carchain

Share and rent cars trustably using the Blockchain!

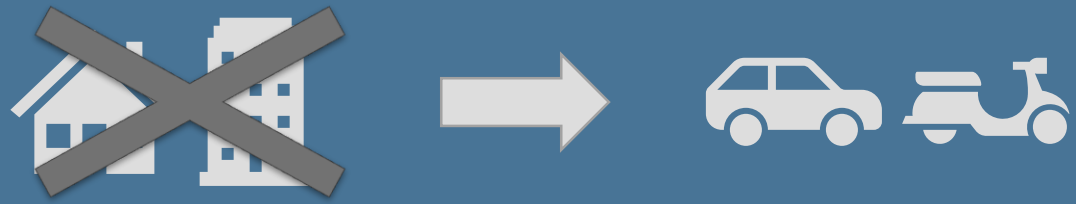
Gliederung

1. Einleitung / Einführung (Simon)
    a. Aufgabe
    b. Idee
    c. Umsetzung & Vision
2. Architektur
    a. Einleitung/Überblick
    b. Server (Bilder) (Faiß)
    c. App (Mieten) (Simon)
    d. Blockchain (Basti)
    e. SmartCar (RaspberryPi) (Nils)
3. Ergebnis: Carchain
    a. Fakten / Stand (Simon)
    b. Demo (Alle?)
4. Fazit (Simon?)

carchain

Einleitung: Aufgabenstellung

„Mietvorgänge mit Blockchain verwalten!“

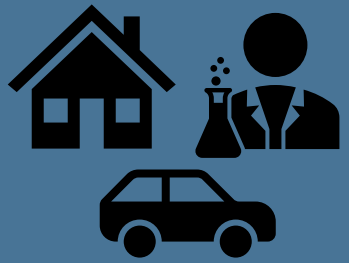


→ Carsharing via Blockchain

CAR 2GO → ethereum

carchain

Die Idee...

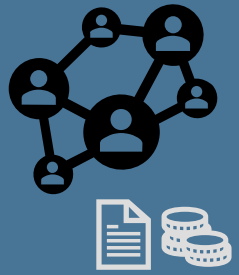


carchain

Die Idee...

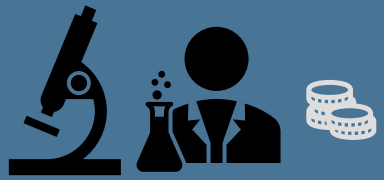
carchain

Die Idee...



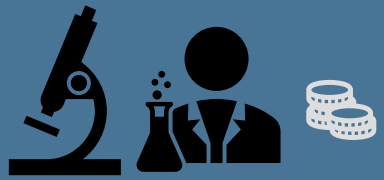
carchain

Die Idee...



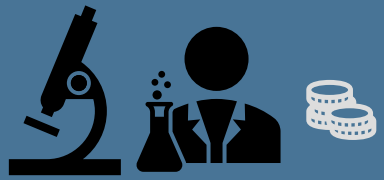
carchain

Die Idee...



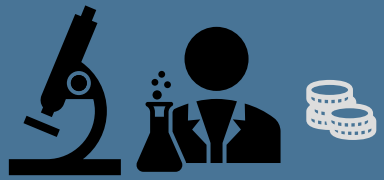
carchain

Die Idee...



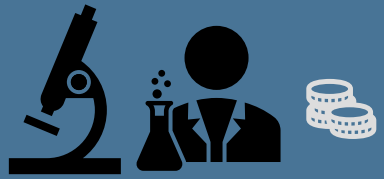
carchain

Die Idee...



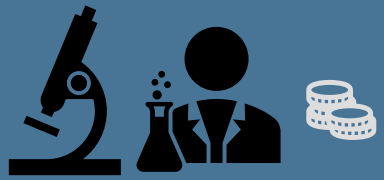
carchain

Die Idee...



carchain

Die Idee...



carchain

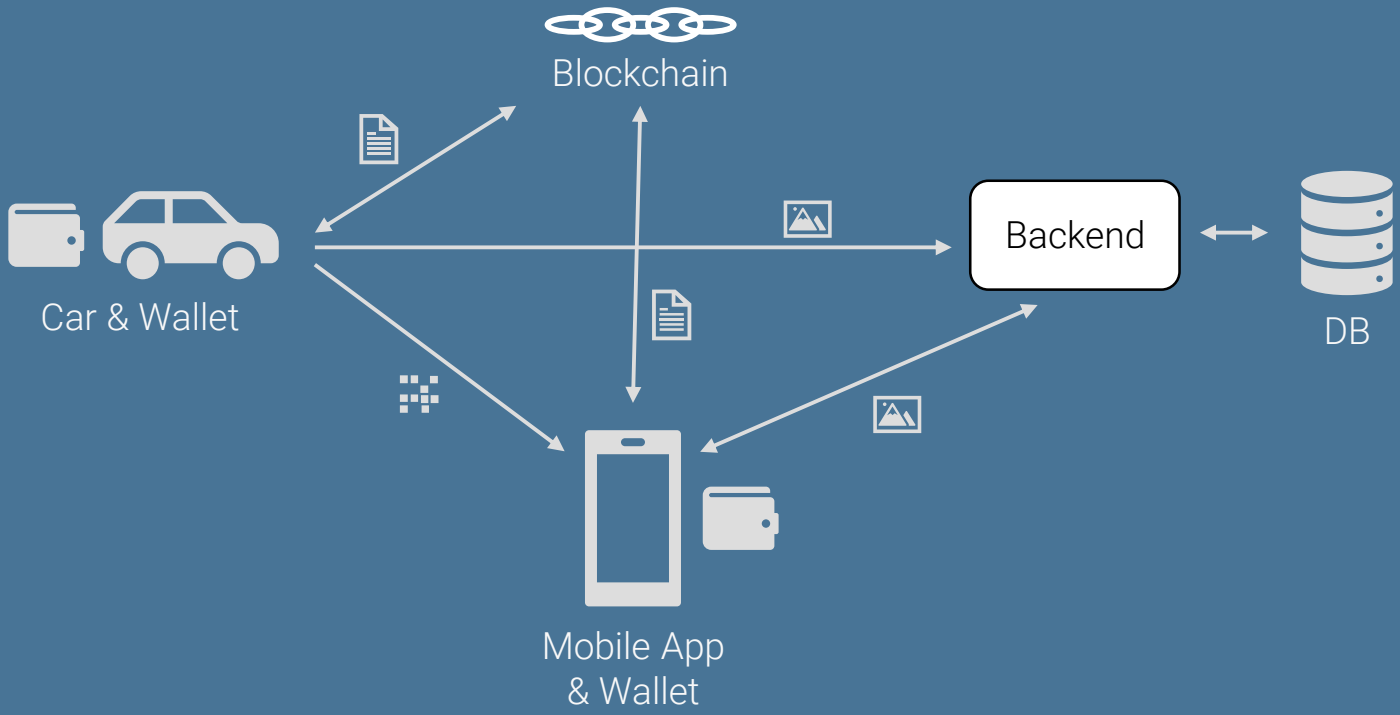
Umsetzung & Vision

Benötigte Komponenten:

- Auto-Adapter (Raspberry)
- mobile App (Android),
- Blockchain & SmartContracts,
- Image-Server

carchain

Carchain: Architekturüberblick v3



carchain

Carchain: Image-Server

- Problem: „Speicher“ auf Blockchain ist sehr teuer!
- Darum: Speichern der Bilder der Autos auf separatem Server

- → Carchain Image-Server

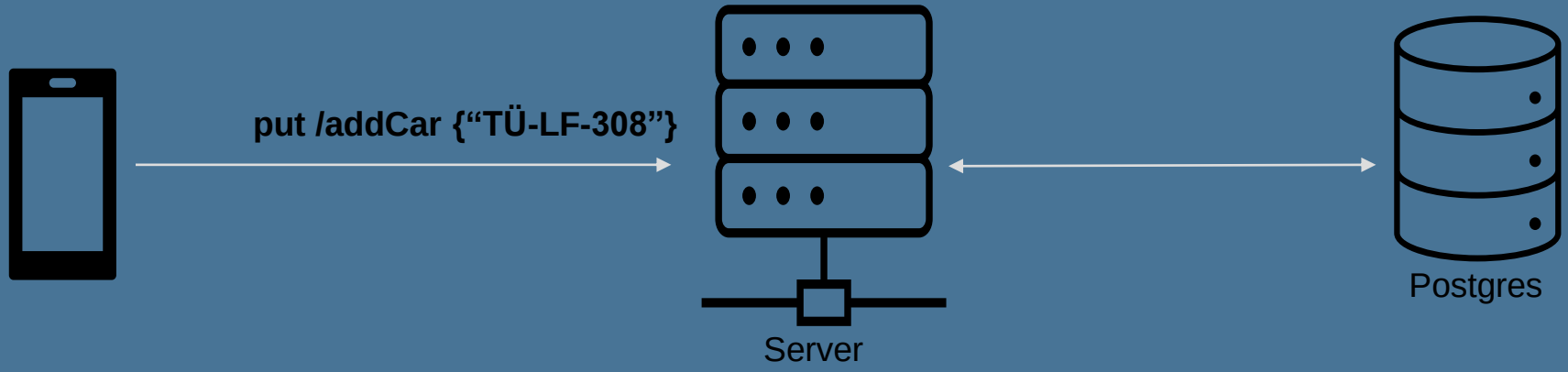
Carchain: Image-Server

PUT	193.196.54.51:3000/register
PUT	193.196.54.51:3000/addImage
GET	193.196.54.51:3000/getImage/1
PUT	193.196.54.51:3000/addCar
GET	193.196.54.51:3000/getImages/TÜ-LF-308
DEL	193.196.54.51:3000/deleteCar/213

- Methoden auf Server:
  - Nutzer registrieren
  - Auto anlegen
  - Auto löschen
  - Bild für Auto hinzufügen
  - Alle Bilder eines Autos abrufen
  - Konkretes Bild abrufen
- Technologien:
  - Node.js & PostgreSQL

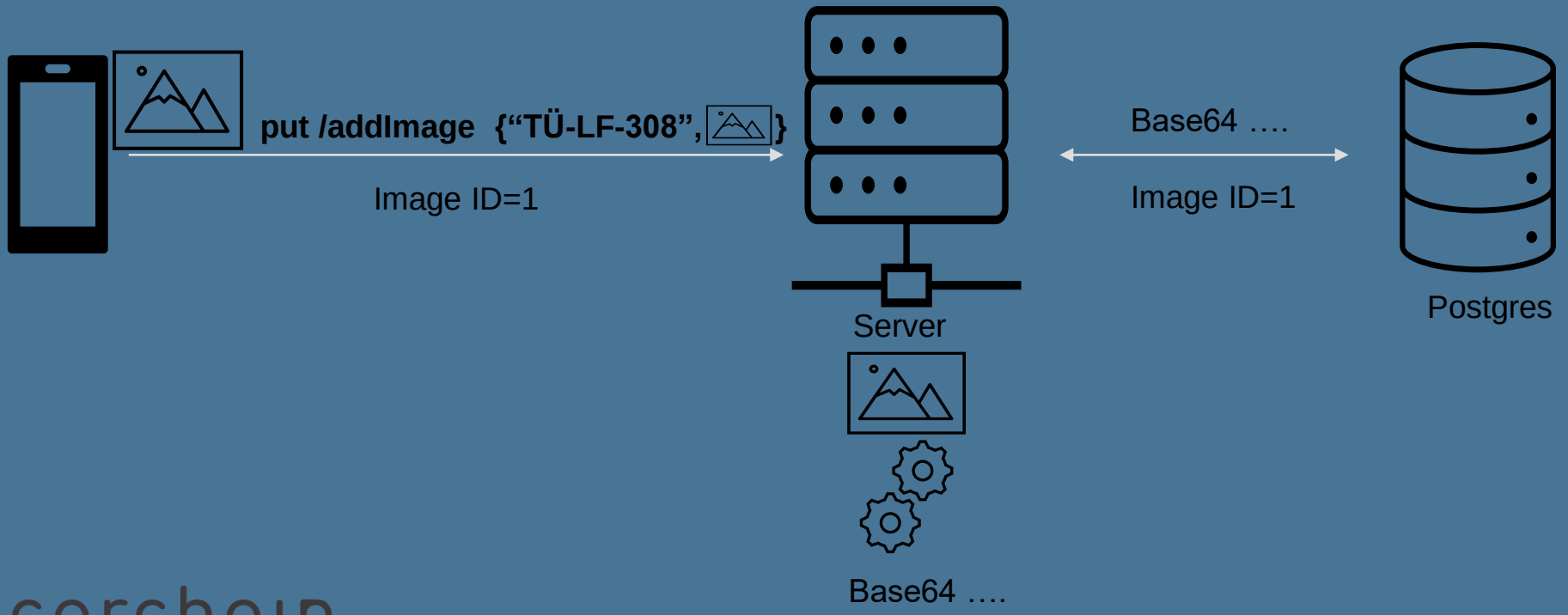
carchain

Carchain: Image-Server-Funktionen



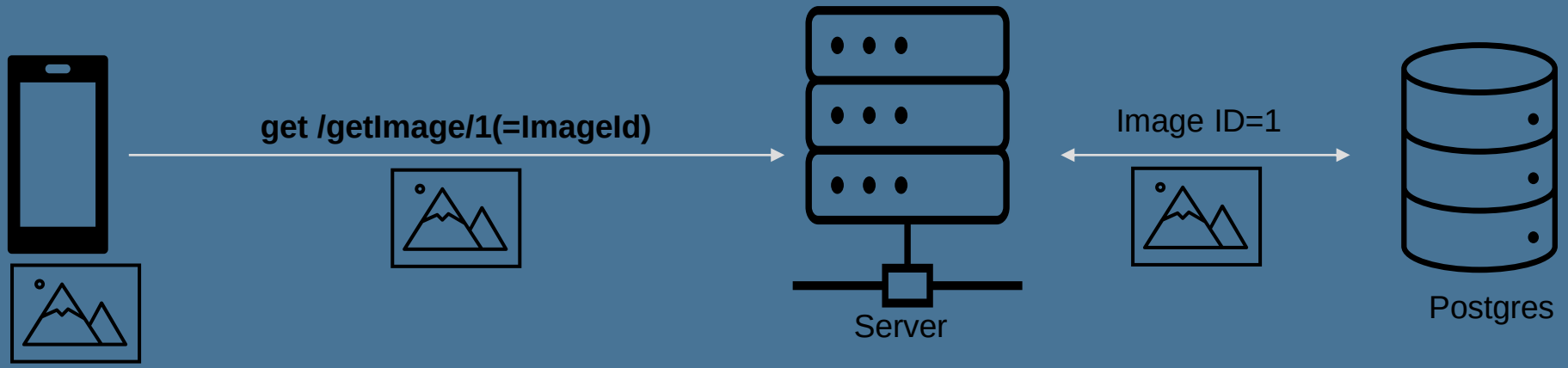
carchain

Carchain: Image-Server-Funktionen



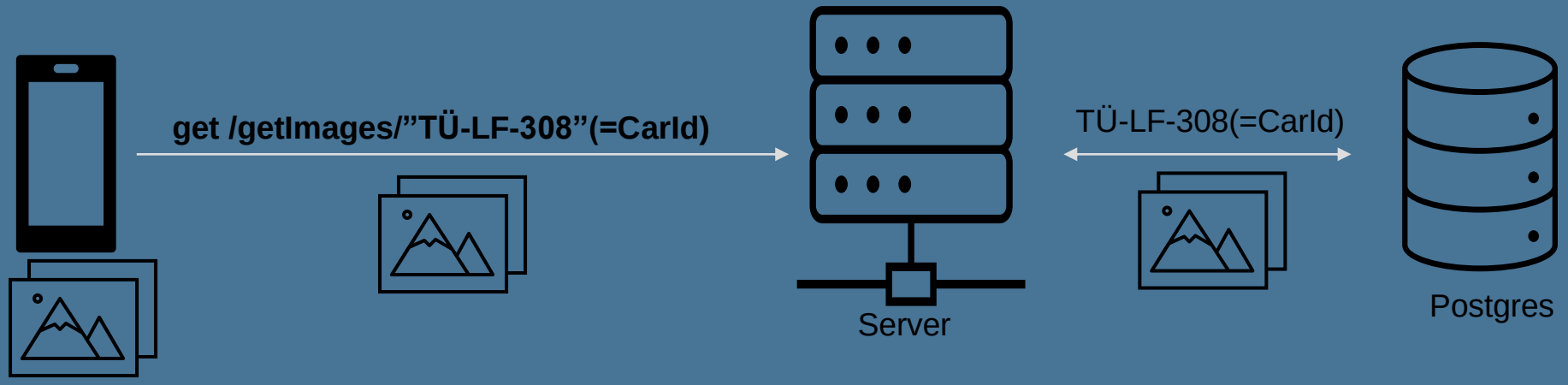
carchain

Carchain: Image-Server-Funktionen



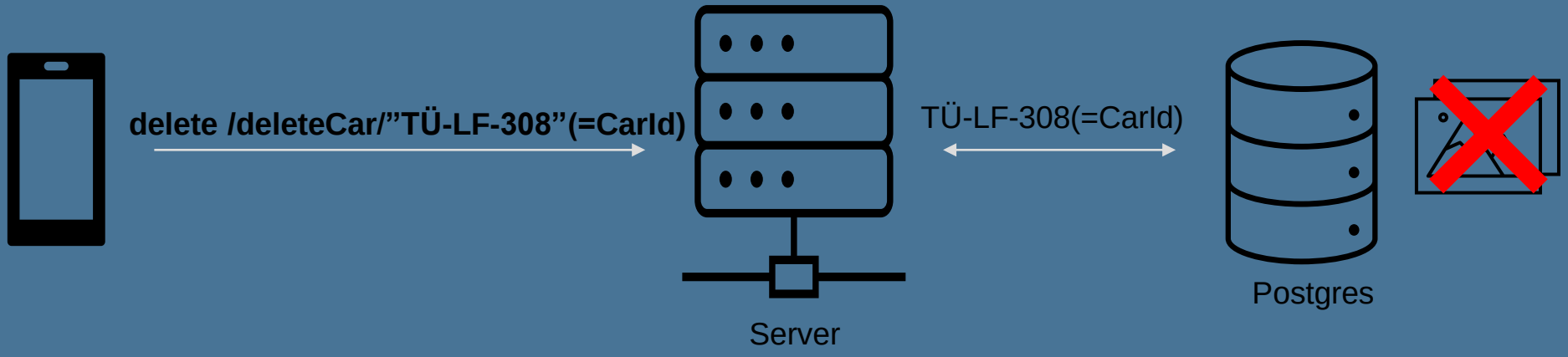
carchain

Carchain: Image-Server-Funktionen



carchain

Carchain: Image-Server-Funktionen



carchain

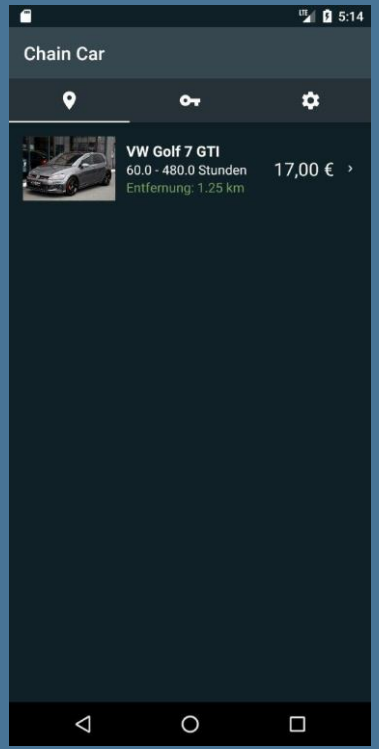
Carchain: Android-App

Funktionen der App:

- Vermieten von Fahrzeugen
- Suchen & Mieten von Fahrzeugen
- Rechnungen & Kosten einsehen (Wallet-Verwaltung)
- Digitaler Autoschlüssel
- Profilverwaltung

carchain

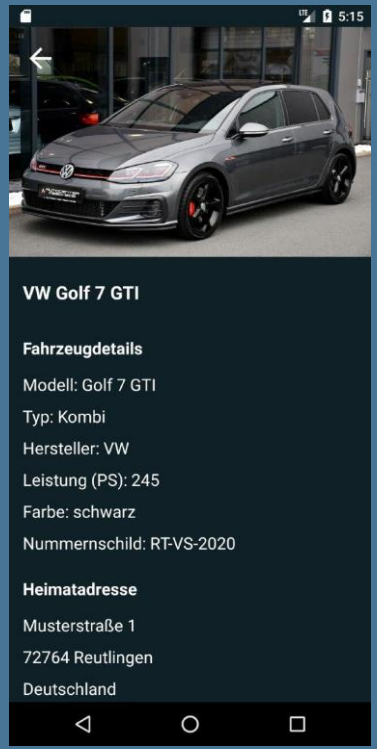
Carchain: Autos einsehen



- Autos können eingesehen werden
  - Filter nach:
    - Entfernung
    - Preis (pro Stunde)
    - Ausleihdauer (min. / max.)

carchain

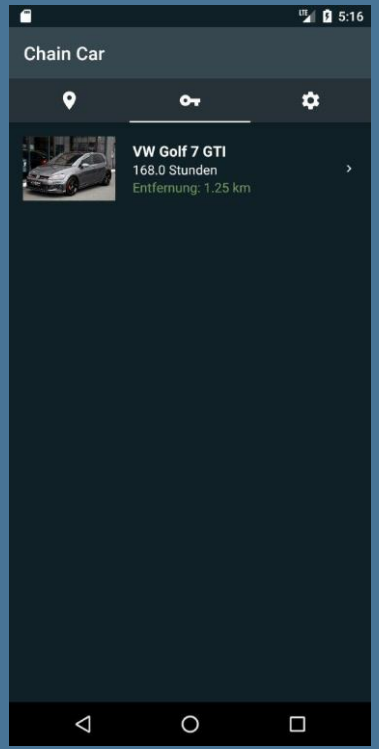
Carchain: Autos einsehen



- Detail-Ansicht zeigt interessante Daten zum Fahrzeug:
  - Modelldaten
  - Nummernschild
  - Heimatadresse
  - Preise (nach Mietlänge)

carchain

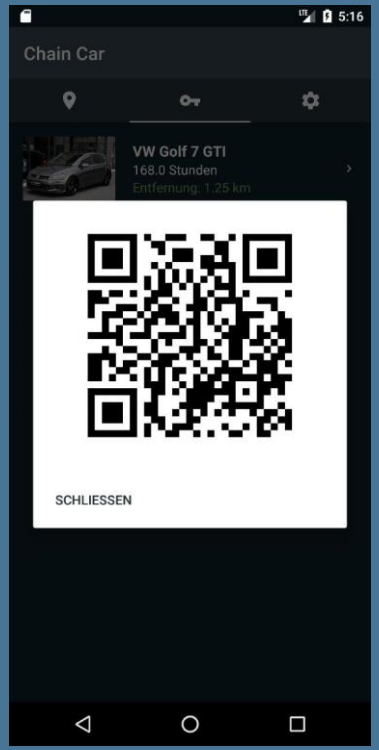
Carchain: Auto mieten



- Mit der Miete wird Auto verfügbar:
  - Digitaler Autoschlüssel freigeschaltet
  - Entfernung
  - Restmietdauer
  - …

carchain

Carchain: Auto nutzen



- Digitaler Autoschlüssel kann genutzt werden
- Am Auto vorhalten
  → Miete wird geprüft und Auto geöffnet

→ Nach Mietende nutzlos!

carchain

Carchain: Probleme der App

- Web3j Bibliothek für Android ist unausgereift:
  - Verbindung mit Blockchain kann hergestellt werden
  - Aufruf der Smart-Contract Methoden schlägt fehl: „Leerer Rückgabewert“
  - → Fehler in Konvertierung/Kommunikation mit eigenen Contracts
  - → Keine Anbindung an Blockchain möglich!

→ Alternative: Andere Library, Bugfix oder Routing über Server

Carchain: Blockchain Allgemein

- Ethereum
- Development Tools:
  - Truffle
  - Ganache bzw. Ganache-Cli
  - Visual Studio - Solidity Extension

Carchain: Smart Contracts

- Solidity
- Ähnlich: Java Script
- Möglichkeiten:
  - structs
  - constructors
  - destructors
  - functions
  - public/ private
  - uvm.

Carchain: Smart Contracts

- Besonderheiten:
  - views
  - Variablentypen wie address
  - Memory

```
function getModell(address identifierCar) public knownCar(identifierCar) view returns (string memory) {
  return carpool[identifierCar].modell;
}
```

- Optimierung:
  - Variablen
  - requires
  - modifier

```
modifier knownCar (address identifierCar) {
  require(carpool[identifierCar].owner != address(0), "Car is not in carpool");
  _;
}
```

carchain

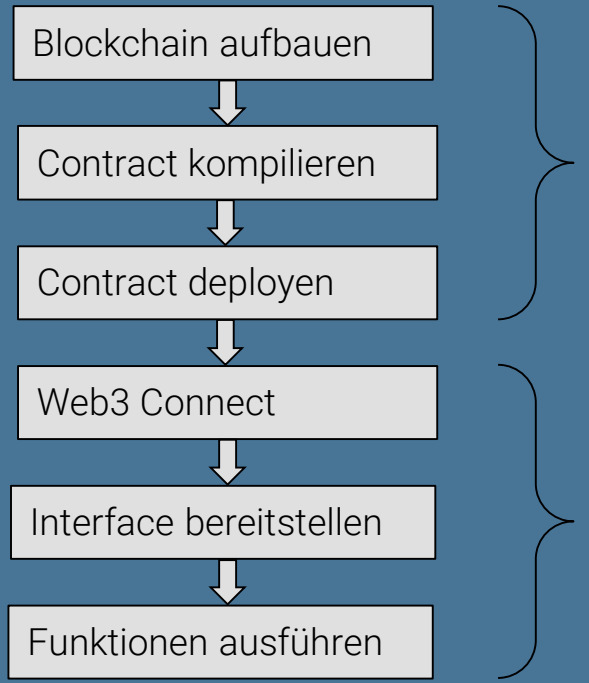
Carchain: Smart Contract Implementierung

- addCar
- removeCar
- rentCar
- mayRent
- isLegalLeaser
- returnCarToCarpool
- getAvaibleVehicles
- resetCars

```
struct Car{
  address owner;
  CarState currentState;
  address leaser;
  uint256 timeRented;
  uint256 amountEarned;
  int256 longitude;
  int256 latitude;
  string nummernschild;
  string modell;
  string typ;
  string hersteller;
  string farbe;
  uint256 ps;
  uint256 mietpreis;  //pro Minute
  uint256 maxMietdauer;
  uint256 minMietdauer;
}
```

carchain

Carchain: Funktionsweise



Commands

```
ganache-cli -m "dragon canoe knife need marine business
arctic honey make layer company solar"
-h "193.196.54.51" -p 8545 -e 10000 &
```

```
truffle migrate --network ganacheOnServer --reset
```

Programmiersprache

carchain

Carchain: Funktionsweise

```
let Web3 = require('web3')
let web3 = new Web3();
web3.setProvider(
    new web3.providers.HttpProvider('http://localhost:8545')
    );
```

```
let carchainStr = fs.readFileSync(__dirname + '/ethereum/build/contracts/carchain.json', 'utf-8');
let carchainAbi = JSON.parse(carchainStr);
let carchain = new web3.eth.Contract(carchainAbi.abi, contractAdress);
```

```
await carchain.methods.addCar(carWallet, "RT-VS-2020", "VW Golf 7 GTI", "Kombi", "VW", "schwarz", 245, 17, 480, 60).send({from: fromAddress, gas: 6000000});
```

carchain

Carchain: SmartCar (RaspberryPi)

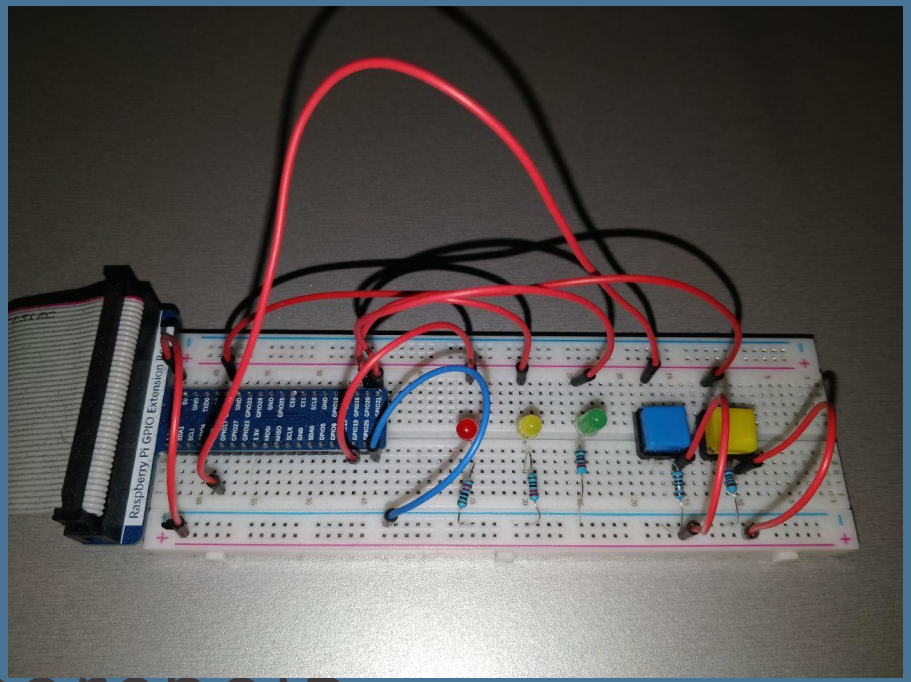
- Aufbau der Entwicklungsumgebung:

- Anschließen und aktivieren der Kamera
- Ausgedruckter physischer QR-Code
- T-Cobbler & Steckplatine mit 3 LEDs und 2 Buttons
- Test-Skripte (dev-ausleihen.js & dev-zurueckgeben.js)
- Externer Zugriff
  - OpenSSH enabled + Public Key Authentifizierung
  - Statische private IP am Pi (192.168.178.42)
  - DynDNS Eintrag an der FRITZ!Box (carchain-pi.dnsuser.de <-> Wechselnde öffentliche IP)
  - Portweiterleitung an der FRITZ!Box (Wechselnde öffentliche IP:22 <-> 192.168.178.42:22)

- → SSH-Zugriff auf Pi im lokalen Netz über carchain-pi.dnsuser.de möglich

Carchain: SmartCar (RaspberryPi)

Aufbau der Entwicklungsumgebung:

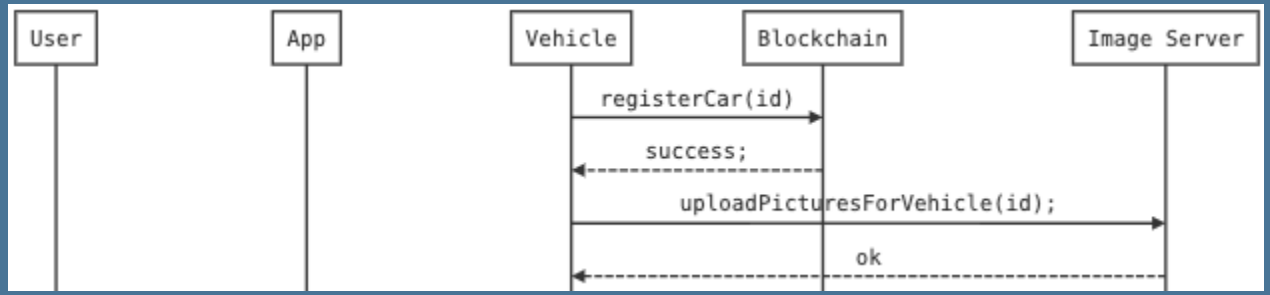


carchain

Carchain: SmartCar (RaspberryPi)

Funktionalität aus dem Pflichtenheft (NodeJS):

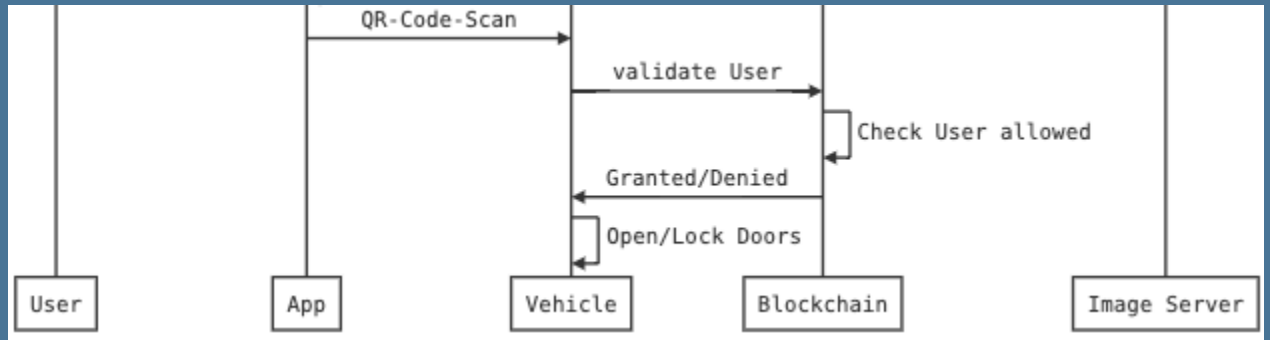
- Registrieren
  - OnOff-Modul für steuern der GPIO-Pins (LEDs+Buttons)
  - Bei Knopfdruck: Registrieren (später realisiert in Bereitstellungs-Pipeline, gleiche Funktion)
  - Einbinden der Web3-Schnittstelle
  - Nutzen der Smart-Contract Funktion: “addCar”
  - Picture Upload über HTTP-Put mit newman
  - register-collection.json für Beschreibung der HTTP-Anfrage



carcha

Carchain: SmartCar (RaspberryPi)

- QR-Lookup
  - Bei Knopfdruck: QR-Lookup → Gelbe “In Bearbeitung”-LED
  - Aufnahme und speichern eines Fotos mit Pi-Camera-Modul
  - Suchen nach QR-Code mit Qrcode-Reader-Modul
  - Falls Wallet-Adresse gefunden: Abfragen ob existent an Blockchain (Web3)
  - Nutzen der Smart-Contract Funktion: “isLegalLeaser”
    - True: Grüne LED = Offen (+Gelb aus)
    - False: Rote LED = Geschlossen (+Gelb aus)
  - 15 s Intervall-Schleife im Hintergrund: Überprüfen von “isLegalLeaser” + LED-Steuerung



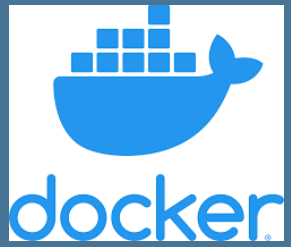
carcha

Carchain: SmartCar (RaspberryPi)

Automatisierte Bereitstellungs-Pipeline (Ansible)

- Konfiguration von Ansible
- Anlegen eines Inventory
- Schreiben eines Playbooks bestehend aus Tasks:
  - Installieren von NodeJS, NPM und dem Node-Exporter über APT + enable Node-Repository
  - Kopieren des privaten SSH-Schlüssels für Zugriff auf Git
  - Konfiguration von OpenSSH (durch ssh_config.j2) & ssh-keyscan git.smagcloud.de
  - Klonen des Repos (git@git.smagcloud.de:DHBW17B/carchain.git)
  - Installieren der benötigten NPM-Pakete (aus npm-requirements.txt)
  - Kopieren des angelegten Unit-Files (durch car_js.service) für automatisches Starten
  - Konfiguration von Systemd (car_js.service & node-exporter starten + enablen)
  - Ausführen von register_car.js (Registrieren des PIs an der BC + Bild-Upload an DB)

Carchain: SmartCar (RaspberryPi)



Monitoring (Prometheus & Grafana)

- DNS-Eintrag des Servers: carchain-server.tk <-> 193.196.54.51
- Betrieben mit Docker, Volumes für persistenten Speicher
- Cronjob: Starten nach Reboot des Servers
- Prometheus: TSDB, Sammelt Metriken über HTTP
  - Konfiguration über YAML-Datei
  - Scrape-Intervall: 15 Sekunden (carchain-pi.dnsuser.de)
  - http://carchain-server.tk:9090/status
- Node-Exporter auf dem RaPi: Abfragen der Metriken
  - Bereitstellen der Metriken als HTTP-Endpunkt auf Port 9100
  - http://carchain-pi.dnsuser.de:9100/metrics
- Grafana: Visualisierung über Dashboards und Alerts
  - http://carchain-server.tk:3000/d/rYdddlPWk/rapi-carchain

carchain

Carchain: SmartCar (RaspberryPi)

Monitoring (Prometheus & Grafana)

carchain

Ergebnis: Carchain - Stand

- Blockchain:
- Image-Server:
- Car-Provisioning:
- App-Anbindung:

carchain

Demo

carchain

Fazit

- Interessantes, nicht triviales Projekt
- Blockchain: spannende Technologie – noch nicht ganz ausgereift
- Diverse spannende Anwendungsgebiete
- Teilweise komplexere Umsetzung
- Schwerer zu debuggen

carchain

carchain

Vielen Dank!