- Ist sie **false** darf der User keine Meldungen erstellen

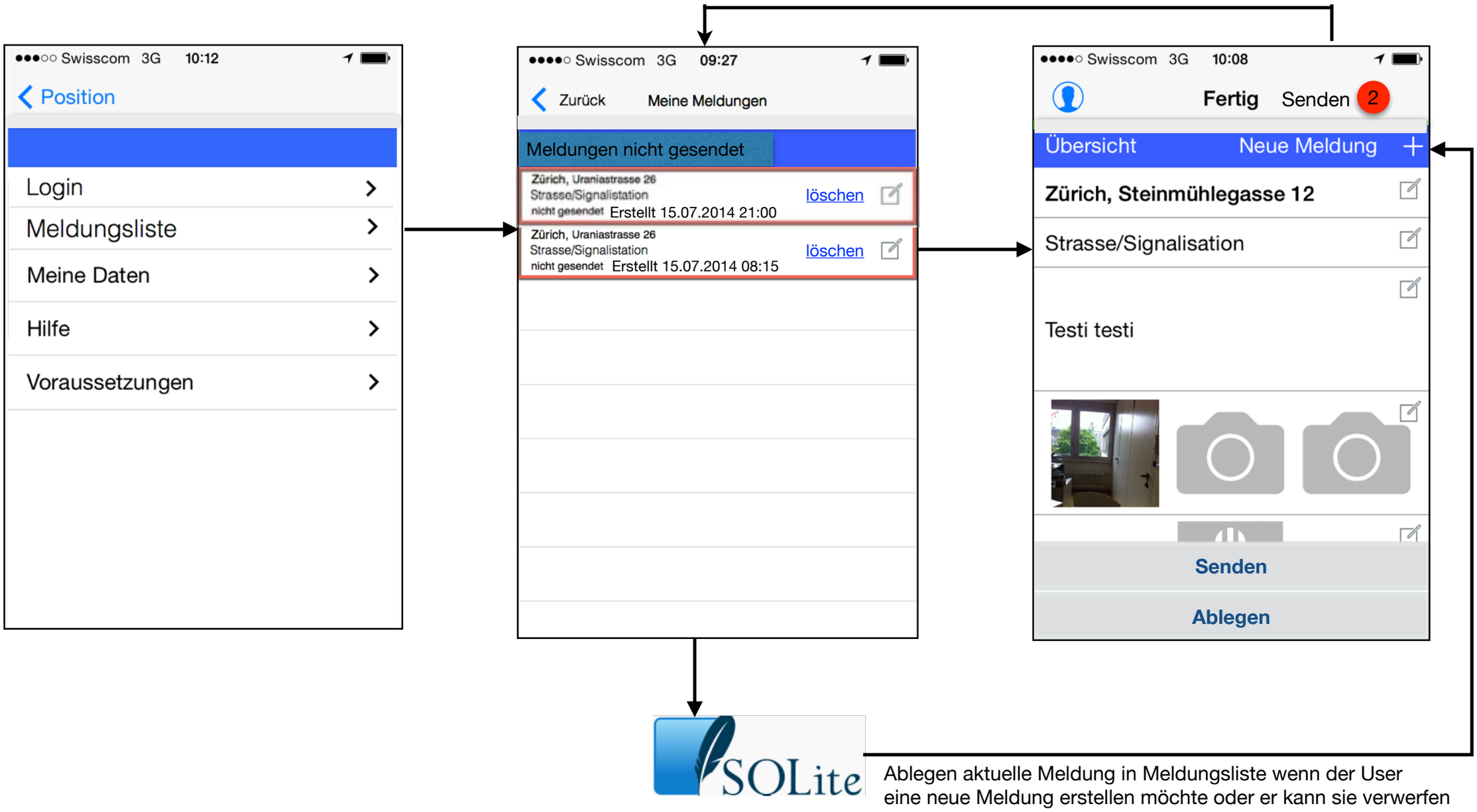
Festgestellte Issues version 1.0

Login


Fehlermeldung wenn Login failed
Swisscom 3G 10:12
Position
Login
Meine Daten
Hilfe
Voraussetzungen
Zurück
Login
Erfassen
Benutzername
Passwort
Anmelden
http_message
pop up window
Angemeldet
Luzern Verkehrsdepartement
verkdep@luz.ch
Letzte Anmeldung: 2014.05.22 18:45
Abmelden
Sobald der User erfolgreich ein Login hat sollen seine Attribute geladen werden
Aufruf Kategorien: /categories/{langISO}/{token} (GET)
Aufruf Brand: /brand/{langISO}/{token} (GET)
Aufruf Kontext:
/contexts/{appId}{latDevice}/{lngDevice}/{langISO}/{countryISO}/{device}/{token} (GET)
loginToken wir auf null gesetzt , “showContext” auf “true” und er kommt auf die Landing-Page zurück
username=<string>
password=<string>
langISO=<string>
countryISO=<string>
appId=<string>
(PUT)
/login
json = {
loginToken:<string>
showContext:<false | true>
createMessageAllowed:<false | true>
}
response(http_status, http_message)
keychain
token=123456234
user
id (counter, key)
id_ou
displayname
username
password
temp_pw
key_activation
key_registration
enable (default 0)
flag_state (default 1)
token
Always send token with request
token=<string>
1
1..n
authority
id (counter, key)
id_ou
id_user
id_ref_role
enable (default 0)
flag_state (default 1)
n
1
ref_role
id (counter, key)
Rollen
ROLE_PRIVATE
ROLE_ACCOUNT
ROLE_USER
ROLE_ROOT

Nicht gesendete Meldungen managen

Diese Module braucht kein Mobile-Interface



Ablegen aktuelle Meldung in Meldungsliste wenn der User eine neue Meldung erstellen möchte oder er kann sie verwerfen

Meldungen ansehen

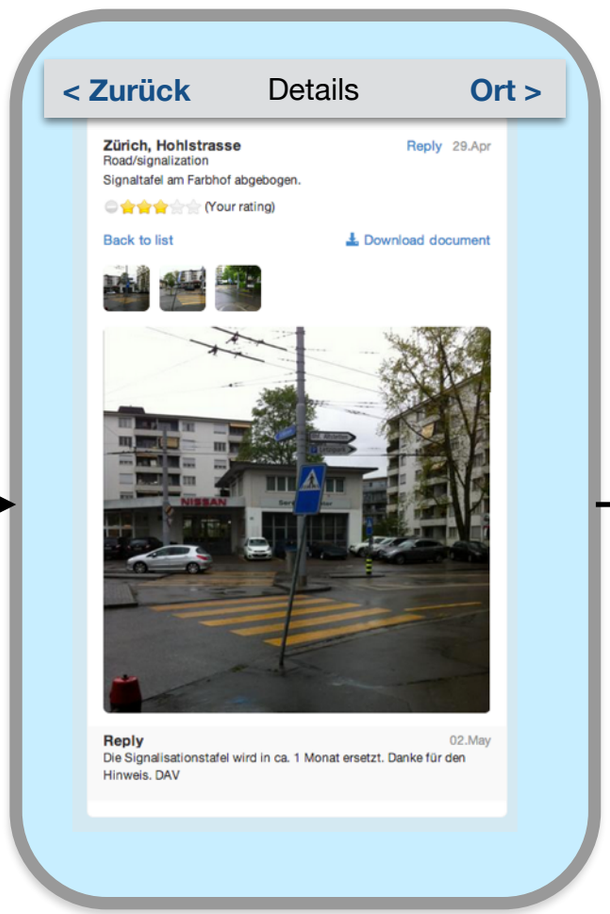
Smooh gibt uns eine URL welche wir eingeben können um die Applikation zurück zu bringen in die iPhone-Native Umgebung.

mvc/mobile/1//items/{appId}{latDevice}/{lngDevice}/{langISO}/{countryISO}/{device}/{token} (GET)

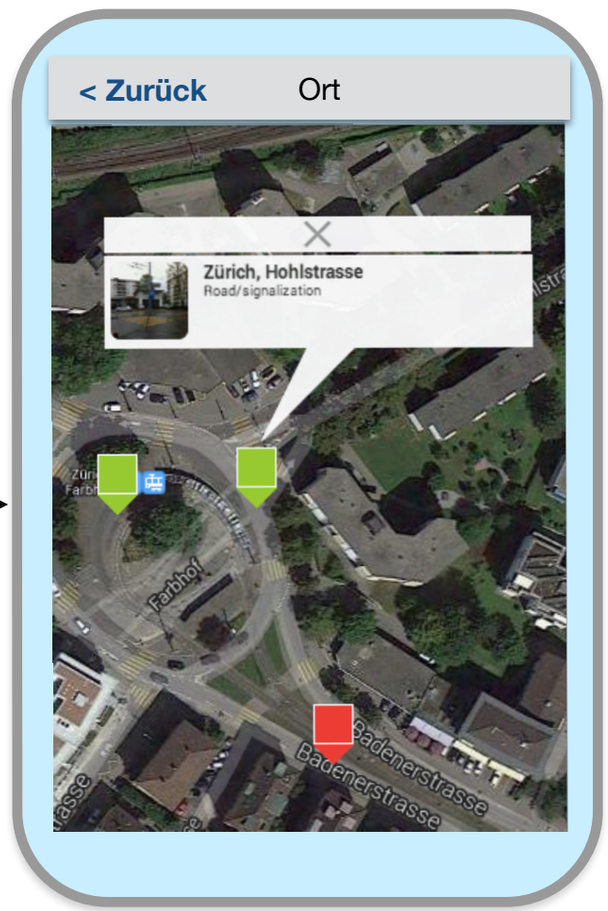
URL

Kontext wählen wenn token null

/item/{id}/{token}
(GET)



/item/map/{id}/{token}
(GET)



HTML5 container

App Kontext

Landing-Page ist immer Google Maps

Aufruf Kategorien: /categories/{langISO}/{token} (GET)

Aufruf Brand: /brand/{langISO}/{token} (GET)

640px - 2xdelta
PNG, deep 24bits

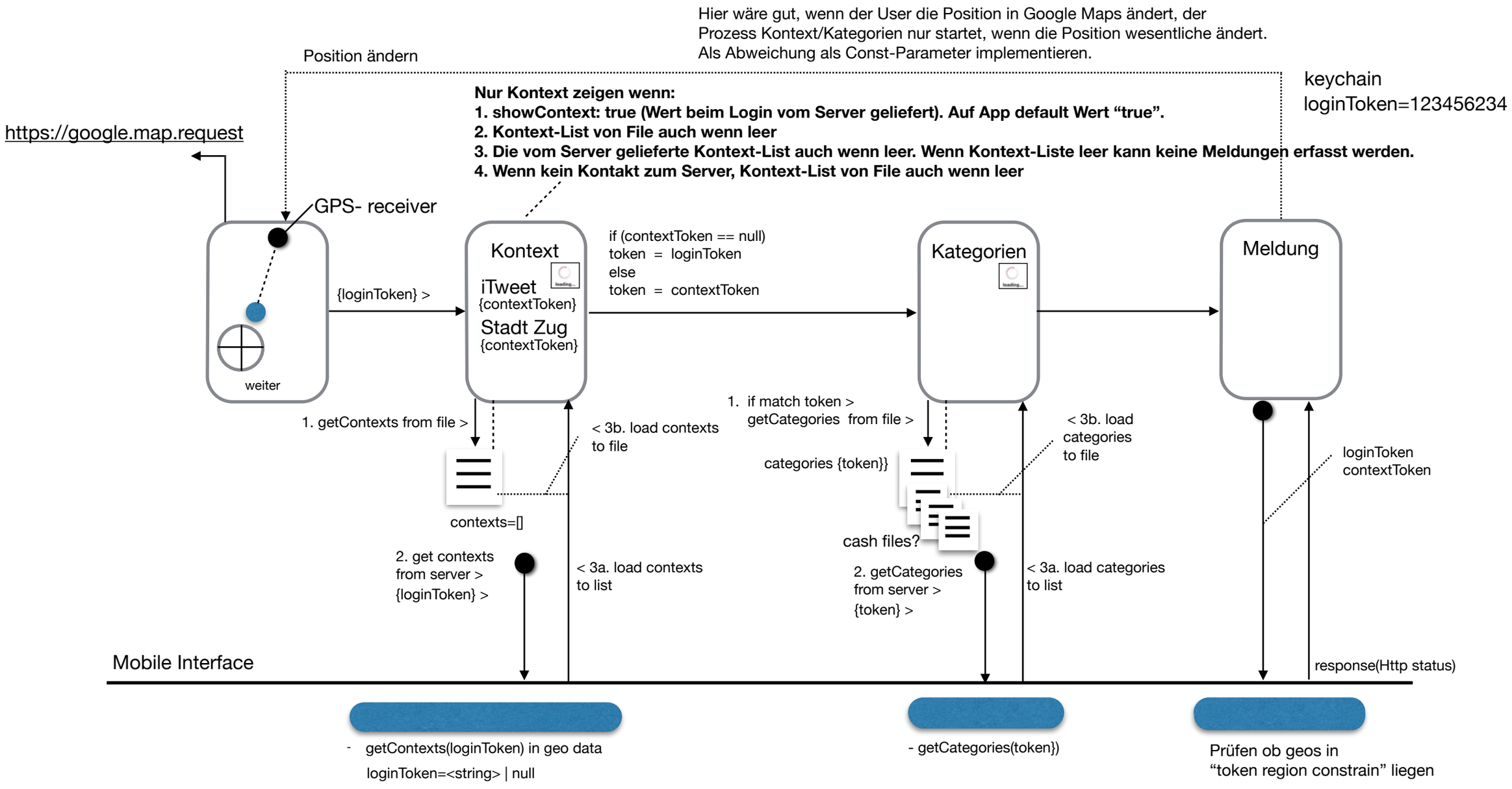
```
json={ contexts= [
        {
                contextToken:<string>,
                urlIconContext: <string>,
                displayName=<string>
        {…,…,…},..]
}
```

mappings_context_user

id_user
context
displayname
txt
id_logo (hasch?)

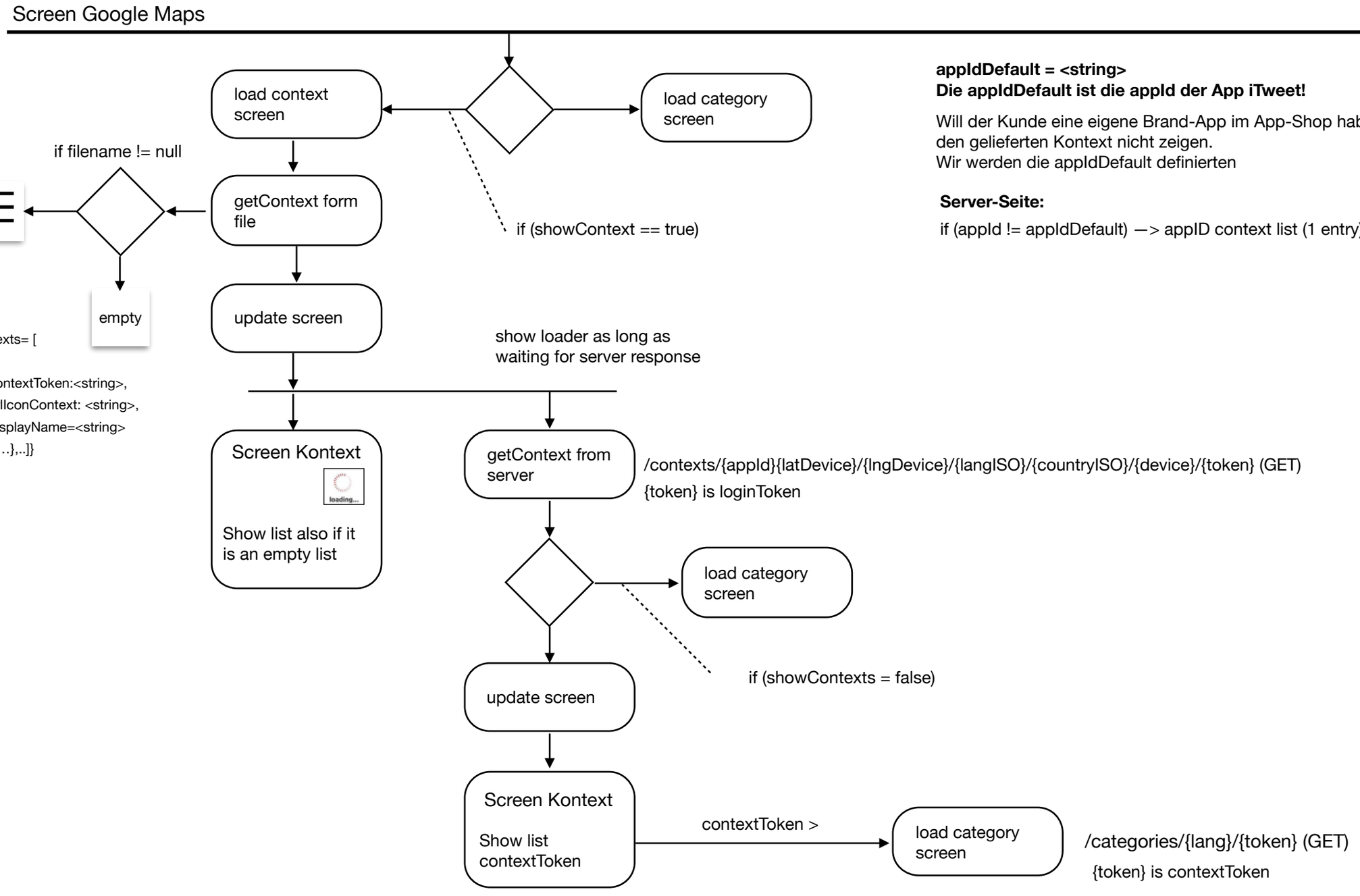
/contexts/{latDevice}/{lngDevice}/{langISO}/{countryISO}/{device}/{token} (GET)

Neuer Erfassungs-Prozess 1.1



Neuer Erfassungs-Prozess 1.1

Aktivitäten-Diagramm Load Context




```
json={ contexts= [
    {
        contextToken:<string>,
        urlIconContext: <string>,
        displayName=<string>
{…,…,…},..]}
```

appIdDefault = <string>
Die appIdDefault ist die appId der App iTweet!

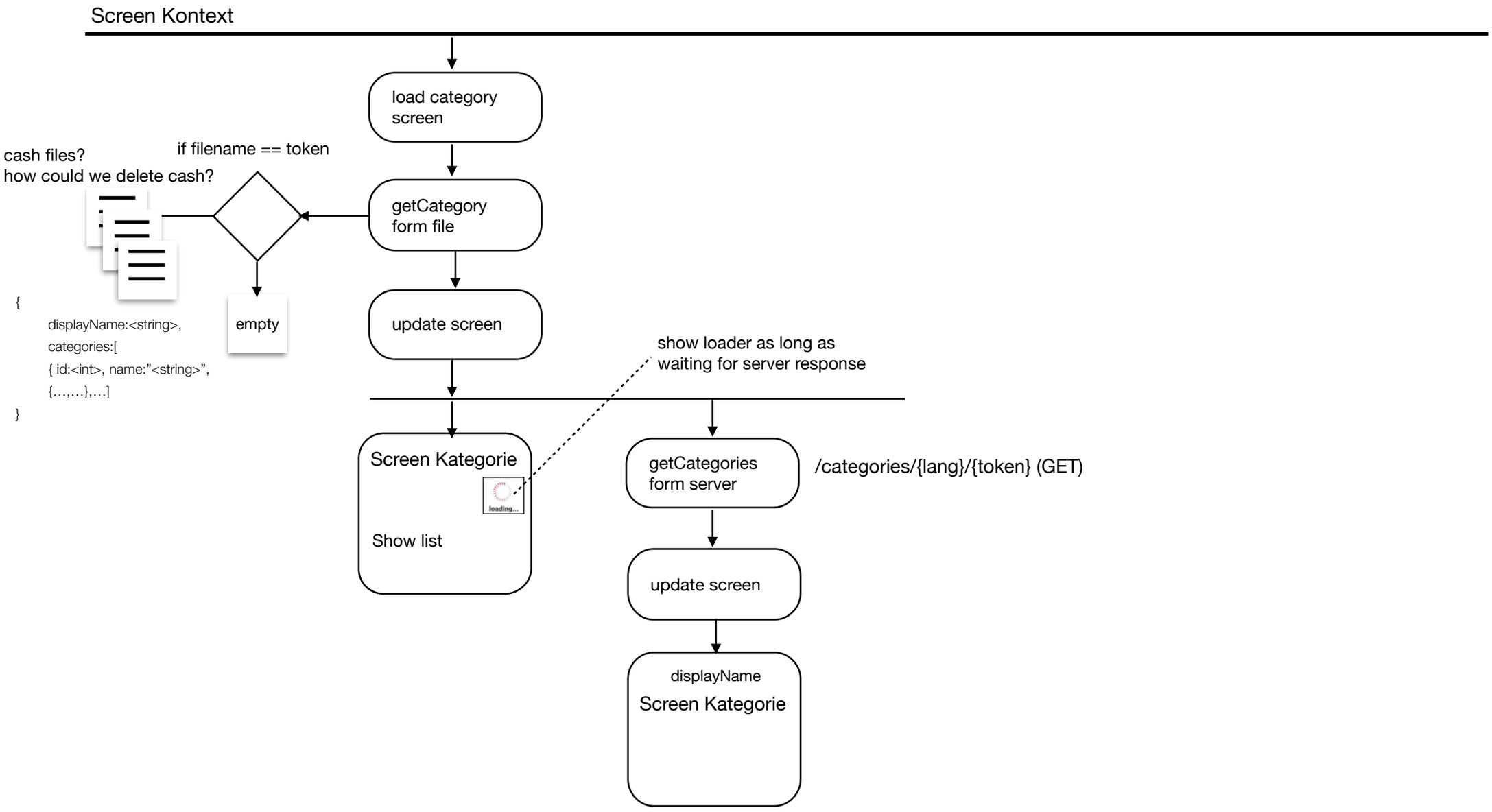
Will der Kunde eine eigene Brand-App im App-Shop haben will so muss man, den gelieferten Kontext nicht zeigen.
Wir werden die appIdDefault definierten

Server-Seite:

if (appId != appIdDefault) —> appID context list (1 entry)

Neuer Erfassungs-Prozess 1.1

Aktivitäten-Diagramm Load Categories



App branding dynamisch

Siehe: http://www.wandern.ch/

categorySelectColor

subheaderColor
subheaderTextColor

Dynamisch token abhängig “displayName” Nutzergruppe.

Logo —> Wie gross soll/kannn das Bild sein?

footerButtonColor
footerButtonTextColor

Aufruf Brand: /brand/{langISO}/{countryISO}/{device}/{token} (GET)

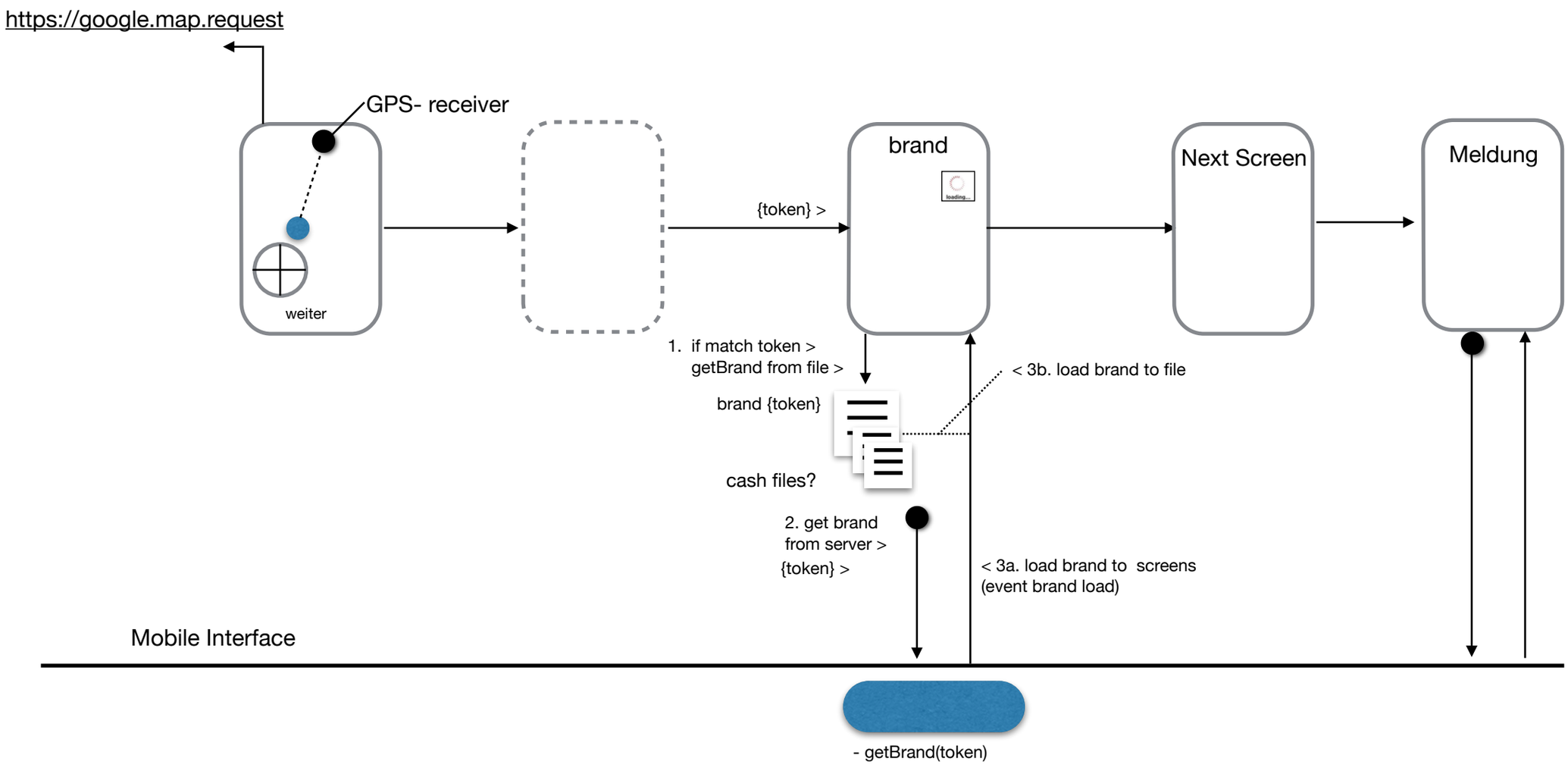
Response Brand

```
{
    categrorySelectorColor: “<string>”,
    subheaderColor: “<string>”,
    subheaderColorText:“<string>”
    footerButtonColor:“<string>”
    footerButtonColorText:“<string>”
    urlIconLogo:”<string>”
}
```

brand

- id
- id_user
- categrory_selector_color
- subheader_color
- subheader_color_Text
- footer_button_color_text
- footer_button_color_text
- url_context_logo
- txt

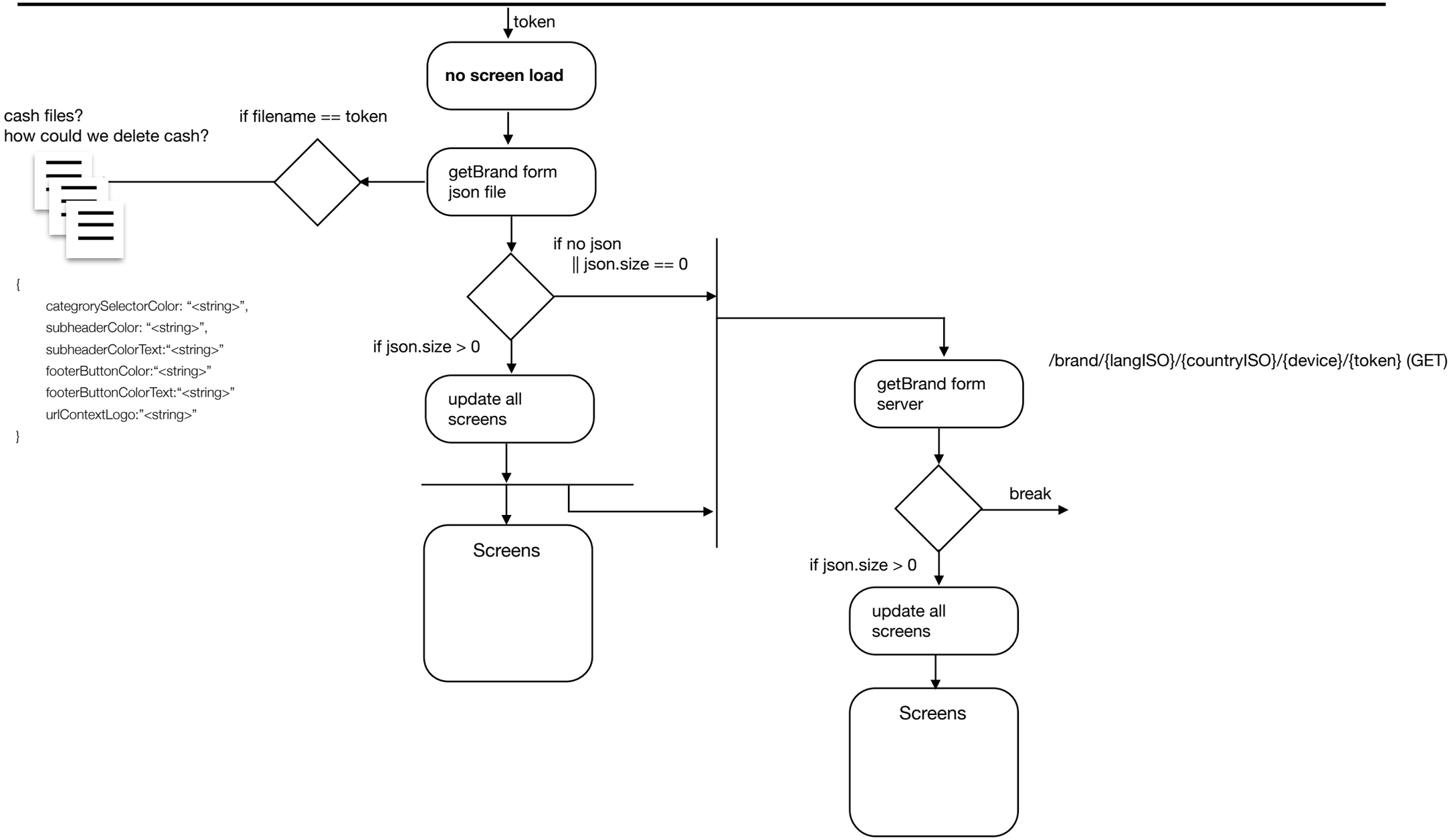
App-Brand dynamisch



App-Brand dynamisch

Aktivitäten-Diagramm Load Brand

Screen Kontext



Darf keine Meldunge erfassen

createMessageAllowed: false

Nach meiner Meinung kann er Meldungen wleche er noch in der Meldungsliste abschicken da dort **loginToken und contextToken** genommen werden mit denen er die Meldungen erfasst hat.
Ist das so?

Festgestellte Issues version 1.0

- App stürzt ab: Wenn man den Ortungsdienst ausgeschaltet hat und auf dem Google maps Screen “Aktuell” drückt stürzt die App ab. Habe aber auch schon gesehen dass ein Popup-Window kommt das sagt, man solle den Ortungsdienst einschalten.

- Button “Meldung erfassen” auf Google Maps Screen blockiert: Habe festgestellte, dass solange Google Maps die Karte nicht geladen hat der Button blockiert bleibt und man nicht weiter gehen kann zu den Kategorien. Wie könnte man so was lösen?

- Scrollen der Kategorien blockiert: Habe festgestellte, dass manchmal wenn man vom Google Maps Screen kommt der Kategorien-Screen wie ein Freeze hat, es wird nicht alles geladen und man kann nicht scrollen. Kann sein dass der am laden ist über https? Wie funktioniert das genau?