# Blockchain Development

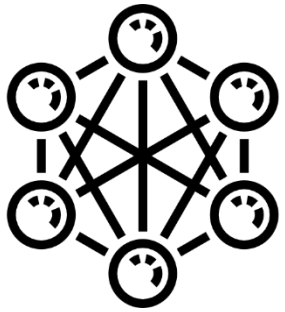
## Was brauche ich und wie fange ich an?

**Gregor Sachnik**

# Blockchain
## Grundlagen

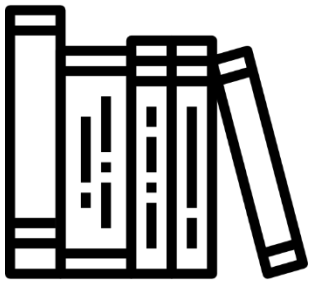
# Was muss ich an Können und Wissen mitbringen?

## Blockchain



Was ist sie?
Wie funktioniert sie?
Was kann sie?
& etc

## Wissen

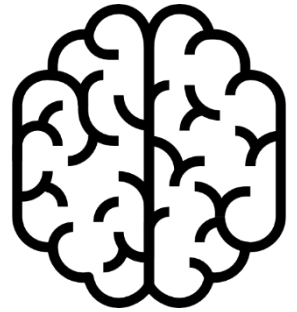


Block, Hash,
GasPrice, ABI, JSON,
Digital Twin, Dapp,
Smart Contract
& etc

## Informatik

IT Grundlagen,
Umgang mit einem
OS und Software,
Programmierskills,
Googleskills
& etc

## Allgemein

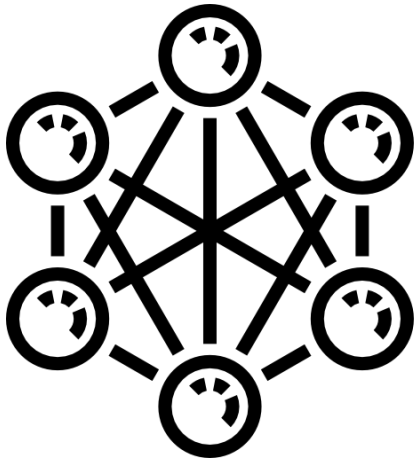


Abstraktes Denken,
Reflektion von
Wissen,
Auffassungsgabe
& etc

# Blockchain - Warum?

## Warum eine Blockchain nutzen?

Die Blockchain will es ermöglichen, dass ein **grenzenloser Austausch** von Informationen, Gütern und Werten stattfinden kann.



Dabei soll jedoch auf einen klassischen **Intermediär** (Konzerne, Banken oder Regierungen) verzichtet werden, der Transaktionen und Verträge auf ihre Richtigkeit überprüft.

Diese Aufgabe übernimmt die Eigenschaft der Blockchain. **Nutzer signieren Transaktionen** von anderen Nutzern auf ihre Richtigkeit.

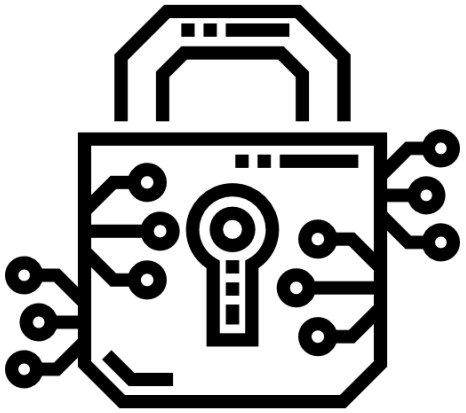
Ebenfalls **überwachen alle Nutzer** die Blockchain auf Einhaltung der gesetzten Regeln und einer nicht Manipulation dessen.

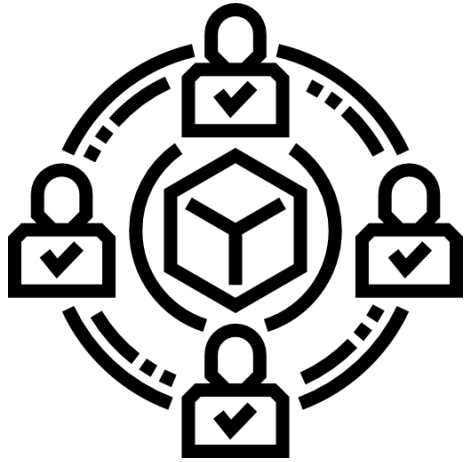
# Blockchain (Definition)

## Was ist eine Blockchain?

Kurz geschrieben handelt es sich bei der Blockchain um ein **dezentrales Peer-to-Peer-Netzwerk**.

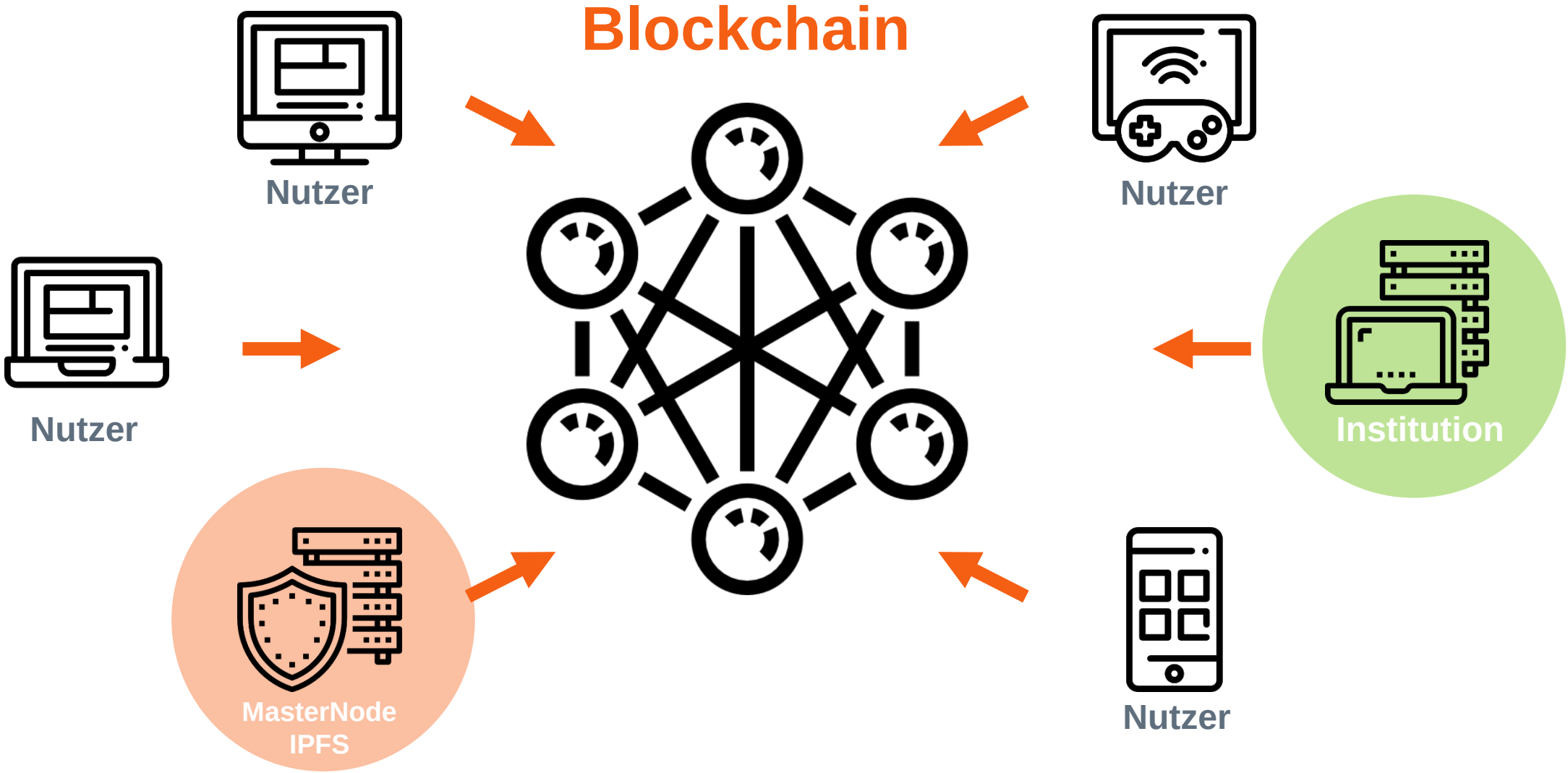
Mit Hilfe von **Kryptografie** wird ein **sicheres, transparentes** und wahlweise **anonymes** System generiert.





Das Vertrauen der Nutzer in das System wird obsolet, da jeder Nutzer die anderen Nutzer im System überwacht.

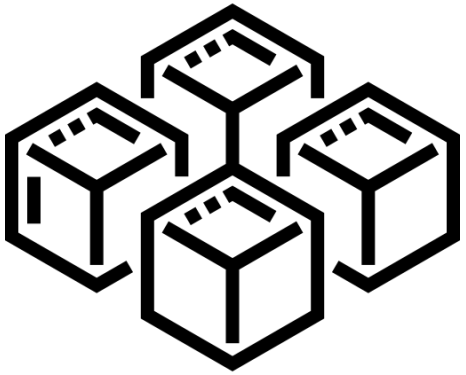
# Blockchain - Skizze



# Blockchain (Beschreibung)

## Was ist eine Blockchain?

Eine Blockchain ist eine **dezentrale Datenbank** die auf einer **Kettenstruktur** basiert. Ein sogenannter **GenesisBlock** bildet den Anfang und weitere **Blöcke** werden nacheinander hinten angehängt.



Der Clou dabei ist, dass jeder Block nur sich selbst und seinen Vorderblock in der Struktur kennt. Mit einziger Ausnahme des GenesisBlocks. Dadurch entsteht eine Unveränderbarkeit der fortwährenden Kettenstruktur.

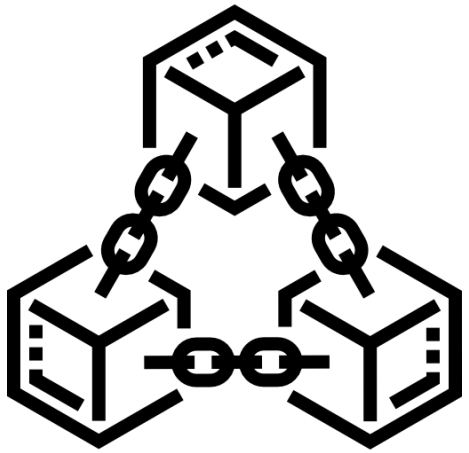
Die Blockchain liegt bei jedem **teilnehmenden Benutzer als Abbild** vor.

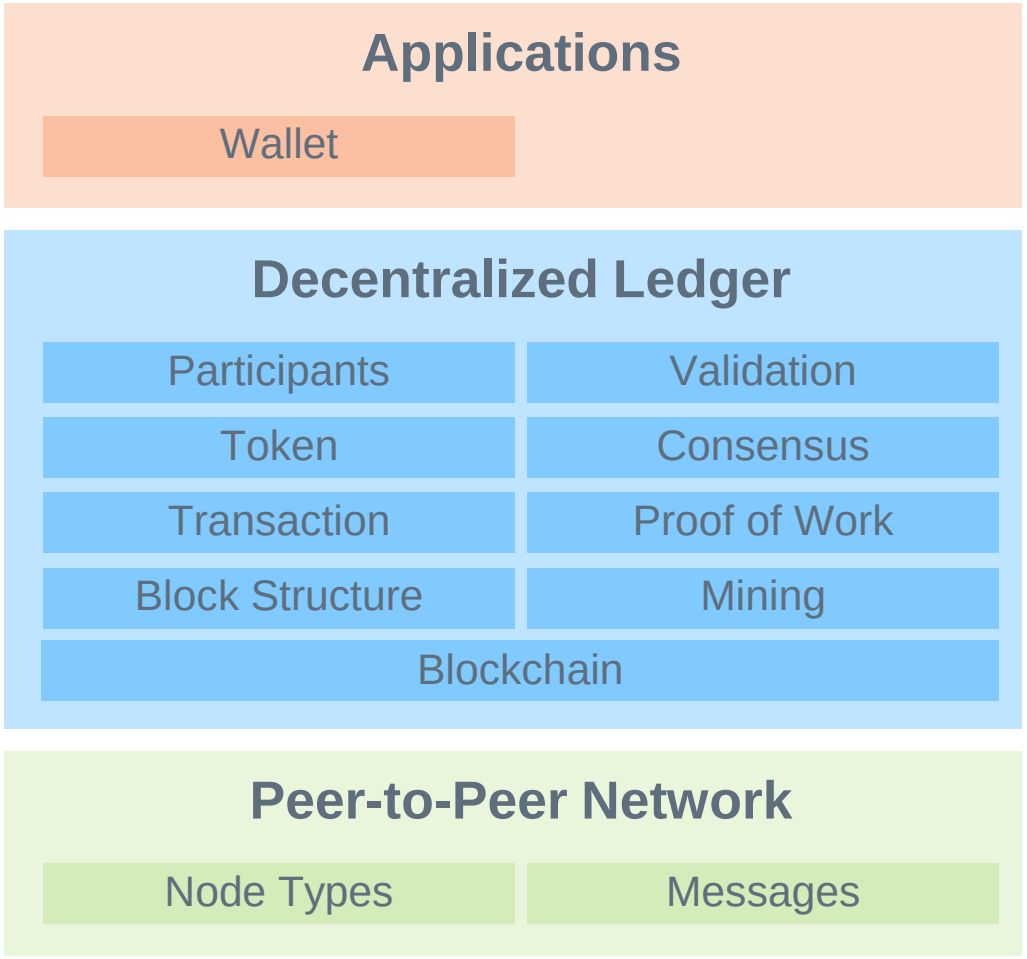
Dadurch **löst sie ein Ver-/ Misstrauensproblem** unter den Nutzern.

# Blockchain - Architektur

## Wie ist die Architektur aufgebaut?

Die Ebene **Applications** kann mit Hilfe der **Decentralized** Ebene, die auf der **Peer-to-Peer** Ebene aufbaut, erstellt werden.



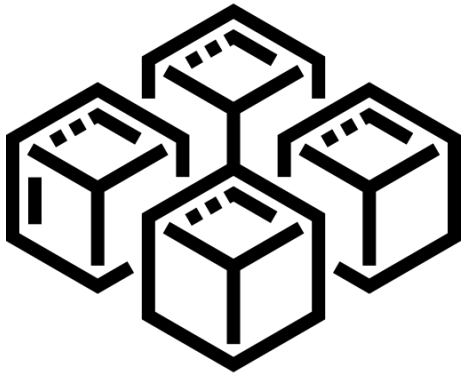


# Blockchain - Blöcke

## Wie ist ein Block aufgebaut?

Ein Block besteht aus **mehreren Informationskomponenten**.

Diese alle zusammen werden in einen **Hash** für den gesamten Block zusammengefasst.



## Was ist ein Block aufgebaut?

**BlockHeader**

- BlockNumber
- Hash
- PreviousHash
- Timestamp
- … und weiteres
- TransactionRoot
- StateRoot
- ReceiptsRoot

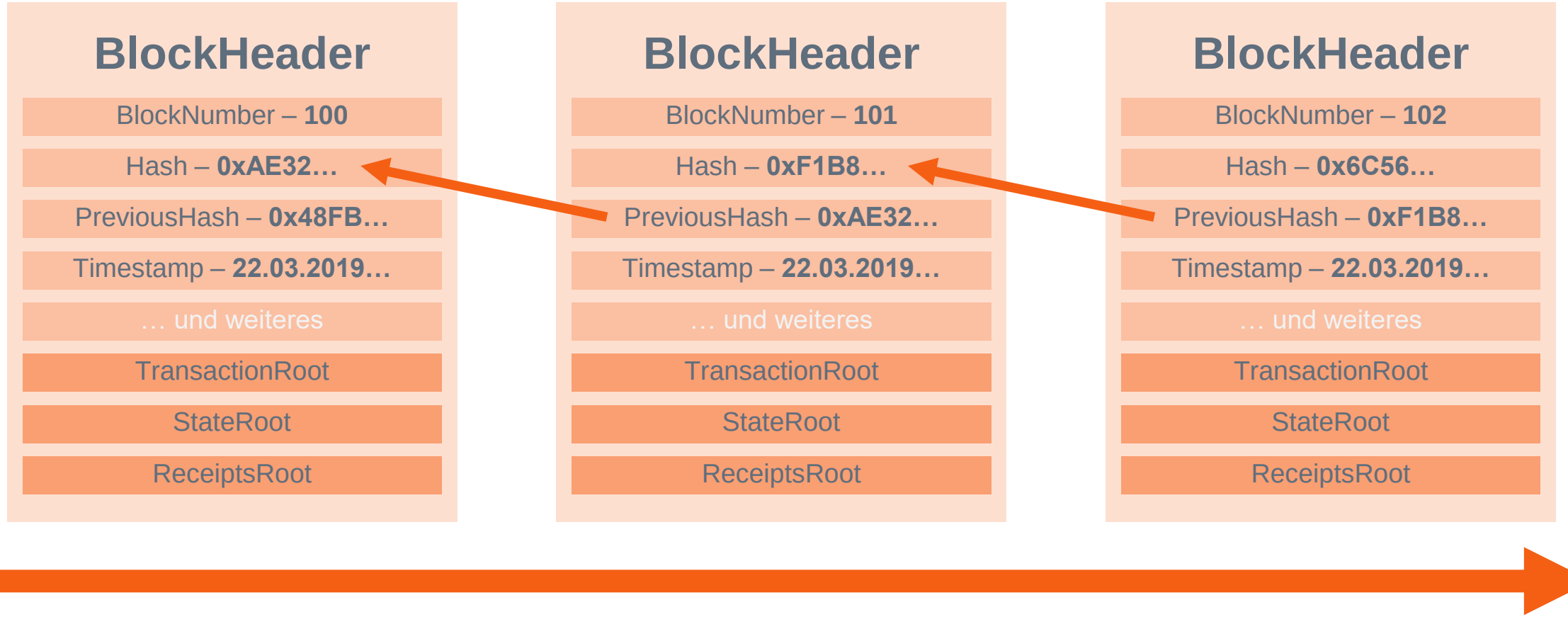
- TransactionRoot

Sind wie eine Art Laderäume, die nach dem **MerkleTree** Prinzip funktionieren.

- StateRoot
- ReceiptsRoot

Funktionieren nach dem **Patricia-Trie**.

# Blockchain - Kettenstruktur



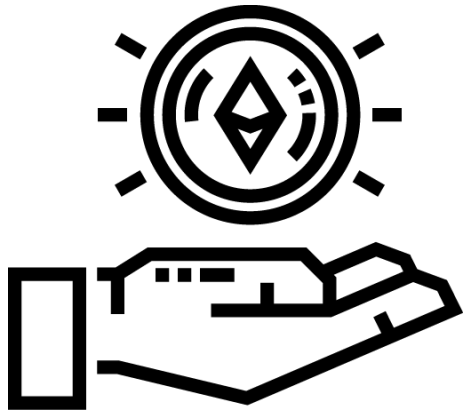

**Blockchain - Kettenstruktur**

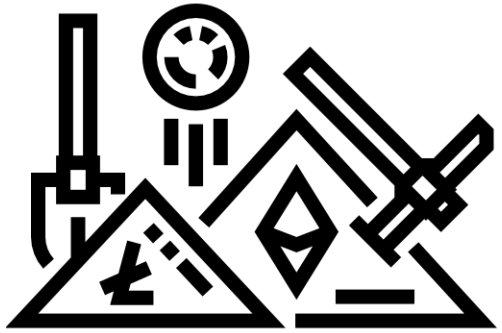
# Blockchain - Mining

## Was ist Mining?

**Proof-of-Work**:

In einem Wettstreit versuchen **Miner**, Nutzer mit spezieller Hardware zur Überprüfung von Trans-aktionen, ein mathematisches Rätsel zu lösen.





Nach erfolgreichem Lösen, werden dann die Transaktionen (**Gas**) validiert und neue Coins generiert.

Als Bonus für seine Arbeit bekommt der Miner eine Belohnung in Form von Coins.

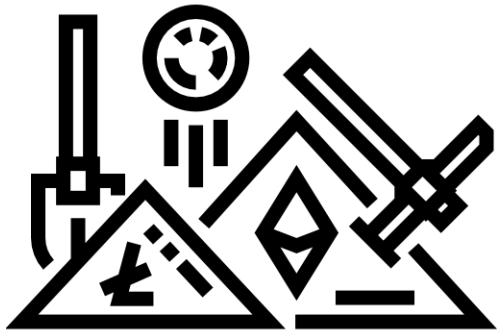
# Blockchain - Validatoren

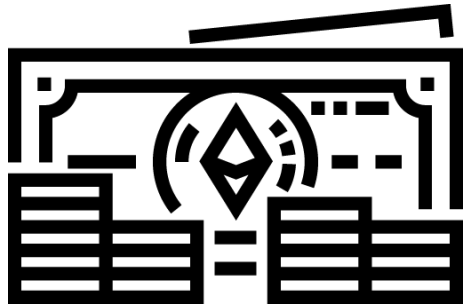
## Sind Validatoren eine Alternative?

**Proof-of-Stack**:

**Validatoren** ersetzen Miner und agieren mit allen Rechten und Pflichten auch wie diese.

Ausnahme, die Validatoren legen einen Teil ihrer Coins als **Stake** (Einstatz) zur Seite.





Der Stake soll vor betrügerischen Verhalten schützen. Eine Bestrafung führt zum Verlust eines Teils seines Stacks.

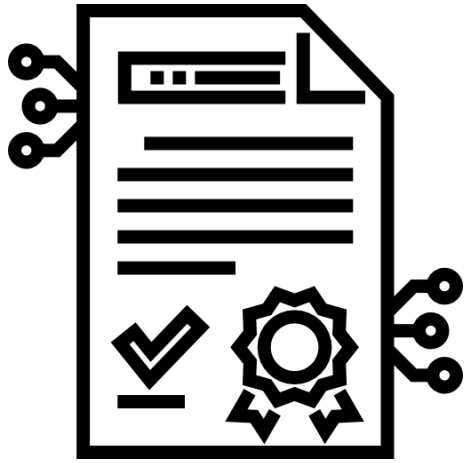
Es gilt auch, je höher der Stack, desto höher die Chance als Validator genommen zu werden.

# Ethereum – Smart Contract

## Was ist ein Smart Contract?

Ein Smart Contract ist ein determinierter **digitaler Vertrag**. Wie in der realen Welt werden im SM **ausgehandelte Regeln** niedergeschrieben. Mit dem Unterschied, dass je nach festgelegtem Ereignis im SM, sich diese **automatisch ausführen**.

In der Blockchain wird ein SM dem Status eines Nutzer gleichgesetzt. Damit andere Nutzer in der Blockchain diese auf Einhaltung und dessen nicht Manipulation überwachen können.



## Vorteile:

- Strikte Einhaltung der Regeln
- Automatisch ausgeführt (Trigger)
- Können eine Lebensdauer bekommen
- Sparen Zeit und Geld

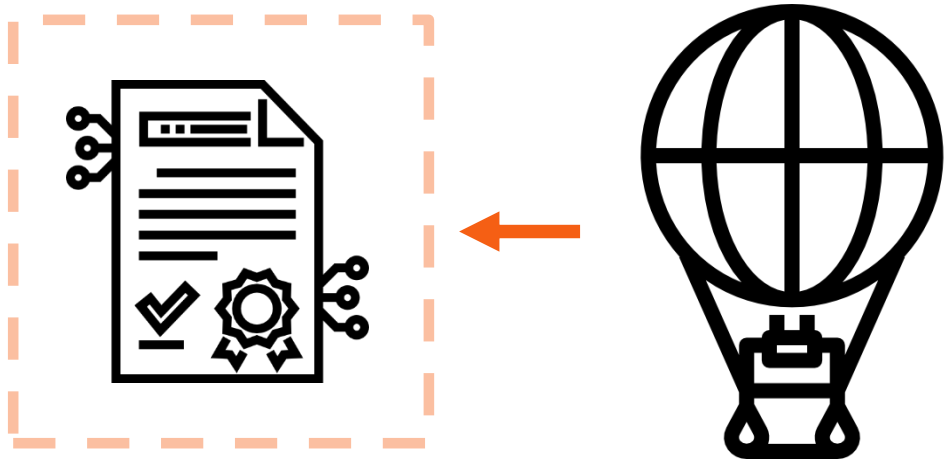
# Ethereum – Digital Twin

## Was ist ein Digital Twin?

Ein Digital Twin ist ein **digitales Abbild** eines in der realen Welt existierenden Objektes.

Idealerweise soll ein DT **einmalig in der Blockchain existieren**, damit dieser einzigartig bleibt.

Wie der Smart Contract wird der DT dem Status eines Nutzer gleichgesetzt. Damit andere Nutzer in der Blockchain diese auf nicht Manipulation überwachen können.



## Vorteile:

- Bildet ein Objekt digital ab
- Sofern einmalig, kann eine fortlaufende Historie bekommen
  - Mehre Nutzer können auf das gleiche reale Objekt referenzieren (Versicherungen, Banken)

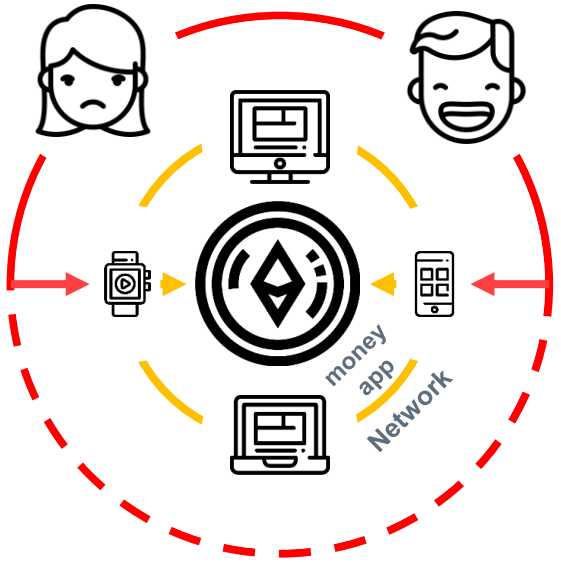
# Ethereum – Dezentralized Application

## Was ist eine DApp?

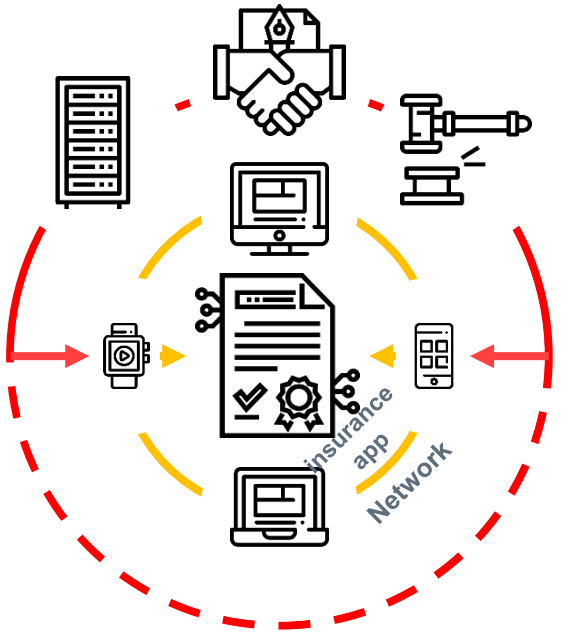
Eine Desktop oder Web **Application** die mit der Blockchain interagiert.

Man unterscheidet in drei Typen:

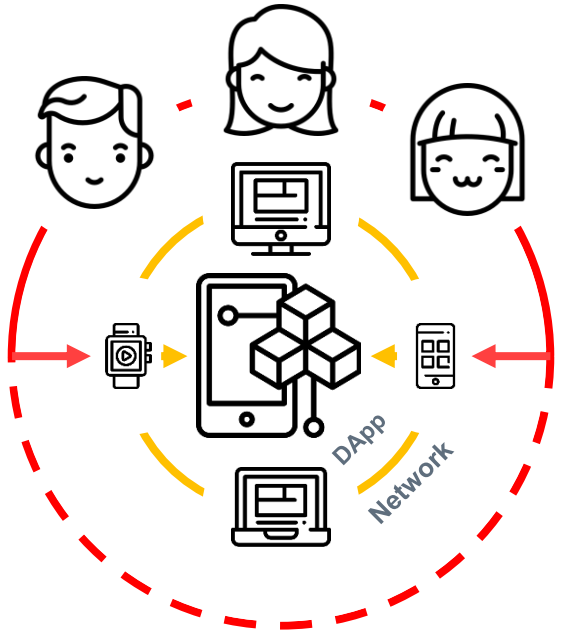
- Apps die Geld verwalten
- Apps wo Geld involviert ist
- Apps die Wahlen-, Steuerungs- oder Rechteelemente beinhalten




Peer-to-peer transactionen




external authorities (oracles)




interaction with blockchain

# Entwicklung
## Ethereum-Blockchain

# Programmieren – Was muss ich für eine WebDApp wissen?

## Reichen meine Skills?

**Frontend:**

- HTML & CSS (Web DApp)

**Backend:**

- Javascript (web3.js)
- Java (web3.j)
- Python (web3.py)
- C
- Go
- Node.js

**Betriebssystem**

- Windows, MacOS oder Linux

**Software & Tools**

- Web3 (Bibliothek)
- Truffle (Framework)
- Ganache (lokale Test-Blockchain)
- Parity & Geth (lokale Wallets)
- MetaMask (Browser Wallet)
- Lite-Server & Browser-Sync (Live-Server)

**IDE**

- Remix (IDE + virtuelle Test-Blockchain)
- VS Code, Atom, Webstorm oder etc

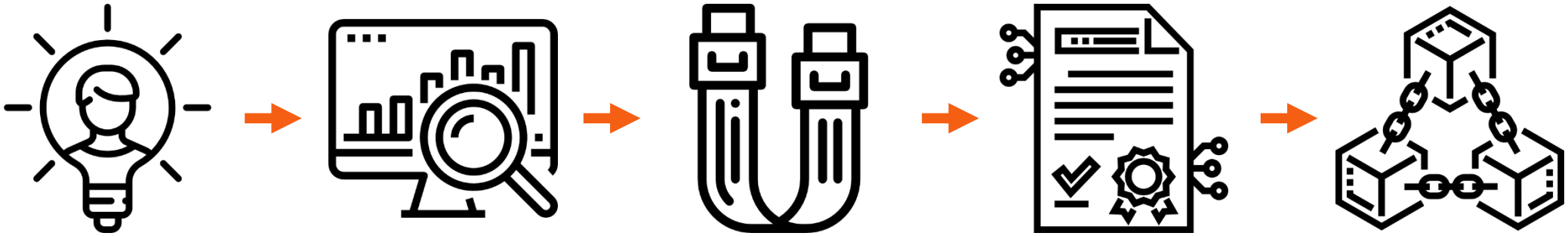
# Entwicklung

## Wie fange ich an?

Vorab sollte man sich im klaren sein, dass eine Blockchain **kein bestehendes System zu 100% ablösen kann**. Eine Integration der Blockchain sollte eher als **Erweiterung angesehen** werden, wo „Dinge“ **für immer persistent gespeichert** werden.

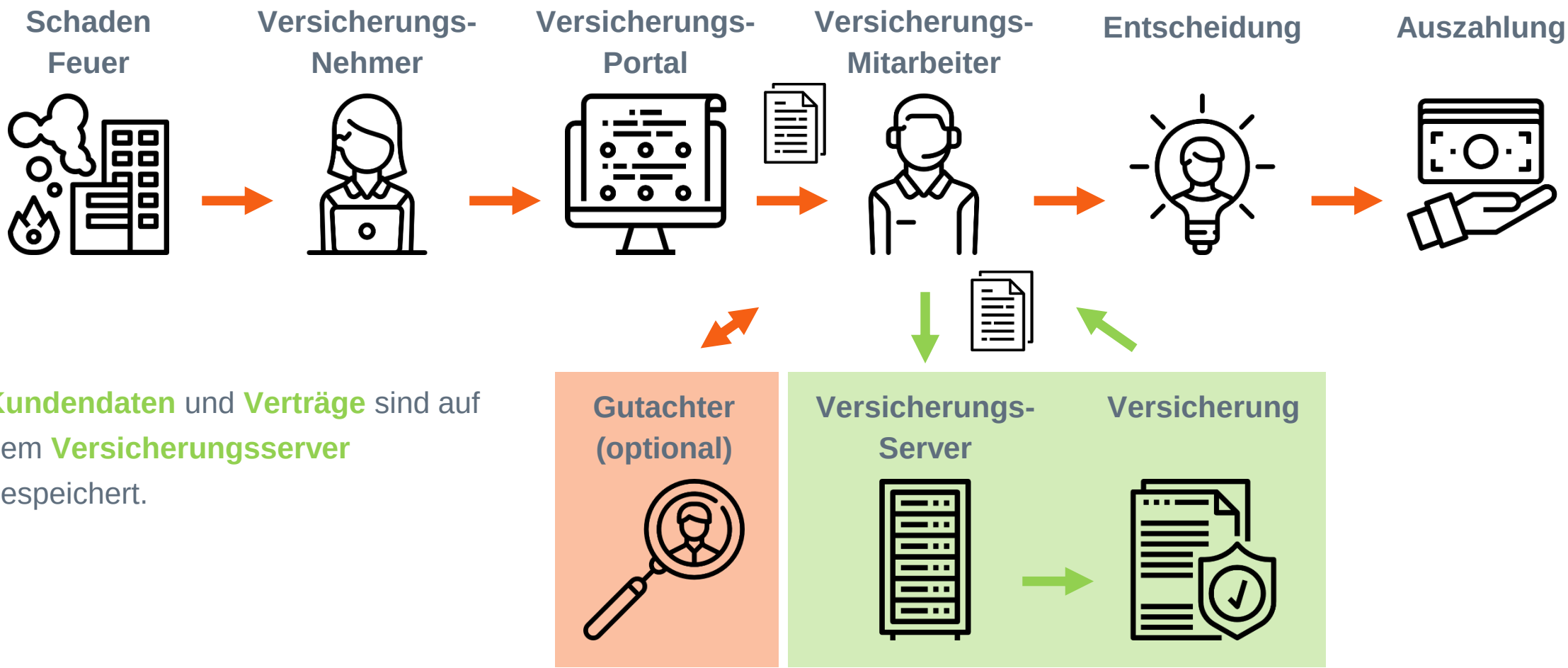
- Business Logic ausarbeiten

  (Was macht Sinn für immer in die Blockchain zu speichern?)
- (Bestehendes System analysieren)
- Schnittstelle für sinnige Integration finden oder bauen
- Smart Contract und etc Ideen coden und in die Blockchain deployen
- System an die Blockchain anbinden



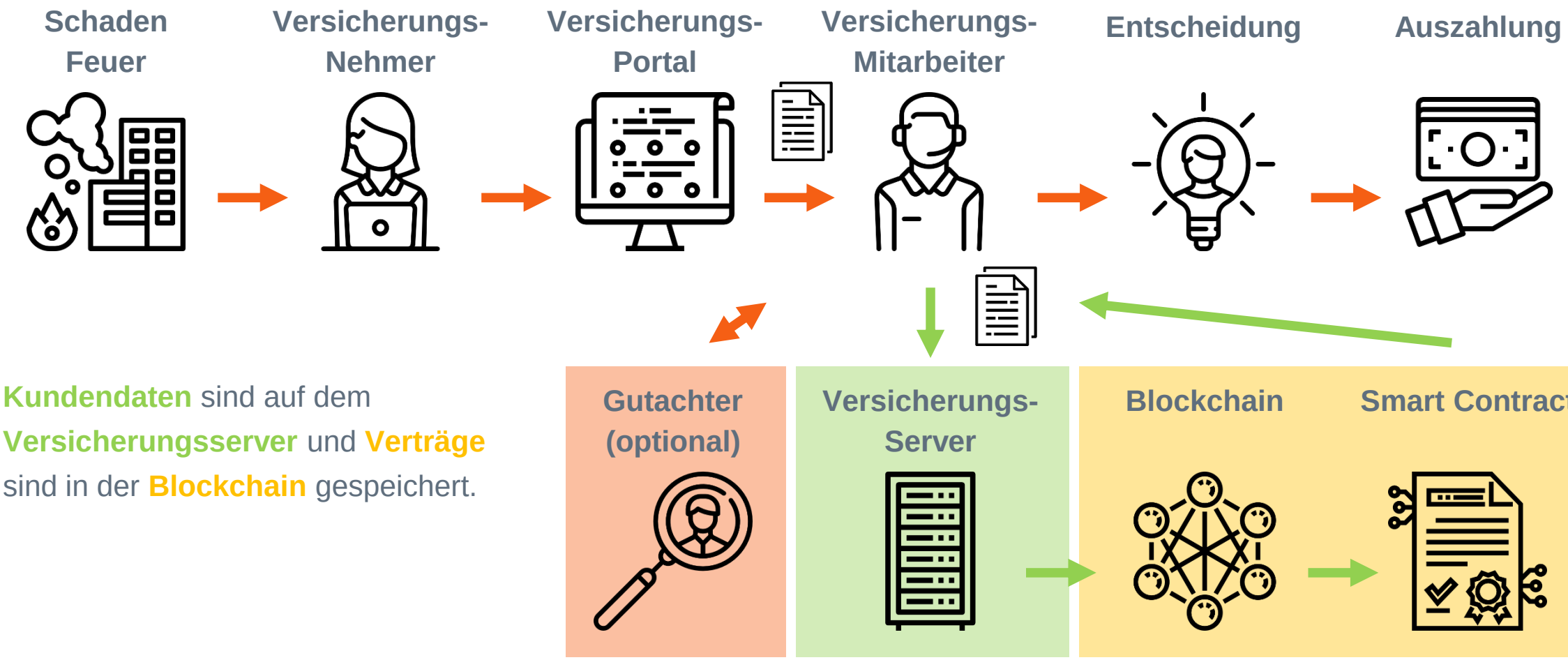
# Beispiel
## Versicherung

# Entwicklung – Use Case Beispiel



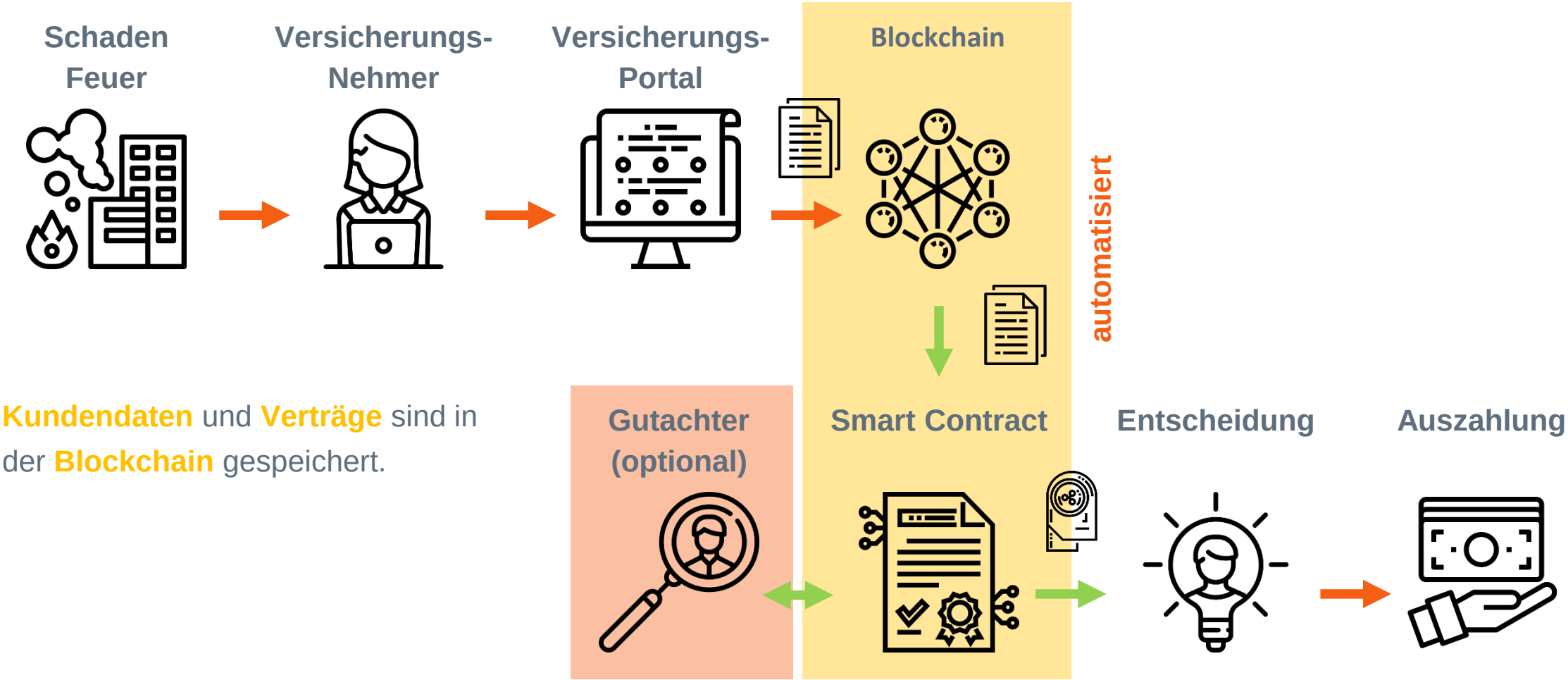

**Kundendaten** und **Verträge** sind auf dem **Versicherungsserver** gespeichert.

# Entwicklung – Use Case Beispiel mit Blockchain (Hybrid-Lösung)




**Kundendaten** sind auf dem **Versicherungsserver** und **Verträge** sind in der **Blockchain** gespeichert.

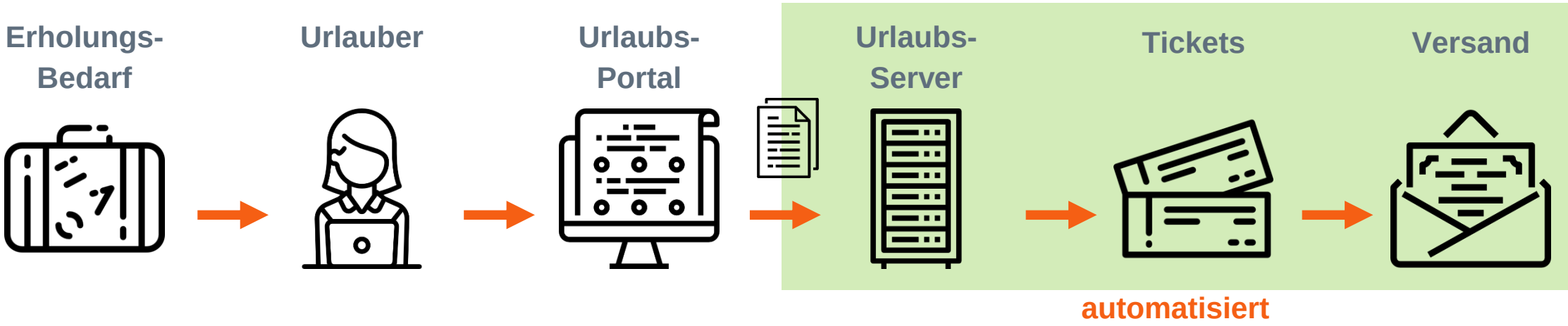
# Entwicklung – Use Case Beispiel mit Blockchain (Voll-Lösung)




**Kundendaten** und **Verträge** sind in der **Blockchain** gespeichert.

# Beispiel
## Reiseveranstalter

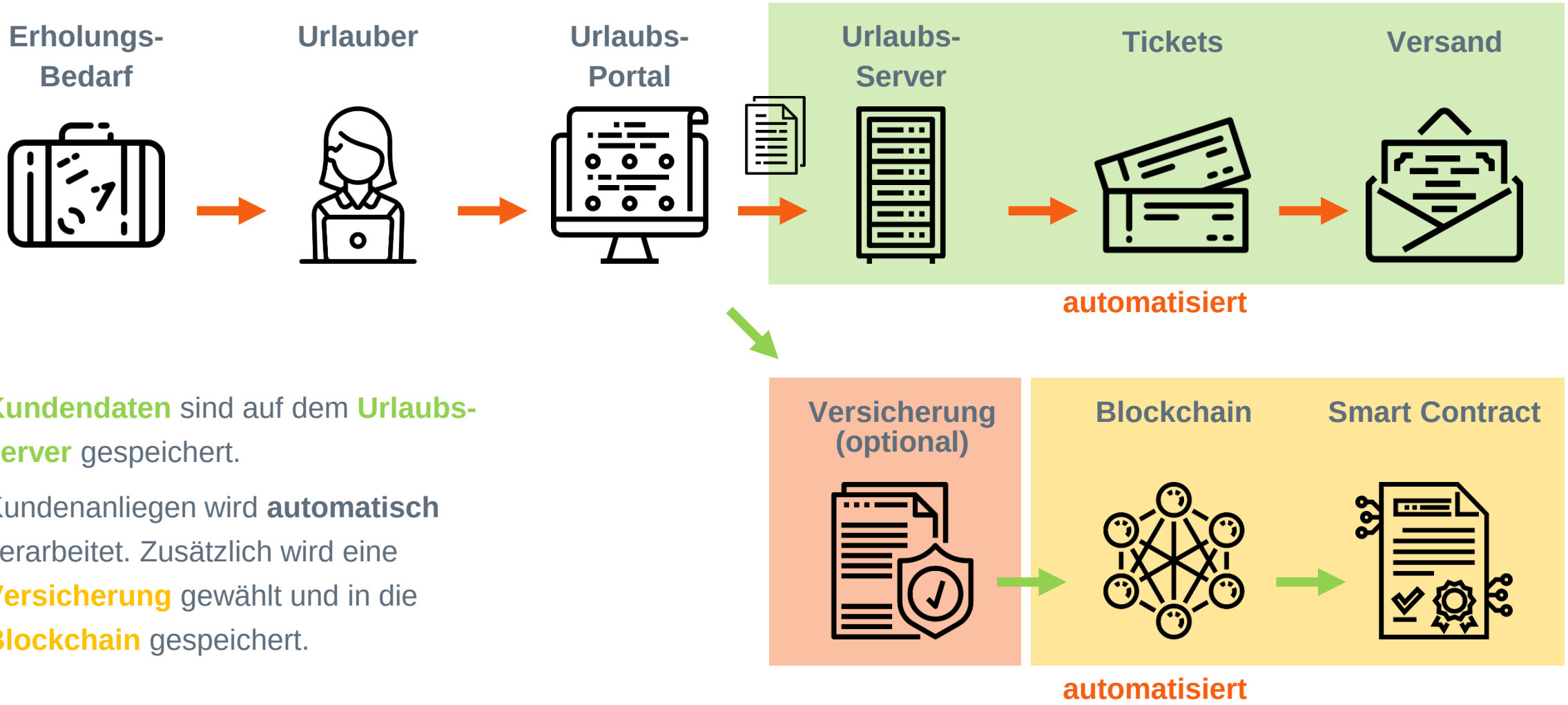
# Entwicklung – Use Case Beispiel




**Kundendaten** sind auf dem **Urlaubs-server** gespeichert.

Kundenanliegen wird **automatisch** verarbeitet.

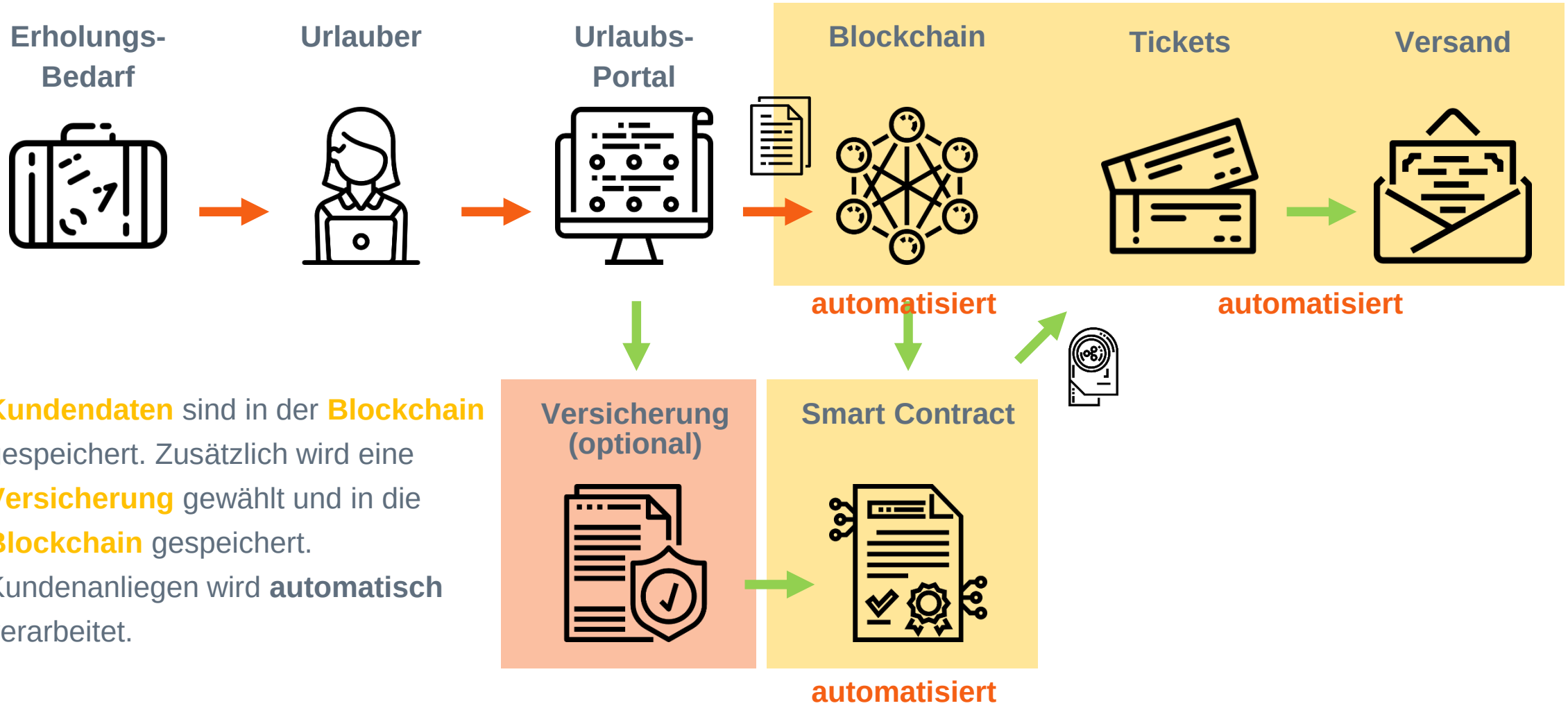
# Entwicklung – Use Case Beispiel mit Blockchain (Hybrid-Lösung)




**Kundendaten** sind auf dem **Urlaubs-server** gespeichert.

Kundenanliegen wird **automatisch** verarbeitet. Zusätzlich wird eine **Versicherung** gewählt und in die **Blockchain** gespeichert.

# Entwicklung – Use Case Beispiel mit Blockchain (Voll-Lösung)




**Kundendaten** sind in der **Blockchain** gespeichert. Zusätzlich wird eine **Versicherung** gewählt und in die **Blockchain** gespeichert. Kundenanliegen wird **automatisch** verarbeitet.

# Development Setup
## Einrichtung

# Development Setup – Linux

- **1 – Linux-OS-Auswahl**
  Getestete und produktive Linux-OS:
  Ubuntu 18.04, **Ubuntu 18.04.2** und Ubuntu 18.10
- **2 – Software**
  node.js (10.15.*), Web-IDE mit Solidity-Plugin
  (VS Code, Atom, Sublime Text3 und Webstorm)
- **3 – npm Rechte reparieren/setzen für global**
  NPM fix Permissions
- **4 – (global) apt Installationen (Packages)**
  Im **Terminal**

```
sudo apt update
sudo apt -y upgrade
sudo apt install -y git
sudo apt install -y build-essential
sudo apt install -y python
```

- **5 – (global) npm Installationen (Packages)**
  Im **Terminal**

```
npm install -g truffle
npm install -g lite-server
npm install -g browser-sync
```

- **6 – Ganache (lokale Testumgebung)**
  Ganache
- **7 – BrowserWallet**
  MetaMask

# Development Setup – Windows

- **1 – OS-Auswahl**
  Getestete und produktive OS-Versionen:
  Windows 10
- **2 – Software**
  node.js (10.15.*), git (aktuell),
  Web-IDE mit Solidity-Plugin
  (VS Code, Atom, Sublime Text3 und Webstorm)
- **3 – Tools**
  Python (2.7.16)
  Microsoft Build Tool 2015
- **4 – (global) npm Installationen (Packages)**
  Im **Terminal** oder **Eingabeaufforderung**

```
npm install -g truffle
npm install -g lite-server
npm install -g browser-sync
```

- **5 – Ganache (lokale Testumgebung)**
  Ganache
- **6 – BrowserWallet**
  MetaMask

**Randinfo:**

Beim ersten Versuch truffle mit npm unter MacOS zu installieren, kommt ein Fehler und es wird angeboten Xcode-Software zu installieren. Diese Software installieren. Danach klappt alles.

# Development Setup – MacOS

- **1 – OS-Auswahl**
  Getestete und produktive OS-Versionen:
  MacOS (> 10.6.8)
- **2 – Software**
  node.js (10.15.*), git (aktuell),
  Web-IDE mit Solidity-Plugin
  (VS Code, Atom, Sublime Text3 und Webstorm)
- **3 – npm Rechte reparieren/setzen für global**
  NPM fix Permissions
- **4 – (global) npm Installationen (Packages)**
  Im **Terminal** oder **Eingabeaufforderung**

```
npm install -g truffle
npm install -g lite-server
npm install -g browser-sync
```

- **5 – Ganache (lokale Testumgebung)**
  Ganache
- **6 – BrowserWallet**
  MetaMask

**Randinfo:**

Beim ersten Versuch truffle mit npm unter MacOS zu installieren, kommt ein Fehler und es wird angeboten Xcode-Software zu installieren. Diese Software installieren. Danach klappt alles.

# github
## Repository

# github - Repository

edicos

**https://github.com/x37ts52/guestbook_for_blockchain**

# Kontakt

# Edicos Kontakt

## Ihr Ansprechpartner


Gregor Sachnik

Frontend Developer
Tel.: +49 (0) 511 / 475395 – 32
gregor.sachnik@edicos.de

edicos Consulting & Software GmbH & Co. KG
Leisewitzstr. 4
30175 Hannover
Deutschland
Tel.: +49 (0) 511 / 475395 – 0



www.edicos.de