Modul 335 – Racr

Aurelio Clavadetscher und Raul Faria

Ein Bild, das Grafiken, Grafikdesign, Schrift, Logo enthält.

Automatisch generierte Beschreibung

Inhalt

[Unsere Idee 3](#_Toc178600328)

[Events Planen: 3](#_Toc178600329)

[Strecken erfassen: 3](#_Toc178600330)

[Setups erfassen: 3](#_Toc178600331)

[MychronSession: 3](#_Toc178600332)

[Mockups 4](#_Toc178600333)

[Architektur 4](#_Toc178600334)

[Datenmodell 5](#_Toc178600335)

[Kommunikation Frontend-Database 5](#_Toc178600336)

[Verwendete Technologien 6](#_Toc178600337)

[Testing 7](#_Toc178600338)

[Testfall 1: Erstellung eines Tracks im UI 7](#_Toc178600339)

[Testfall 1: Durchgeführt durch Aurelio 7](#_Toc178600340)

[Testfall 2: Überprüfung der Datenbank 8](#_Toc178600341)

[Testfall 2: Durchgeführt durch Aurelio 8](#_Toc178600342)

[Testfall 3: Erstellung eines Setups im UI 9](#_Toc178600343)

[Testfall 3: Durchgeführt durch Aurelio 9](#_Toc178600344)

[Testfall 4: Überprüfung der Datenbank 10](#_Toc178600345)

[Testfall 4: Durchgeführt durch Aurelio 10](#_Toc178600346)

[Testfall 5: Erstellung eines Events im UI 11](#_Toc178600347)

[Testfall 5: Durchgeführt durch Aurelio 11](#_Toc178600348)

[Testfall 6: Überprüfung der Datenbank 12](#_Toc178600349)

[Testfall 6: Durchgeführt durch Aurelio 12](#_Toc178600350)

[Testfall 7: Erstellung einer MychronSession im UI 13](#_Toc178600351)

[Testfall 7: Durchgeführt durch Aurelio 13](#_Toc178600352)

[Testfall 8: Überprüfung der Datenbank 14](#_Toc178600353)

[Testfall 8: Durchgeführt durch Aurelio 14](#_Toc178600354)

[Reflexion 15](#_Toc178600355)

[Vorbereitung zur Veröffentlichung 15](#_Toc178600356)

[Installationsanleitung 15](#_Toc178600357)

# Unsere Idee

Wir wollen eine Mobile-App entwickeln von Gokart Fahrer für Gokart Fahrer. Auf der App sollte man Events planen, Strecken erfassen, Rundenzeiten austauschen, Wetterbedingungen für Events sehen, Messdaten vom Bordcomputer teilen und Kommentare schreiben.

## Events Planen:

Es sollte die Möglichkeit geben, um Events zu planen. Events bestehen aus dem Datum, der Strecke, ob es ein Rennen oder Training ist und einer kurzen Beschreibung. Diese Events können dann andere Rennfahrer sehen.  
Fahrer können dann dem Event beitreten. Es ist öffentlich ersichtlich, wer bei einem Event mitmacht.

## Strecken erfassen:

Es sollte die Möglichkeit geben, Strecken zu erfassen, dazu gehört der Standort, der Name, die Beschreibung, die Website, Setups und von wem sie erstellt wurde.

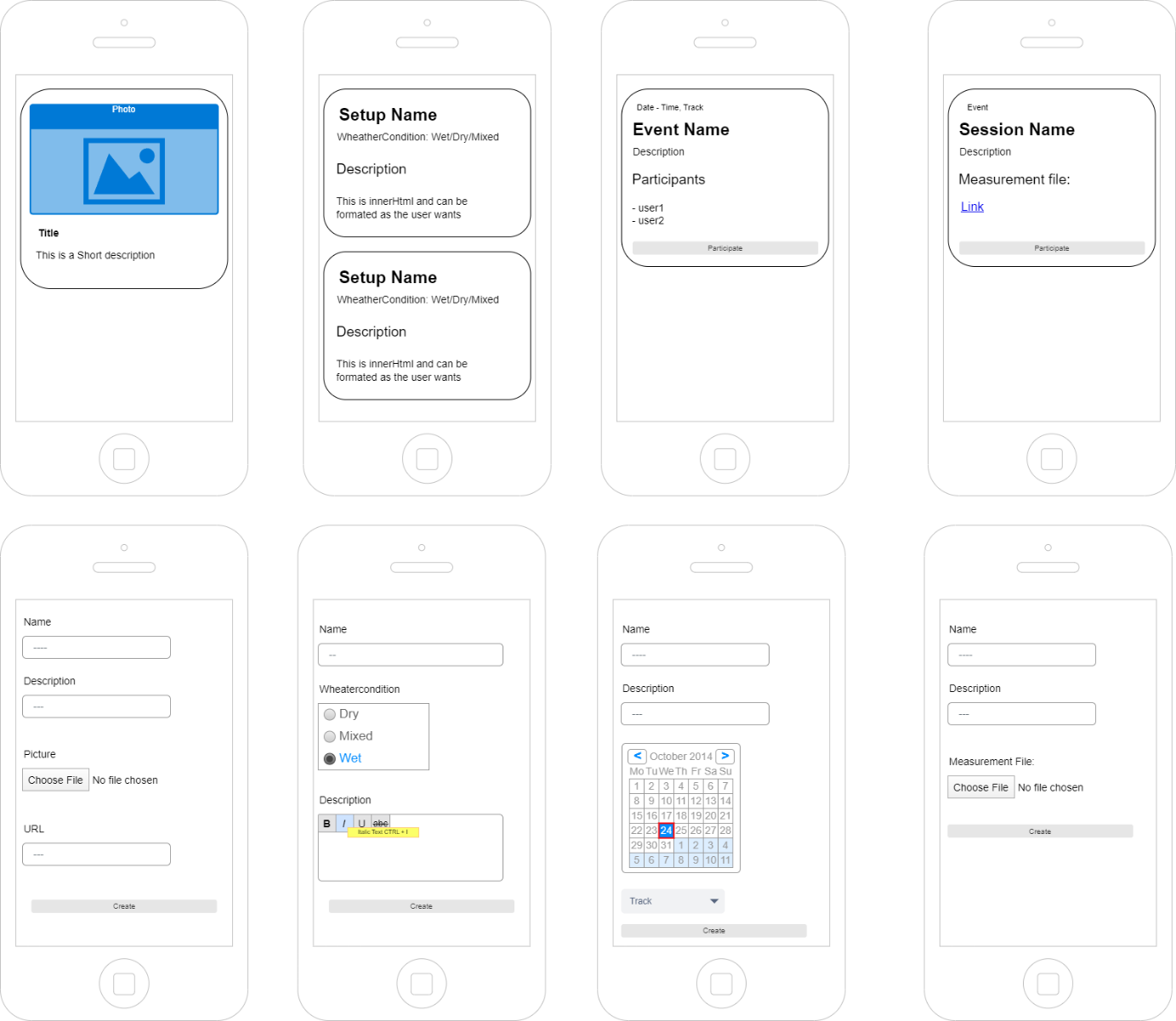
## Setups erfassen:

Es sollte eine Möglichkeit geben, sich Notizen über das Setup des Karts zu machen, z.B Radsturz, Vergaserdüse etc.  
Setups sind von allen einsehbar und gehören zu einer Strecke, können aber optional noch zu einem Event gehören.

## MychronSession:

Mychron sessions sind Daten von den Boardcomputer die jeweils an den Gokarts sind. Eine Mykron session besteht aus der Messdatendatei, der gefahrenen Strecke und dem Setup.

# Mockups



Bessere Ansicht:

**

# Architektur

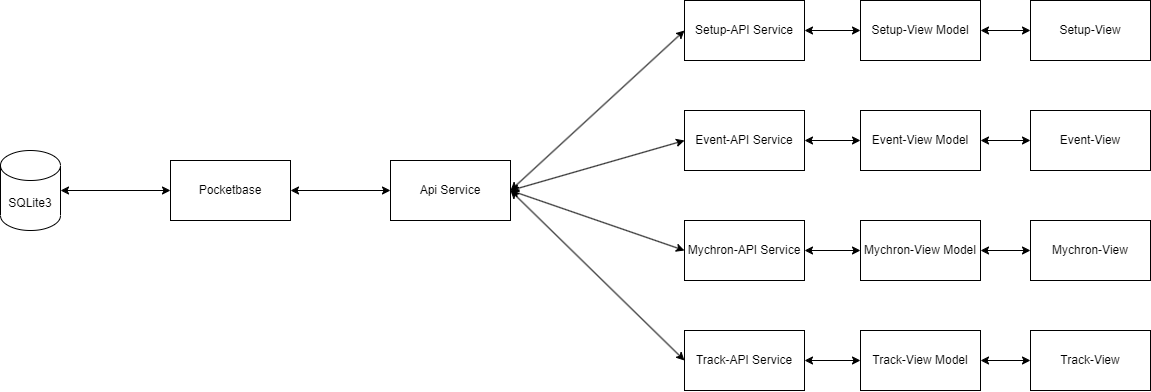
Wir haben eine 3-Tier Architektur verwendet, da wir beide bereits Erfahrung mit dieser Architektur haben. Eine 3-Tier-Architektur teilt ein Softwaresystem in drei Schichten auf: Datenbank, Backend und Frontend. Die **Datenbank** speichert Daten und stellt sie dem Backend zur Verfügung. Das **Backend** (auch als Logikschicht bezeichnet) verarbeitet die Daten, führt Geschäftslogik aus und kommuniziert mit der Datenbank und dem Frontend. Das **Frontend** ist die Benutzerschnittstelle, über die Nutzer mit dem System interagieren. Diese Trennung verbessert Skalierbarkeit, Wartbarkeit und Sicherheit des Systems. Skalierbarkeit und Wartbarkeit ist für dieses kleine Projekt zwar nicht notwendig, jedoch ist es immer gut die Best Practices zu verwenden und zu vertiefen.

# Datenmodell

Ein Bild, das Text, Screenshot, Schrift, Diagramm enthält.

Automatisch generierte Beschreibung

# Kommunikation Frontend-Database



# Verwendete Technologien

In eurer 3-Tier-Architektur habt ihr **Angular** für das Frontend, **Pocketbase** für das Backend und **SQLite3** für die Datenbank verwendet:

1. **Frontend: Angular**
   * **Technologie**: Angular ist ein TypeScript-basiertes Frontend-Framework.
   * **Grund für den Einsatz**: Angular bietet eine strukturierte Art zur Entwicklung von Single-Page-Applications (SPAs) und unterstützt eine klare Trennung von Komponenten und Logik. Ausserdem Arbeiten ich und Raul beide im Betrieb mit Angular und haben somit Bereits Erfarhung
2. **Backend: Pocketbase**
   * **Technologie**: Pocketbase ist ein leichtgewichtiges Backend in Go, das Authentifizierungs- und Dateispeicherfunktionen integriert.
   * **Grund für den Einsatz**: Pocketbase ist einfach zu konfigurieren und ermöglicht die schnelle Entwicklung eines kompletten Backends. Es ist besonders für kleinere Projekte geeignet, da es Datenbank und backend in einer Lösung kombiniert, was Entwicklungszeit und Ressourcen spart.
3. **Datenbank: SQLite3**
   * **Technologie**: SQLite ist eine C-basierte, eingebettete SQL-Datenbank, die in der Anwendung selbst läuft, ohne dass ein separater Server benötigt wird.
   * **Grund für den Einsatz**: SQLite wird häufig eingesetzt, da es sehr ressourcenschonend ist und keine aufwändige Installation oder Verwaltung benötigt. Es speichert die gesamte Datenbank in einer einzigen Datei, was es ideal für lokale Datenspeicherung, mobile Apps und kleine Webanwendungen macht, wo eine vollwertige Datenbanklösung überdimensioniert wäre.

# Testing

## Testfall 1: Erstellung eines Tracks im UI

**Beschreibung: Der Benutzer erstellt einen Track über das UI und füllt alle erforderlichen Felder (Name, Beschreibung, Bild, Website) aus.**

**Schritte:**

1. **Navigiere zur Track-Create page im UI.**
2. **Fülle die Felder aus:**
   1. **Name: "Beispiel Track"**
   2. **Beschreibung: "Dies ist eine Testbeschreibung"**
   3. **Bild: Lade ein gültiges Bild hoch (z.B. JPEG, PNG).**
   4. **Website: Trage eine gültige URL ein (z.B.** [**https://example.com**](https://example.com)**).**
3. **Klicke auf „Create Track“.**

**Erwartetes Ergebnis:**

* **Der Track erscheint in der Track-Liste im UI.**

## Testfall 1: Durchgeführt durch Aurelio

|  |  |
| --- | --- |
| **Durchgeführt** |  |
| **Kriterien erfüllt** |  |

## Testfall 2: Überprüfung der Datenbank

**Beschreibung: Überprüfe, ob der neu erstellte Track in der Datenbank angezeigt wird.**

**Schritte:**

1. **Navigiere zur Track-Create page im UI.**
2. **Fülle die Felder aus:**
   1. **Name: "Beispiel Track"**
   2. **Beschreibung: "Dies ist eine Testbeschreibung"**
   3. **Bild: Lade ein gültiges Bild hoch (z.B. JPEG, PNG).**
   4. **Website: Trage eine gültige URL ein (z.B.** [**https://example.com**](https://example.com)**).**
3. **Klicke auf „Create Track“.**
4. **Gehe auf die Pocketbase Seite**
5. **Gehe zur Track Collection**

**Erwartetes Ergebnis:**

* **Der Track wird korrekt mit allen Details (Name, Beschreibung, Bild, Website) in der Liste angezeigt.**

**3. Testdaten:**

* **Track-Name: "Beispiel Track"**
* **Beschreibung: "Dies ist eine Testbeschreibung"**
* **Bild: Beispielbild (z.B. test\_image.png)**
* **Website: "**[**https://example.com**](https://example.com)**"**

## Testfall 2: Durchgeführt durch Aurelio

|  |  |
| --- | --- |
| **Durchgeführt** |  |
| **Kriterien erfüllt** |  |

## Testfall 3: Erstellung eines Setups im UI

**Beschreibung: Der Benutzer erstellt ein Setup über das UI und füllt alle erforderlichen Felder (Name, Beschreibung, Wheater condition) aus.**

**Schritte:**

1. **Navigiere zur Setup-Create page im UI.**
2. **Fülle die Felder aus:**
   1. **Name: "Beispiel Setup"**
   2. **Beschreibung: "**Dies ist eine Testbeschreibung, **Dieser Text ist fett** und *dieser Kursiv***"**
   3. **Wheater condition: Dry**
3. **Klicke auf „Create Setup“.**

**Erwartetes Ergebnis:**

* **Der Track erscheint in der Setup-Liste im UI.**
* **Die Description wird korrekt dargestellt**

## Testfall 3: Durchgeführt durch Aurelio

|  |  |
| --- | --- |
| **Durchgeführt** |  |
| **Kriterien erfüllt** |  |

## Testfall 4: Überprüfung der Datenbank

**Beschreibung: Überprüfe, ob das neu erstellte Setup in der Datenbank angezeigt wird.**

**Schritte:**

1. **Navigiere zur Setup-Create page im UI.**
2. **Fülle die Felder aus:**
   1. **Name: "Beispiel Setup"**
   2. **Beschreibung: "**Dies ist eine Testbeschreibung, **Dieser Text ist fett** und *dieser Kursiv***"**
   3. **Wheater condition: Dry**
3. **Klicke auf „Create Setup “.**
4. **Gehe auf die Pocketbase Seite**
5. **Gehe zur Setup Collection**

**Erwartetes Ergebnis:**

* **Das Setup wird korrekt mit allen Details (Name, Beschreibung, Wheatercondition) in der Liste angezeigt.**

**3. Testdaten:**

* + **Name: "Beispiel Setup"**
  + **Beschreibung: "**Dies ist eine Testbeschreibung, **Dieser Text ist fett** und *dieser Kursiv***"**
  + **Wheater condition: Dry**

## Testfall 4: Durchgeführt durch Aurelio

|  |  |
| --- | --- |
| **Durchgeführt** |  |
| **Kriterien erfüllt** |  |

## Testfall 5: Erstellung eines Events im UI

**Beschreibung: Der Benutzer erstellt ein Event über das UI und füllt alle erforderlichen Felder (Name, Beschreibung, Datum, Track) aus.**

**Schritte:**

1. **Navigiere zur Track-Create page im UI.**
2. **Fülle die Felder aus:**
   1. **Name: "Beispiel Track"**
   2. **Beschreibung: "Dies ist eine Testbeschreibung"**
   3. **Bild: Lade ein gültiges Bild hoch (z.B. JPEG, PNG).**
   4. **Website: Trage eine gültige URL ein (z.B.** [**https://example.com**](https://example.com)**).**
3. **Navigiere zur Event-Create page im UI.**
4. **Fülle die Felder aus:**
   1. **Name: "Beispiel Event"**
   2. **Beschreibung: "**Beispielevent**"**
   3. **Datum: 01.01.2000**
   4. **Track: Beispiel Track**
5. **Klicke auf „Create Event“.**

**Erwartetes Ergebnis:**

* **Der Track erscheint in der Event-Liste im UI.**
* **Man kann am Event Teilnehmen**

## Testfall 5: Durchgeführt durch Aurelio

|  |  |
| --- | --- |
| **Durchgeführt** |  |
| **Kriterien erfüllt** |  |

## Testfall 6: Überprüfung der Datenbank

**Beschreibung: Überprüfe, ob das neu erstellte Event in der Datenbank angezeigt wird.**

**Schritte:**

1. **Navigiere zur Track-Create page im UI.**
2. **Fülle die Felder aus:**
   1. **Name: "Beispiel Track"**
   2. **Beschreibung: "Dies ist eine Testbeschreibung"**
   3. **Bild: Lade ein gültiges Bild hoch (z.B. JPEG, PNG).**
   4. **Website: Trage eine gültige URL ein (z.B.** [**https://example.com**](https://example.com)**).**
3. **Navigiere zur Event-Create page im UI.**
4. **Fülle die Felder aus:**
   1. **Name: "Beispiel Event"**
   2. **Beschreibung: "**Beispielevent**"**
   3. **Datum: 01.01.2000**
   4. **Track: Beispiel Track**
5. **Klicke auf „Create Event“.**
6. **Gehe auf die Pocketbase Seite und dann zu Collection Events**

**Erwartetes Ergebnis:**

* **Das Event wird korrekt mit allen Details (Name, Beschreibung, Datum und Track) in der Liste angezeigt.**

**Testdaten:**

* + **Name: "Beispiel Event"**
  + **Beschreibung: "**Beispielevent**"**
  + **Datum: 01.01.2000**

## Testfall 6: Durchgeführt durch Aurelio

|  |  |
| --- | --- |
| **Durchgeführt** |  |
| **Kriterien erfüllt** |  |

## Testfall 7: Erstellung einer MychronSession im UI

**Beschreibung: Der Benutzer erstellt eine MychronSession über das UI und füllt alle erforderlichen Felder (Name, Beschreibung, Mychron-File, Track) aus.**

**Schritte:**

1. **Navigiere zur Mychron-Create page im UI.**
2. **Fülle die Felder aus:**
   1. **Name: "Beispiel Track"**
   2. **Beschreibung: "Dies ist eine Testbeschreibung"**
   3. **Bild: Lade ein gültiges Bild hoch (z.B. JPEG, PNG).**
   4. **Website: Trage eine gültige URL ein (z.B.** [**https://example.com**](https://example.com)**).**
3. **Navigiere zur Mychron-Create page im UI.**
4. **Fülle die Felder aus:**
   1. **Name: "Beispiel Mychron"**
   2. **Beschreibung: "**Beispiel Mychron**"**
   3. **Datum: Lade ein .Txt hoch.**
   4. **Track: Beispiel Track**
5. **Klicke auf „Create MychronSession“.**

**Erwartetes Ergebnis:**

* **Der Track erscheint in der Mychron-Liste im UI.**

## Testfall 7: Durchgeführt durch Aurelio

|  |  |
| --- | --- |
| **Durchgeführt** |  |
| **Kriterien erfüllt** |  |

## Testfall 8: Überprüfung der Datenbank

**Beschreibung: Überprüfe, ob die neu erstellte MychronSession in der Datenbank angezeigt wird.**

**Schritte:**

1. **Navigiere zur Mychron-Create page im UI.**
2. **Fülle die Felder aus:**
   1. **Name: "Beispiel Track"**
   2. **Beschreibung: "Dies ist eine Testbeschreibung"**
   3. **Bild: Lade ein gültiges Bild hoch (z.B. JPEG, PNG).**
   4. **Website: Trage eine gültige URL ein (z.B.** [**https://example.com**](https://example.com)**).**
3. **Navigiere zur Mychron-Create page im UI.**
4. **Fülle die Felder aus:**
   1. **Name: "Beispiel Mychron"**
   2. **Beschreibung: "**Beispiel Mychron**"**
   3. **Datum: Lade ein .Txt hoch.**
   4. **Track: Beispiel Track**
5. **Klicke auf „Create Mychron “.**
6. **Gehe auf die Pocketbase Seite und dann zu Collection Mychron**

**Erwartetes Ergebnis:**

* **Das Mychron wird korrekt mit allen Details (Name, Beschreibung, File und Track) in der Liste angezeigt.**

**Testdaten:**

* + **Name: "Beispiel Mychron "**
  + **Beschreibung: "** Beispiel Mychron

## Testfall 8: Durchgeführt durch Aurelio

|  |  |
| --- | --- |
| **Durchgeführt** |  |
| **Kriterien erfüllt** |  |

# Reflexion

Wir konnten den Soll-Zustand sehr gut erfüllen, weil wir diese Technologien schon öfters benutzt haben und wir schon viele Teamarbeiten zusammen bewältigt haben und konnten somit gut im Team arbeiten und planen. Die Dokumentation haben wir ein bisschen unterschätzt, da sie Trotzdem nicht so schnell erledigt ist wie wir gedacht haben. Ausserdem ist bei der Dokumentation auch die Motivation ein grosser Faktor, denn wir beide schreiben nicht gerne Dokumentationen also haben wir die bisschen vernachlässigt aber gegen Endo konnten wir unseren inneren Schweinehund überwinden und konnten sie ohne Probleme Fertigstellen.

# Vorbereitung zur Veröffentlichung

Um unser App zu veröffentlichen, müssten wir noch Eine ABG, ein Impressum und eine Datenschutzrichtlinie haben. Ausserdem müsste die Speicherung der Daten DSGVO entsprechen, denn wenn dies nicht gegeben ist kann es zu rechtlichen Problemen kommen. Jenachdem wo man die App veröffentlichen will müssen davor noch andere Sachen gemacht werden. Im Appstore muss man dem Apple Developer Programm beitreten, dies kostet 100Chf im Jahr. Dann wird die App reviewed und falls alles klappt, muss man einen Appstore Eintrag erstellen und kann dann die App hochladen. Beim Play-Store ist dies ein bisschen einfacher und ohne Kosten verbunden. Die Richtlinien und die Kontrollen sind nicht so strickt.

# Installationsanleitung

* 1. Code auf lokale Maschine kopieren: [Rfaria06/racr](https://github.com/Rfaria06/racr)
  2. Dependencies installieren (npm install)
  3. Dev- web server starten mit ‘npx ionic serve’. Wenn eine meldung kommt, enter drücken:  
     Need to install the following packages:  
     [ionic@5.4.16](mailto:ionic@5.4.16)  
     Ok to proceed? (y)
  4. Es sollte sich automatisch ein Browserfenster öffnen

Um die App auf iOS zu öffnen sind mehrere Schritte benötigt. Dafür muss man einen Mac mit XCode installiert haben, und entweder ein iPhone das auf Developer mode gestellt ist, oder den iOS Simulator auf MacOS.

1. Code auf lokale Maschine kopieren: [Rfaria06/racr](https://github.com/Rfaria06/racr)
2. Dependencies installieren (npm install)
3. Das Projekt kompilieren ‘npx ng build’
4. Kompilierte assets in iOS- Projekt kopieren ‘npx cap copy’
5. Native- plugins synchronisieren ‘npx cap sync --inline’
6. App öffnen ‘npx cap open ios’. Es wird eine Liste mit verfügbaren geräten angezeigt.
7. Das gewünschte Gerät auswählen, und die App sollte automatisch installiert und geöffnet werden.