

VoteForYourself – The Voting System for everyone based on the Blockchain Technology

VORTWORT

Die Idee für die Umsetzung eines E-Voting Systems in Form einer Blockchain ist das Resultat aus dem umstrittenen Artikel 13 des europäischen Urheberrechtsgesetzes. Die Akzeptanz des Artikels sorgte in diversen Plattformen des Internets (Reddit, Facebook, etc.) für Aufruhr seitens der User (U/LongLiveAnarchyAcres 2019). Axel Voss, ein Mitglied des europäischen Parlaments hatte sich für die Reform der Datenschutz Grundverordnung ausgesprochen und mit dieser Entscheidung für einen Aufruhr im gesamten Netz gesorgt. Eine Aussage der empörten Gesellschaft ist, dass ein Mann, der das Internet nicht versteht, entscheiden soll, wie wir anderen es zu nutzen haben (Stresing 2019). Diese Aussage brachte mich auf die Idee mit der Umsetzung eines E-Voting System mit dem die Nutzer selbst die Möglichkeit bekommen entscheiden zu können. Dabei bietet sich der Einsatz der Distributed Ledger Technologie (DLT) an. Die DLT ist eine Form der Datenspeicherung bei der es keine zentrale Datenbank gibt, sondern die Datenbank auf mehrere Standorte/Regionen/etc. verteilt ist (Azimdoust 2019).

Eine der Haupteigenschaften einer Blockchain ist, dass die Daten eines Blocks nicht verändert werden können, ohne die gesamte Blockchain zu manipulieren. Diese Eigenschaft ermöglicht es eine Wahlmanipulation/-betrug vorzubeugen und ist ein weiterer Punkt für den Vorteil eines E-Voting System. Die Wichtigkeit des Schutzes gegen Wahlmanipulation/-betrug wurde in mehreren Fällen der letzten Jahre bestätigt, betrachtet man die diversen Anschuldigungen von Wahlbetrug diverser Länder, wie z.B. Russland (Hebel 2018).

Im Folgenden werden ich Anforderungen an ein Wahlsystem festlegen, die Eigenschaften einer Blockchain definieren, eine Architektur für mein System erstellen, Klassen definieren, darlegen, wie ich die Eigenschaften der Blockchain in meiner Anwendung umgesetzt habe, den typischen Ablauf einer Wahl präsentieren und im Anschluss meine Ergebnisse mit den Anforderungen an ein Wahlsystem vergleichen. Dabei ist anzumerken, dass ich von utopischen Verhältnissen ausgehe, in denen ohne Probleme mit dem jeweiligen Gesetz die Anwendung umgesetzt werden kann. Abschließen werde ich die Dokumentation mit Optimierungsvorschlägen, wie die Anwendung verbessert werden kann.

Anforderungen an ein Wahlsystem

Im folgenden Abschnitt definiere ich die Anforderungen an ein Wahlsystem (Hjálmarsson and Hreiðarsson 2019):

1. Eine Stimme darf nicht erzwungen werden
2. Eine Stimme darf nicht zum Wähler verfolgbar sein
3. Der Wähler muss das Ausmaß seiner Stimme einsehen können
4. Das System darf nicht von einer dritten Instanz beeinflussbar sein
5. Das System darf nicht von einer einzelnen Instanz kontrolliert werden
6. Nur autorisierte Wähler dürfen an der Wahl teilnehmen

Eigenschaften einer Blockchain

In diesem Abschnitt gehe ich auf die Eigenschaften einer Blockchain ein und beschreiben diese:

Proof-of-work

Jeder Block enthält Daten, die über eine sogenannte Hash-Funktion in einen einzelnen Hash-String umgewandelt werden. Eine Hash-Funktion ist eine Einwegfunktion. Das heißt, dass bei Eingabe der gleichen Daten immer der gleiche Hash-String erstellt wird. Eine weitere Eigenschaft des Blockes ist, dass er zum eigenen Hash-String sowohl den Hash-String des vorherigen Blockes speichert. So wird eine Abhängigkeit zwischen den Blöcken geschaffen, die nicht manipuliert werden kann, da bei den gleichen Daten immer der selbe Hash-String resultieren muss. Eine Manipulation der Daten würde zu einem anderen Hash-String führen.

Mining

Mining ist eine Eigenschaft bei der Abhängig von einem Schwierigkeitswert (difficulty) dem Hash-String ein Attribut hinzugefügt werden kann. Ein Beispiel für ein solches Attribut wäre eine Anzahl von führenden Nullen im Hash-String. So könnte die Difficulty von 2 für zwei führende Nullen im Hash-String stehen („00AHsjahwe212...).

Mining-Reward

Da das Minen von einem Block Rechenleistung und damit Strom kostet, wird den Teilnehmern der Blockchain, die sich dafür bereit erklären ihre Ressourcen zum Mining der Blöcke zur Verfügung zu stellen ein Mining-Reward gutgeschrieben.

Signatures

Da die Distributed Ledger Technology, auf die die Blockchain basiert eine dezentrale Verteilung der Blockchain beinhaltet, würde es gegen das Prinzip der DLT verstoßen einen zentralen Punkt der Userdatenbank zu haben. Dementsprechend gibt es keine Userdatenbank und die Teilnehmer der Blockchain weisen sich mit Private-

Public-Keypairs aus. Die Keys werden dazu genutzt, um getätigte Transaktionen in der Blockchain zu signieren, wie der Erstellung eines Blockes.

Distributed Ledger Technology

Die Distributed Ledger Technology ist eine Art der Datenspeicherung, bei der die Daten nicht zentral von einer Instanz gespeichert und verwaltet werden, sondern auf mehrere Knoten verteilt sind, die alle die gleichen Rechte haben. Eine wichtige Eigenschaft ist dabei, dass es eine Absicherung gegen Manipulation seines eigenen Datensatzes gibt. So wird nämlich immer der eigene Datensatz mit dem Datensatz der Mehrheit verglichen und überschrieben, falls die Mehrheit einen anderen Datensatz hat. Ein Problem bei der Form des Manipulationsschutzes tritt auf, wenn die Mehrheit der Teilnehmer die Daten manipuliert hat.

ARCHITEKTUR

Für die Umsetzung der Anwendung habe ich mich für die Nutzung von einem MEVN-Stack entschieden. Dabei lasse ich für die Entwicklung die MongoDB in einem Docker Container laufen und binde sie über den Container an mein Node.js Backend. Für die Anknüpfung der Datenbank verwende ich das NPM-Package Mongoose. Des Weiteren greift das Backend auf folgende NPM-Packages zu: bcrypt, body-parser, crypto-js, elliptic, express, express-validator, jsonwebtoken, moment.

Für das Frontend verwende ich Vue.js. Da ich mich bei der Umsetzung der Anwendung nicht auf das Frontend fokussieren will, verwende ich die Component-Library Vuetify. Für die Darstellung der Wahlergebnisse in Form von Charts verwende ich Chart.js.

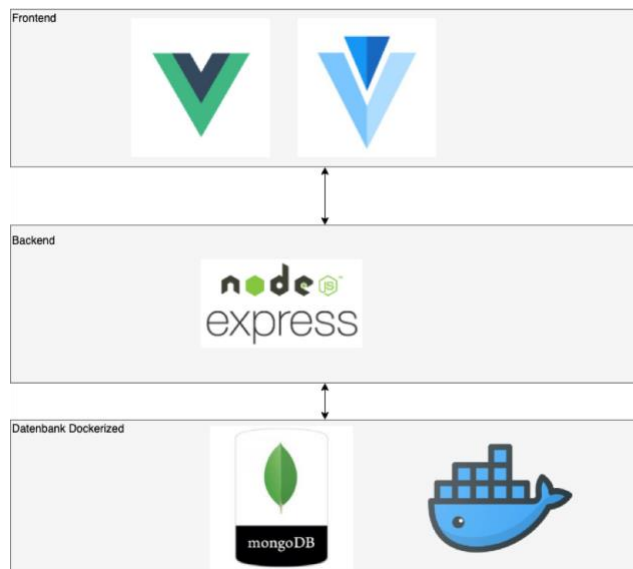


Abbildung 1: Modell der Architektur

In den Referenzen der Dokumentation sind Links zu den jeweiligen Technologien zu finden.

Technische Daten

In diesem Abschnitt beschreibe ich meine Klassen, die ich für die Umsetzung einer elektronischen Wahl benötigen.

Admin

Die Admin-Klasse besteht aus drei Attributen: (1) email, (2) password und (3) districtId. Die ersten beiden Attribute dienen dem Login. Das dritte Attribut dient als Referenz auf den Bezirk, dem ein Admin zugeordnet ist.

```
const adminSchema = new Schema({
  email: {
    type: String,
    required: true
  },
  password: {
    type: String,
    required: true
  },
  districtId: {
    type: Schema.Types.ObjectId,
    ref: 'District',
    required: true
  }
});
```

Abbildung 2: Screenshot des Admin-Schemas

Kandidat

Die Kandidat-Klasse besteht aus vier Attributen: (1) firstname, (2) lastname, (3) faction, und (4) imageUrl. Die ersten beiden Attribute sollen die Vor- und Nachnamen eines Kandidaten darstellen. Das Attribut faction soll die Partei des Kandidaten repräsentieren. Das letzte Attribut soll ein Bild des Kandidaten speichern.

```
const candidateSchema = new Schema({
  firstname: {
    type: String,
    required: true
  },
  lastname: {
    type: String,
    required: true
  },
  faction: {
    type: String,
    required: true,
    default: 'Jerry',
    enum: ['Rick', 'Morty', 'Jerry']
  },
  imageUrl: {
    type: String
  }
});
```

Abbildung 3: Screenshot vom Kandidat-Schema

Wähler

Die Wähler-Klasse besteht aus sechs Attributen: (1) username, (2) password, (3) changedPw, (4) district, (5) hasVoted, und (6) accessibleElection. Die ersten beiden Attribute dienen dem Wähler als Login. Das dritte und fünfte Attribut dient als Flag um zu prüfen, ob der User sein zugewiesenes Passwort schon einmal geändert hat bzw. ob er schon eine Stimme für die Wahl abgegeben hat. Das sechste Attribut ist eine Referenz auf die Wahl für die der Wähler eine Wahlberechtigung bekommen hat.

```
const userSchema = new Schema({
  username: {
    type: String,
    required: true
  },
  password: {
    type: String,
    required: true
  },
  changedPw: {
    type: Boolean,
    required: true,
    default: false
  },
  district: {
    type: Schema.Types.ObjectId,
    ref: 'District',
    required: true
  },
  hasVoted: {
    type: Boolean,
    required: true,
    default: false
  },
  accessibleElection: {
    type: Schema.Types.ObjectId,
    ref: 'Election',
    required: true
  }
});
```

Abbildung 4: Screenshot vom Wähler-Schema

Bezirk

Die Bezirk-Klasse besteht aus drei Attributen: (1) country, (2) city und (3) districtName. Alle drei Attribute dienen der Spezifizierung des Bezirks. Z.B. Bezirk Strombach, Stadt Gummersbach, Land Deutschland.

```
const districtSchema = new Schema({
  country: {
    type: String,
    required: true
  },
  city: {
    type: String,
    required: true
  },
  districtName: {
    type: String,
    required: true
  }
});
```

Abbildung 5: Screenshot vom Bezirk-Schema

Wahl

Eine Wahl-Klasse besteht aus sechs Attributen: (1) startDate, (2) endDate, (3) electionDescription, (4) difficulty, (5) candidates, und (6) blockchain. Die ersten zwei Attribute dienen der zeitlichen Beschränkung der Wahl. Gewählt werden darf, sobald das Startdatum erreicht ist. Sobald ein Enddatum erreicht ist, kann keine Stimme mehr für die Wahl abgegeben werden und sie ist beendet. Das dritte Attribut ist eine Bezeichnung für die Wahl. Das vierte Attribut ist ein Blockchain-spezifisches Attribut, welches in der Eigenschaft Mining erklärt ist. Das fünfte Attribut ist eine Liste an Kandidaten, die für die Wahl aufgestellt sind. Das letzte Attribut ist die Blockchain, die alle abgegebenen Stimmen der Wähler in Form eines Blockes enthält.

```
const electionSchema = new Schema({
  startDate: {
    type: String,
    required: true
  },
  endDate: {
    type: String,
    required: true
  },
  electionDescription: {
    type: String,
    required: true
  },
  difficulty: {
    type: Number,
    required: true,
    default: 2
  },
  candidates: [
    {
      type: Schema.Types.ObjectId,
      ref: 'Candidate',
      required: true
    }
  ],
  blockchain: [
    {
      type: Schema.Types.ObjectId,
      ref: 'Block',
      required: true
    }
  ]
});
```

Abbildung 6: Screenshot vom Wahl-Schema

Block

Die Block-Klasse besteht aus acht Attributen: (1) blockNumber, (2) timestamp, (3) candidate, (4) voter, (5) previousHash, (6) hash, (7) nonce, und (8) signature. Das erste Attribut ist ein Index für den Block. Das 2. Attribut ist ein Zeitstempel, der die Kreierung des Blocks festhalten soll. Das dritte Attribut enthält Daten über den Kandidaten, der eine Stimme vom Wähler bekommen hat. Das vierte Attribut enthält Daten über den Wähler, der seine Stimme abgegeben hat. Das fünfte, sechste und siebte Attribut sind klassische Blockattribute, auf die ich nicht weiter eingehen werde. Das letzte Attribut enthält Daten über die Person, die den Block „gemined“ hat.

```
const blockSchema = new Schema({
  blockNumber: {
    type: Number,
    required: true
  },
  timestamp: {
    type: String,
    required: true
  },
  candidate: {
    type: Schema.Types.ObjectId,
    ref: 'Candidate',
    required: true
  },
  voter: {
    type: Schema.Types.ObjectId,
    ref: 'User',
    required: true
  },
  previousHash: {
    type: String,
    required: true
  },
  hash: {
    type: String,
    required: true
  },
  nonce: {
    type: Number,
    required: true,
    default: 0
  },
  signature: {
    type: String,
    required: true
  }
});
```

Abbildung 7: Screenshot vom Block-Schema

Umsetzung der Eigenschaften einer Blockchain

Im folgenden Abschnitt zeige ich, wie ich die Eigenschaften einer Blockchain umgesetzt habe:

Block

calculateHash()

Die Methode *calculateHash* verwendet die Methode *SHA256* der Library *crypto-js* um einen Hash-Wert zu berechnen. Die Attribute, die ich in dem Hash speicher lauten: *blockNumber*, *timestamp*, *candidate*, *voter*, *previousHash* und *nonce*. Auf die genaue Eigenschaft der Attribute gehe ich nicht ein, da sie im Zusammenhang mit der Blockchain Technology bzw. aus der Definition der Klasse *Block* bekannt sein sollten.

mineBlock()

Die Methode *mineBlock* ist für das sogenannte „Minen“ des Blockes verantwortlich. Dabei wird der Methode eine *difficulty* als Parameter übergeben, die die Anzahl der führenden Nullen bestimmen soll. Bis die Anzahl der Nullen erreicht wird, wird der Hash neu berechnet und pro Durchlauf der *Nonce*-Wert um 1 erhöht.

signBlock()

Die Methode *signBlock* dient dazu den Block mit dem Public Key des Erstellers des Blockes zu signieren. Dies dient dazu genau identifizieren zu können, wer den Block gemined hat.

isValid()

Die Methode *isValid* überprüft den Block auf eine Signatur und gibt einen boolean Wert aus, ob der Block valide ist.

```
blockSchema.methods.calculateHash = function() {
  const hash = SHA256(
    this.blockNumber +
    this.timestamp +
    this.candidate.toString() +
    this.voter.toString() +
    this.previousHash +
    this.nonce
  ).toString();
  return hash;
};

blockSchema.methods.mineBlock = async function(difficulty) {
  while (
    this.hash.substring(0, difficulty) !== Array(difficulty + 1).join('0')
  ) {
    this.nonce++;
    this.hash = this.calculateHash();
  }
  this.signBlock();
  await this.save();
  console.log('Block mined: ${JSON.stringify(this)}');
  return;
};

blockSchema.methods.signBlock = function() {
  this.signature = key.sign(this.hash, 'base64').toDER('hex');
};

blockSchema.methods.isValid = function() {
  if (!this.signature || this.signature.length === 0) {
    throw new Error('No signature!');
  }
};
```

Abbildung 8: Screenshot der Methoden des Blockes

Blockchain

createGenesis()

Die Methode *createGenesis* wird aufgerufen, wenn eine neue Wahl erstellt wird, um diese direkt mit einem Genesis Block zu initialisieren. Anzumerken ist, dass die Abhängigkeiten zu Kandidat und Wähler erfordern, dass ich einen Genesis Wähler und Genesis Kandidaten in der Datenbank hinterlegt haben.

latestBlock()

Die Methode *latestBlock* gibt uns den letzten Block der Blockchain zurück.

addBlock()

Die Methode *addBlock* fügt einen neuen Block der Blockchain hinzu. Dafür wird ein neuer Block erstellt und mit den entsprechenden Daten versehen. Im Anschluss wird dieser Block der Blockchain hinzugefügt.

deleteAllBlocks()

Die Methode *deleteAllBlocks* ist eine Hilfsmethode, die bei der Löschung einer Wahl alle abhängigen Blöcke aus der Datenbank löschen soll.

```

electionSchema.methods.createGenesis = async () => {
  const genesis = new Block({
    blockNumber: 0,
    timestamp: moment(),
    candidate: ObjectId('5cef9dea531481caca068d0'),
    voter: ObjectId('5cef9ede531481caca068d1'),
    previousHash: '0',
    hash: '0',
    nonce: 0,
    signature: '0'
  });
  try {
    const gb = await genesis.save();
    return gb;
  } catch (err) {
    if (!err.statusCode) {
      err.statusCode = 500;
    }
    return err;
  }
};

electionSchema.methods.latestBlock = function() {
  return this.blockchain[this.blockchain.length - 1];
};

electionSchema.methods.addBlock = async function(newBlock) {
  newBlock.previousHash = this.latestBlock().hash;
  newBlock.hash = newBlock.calculateHash();
  await newBlock.mineBlock(this.difficulty);
  await newBlock.save();
  this.blockchain.push(newBlock);
  await this.save();
  return;
};

electionSchema.methods.deleteAllBlocks = async function() {
  for (i = 0; i < this.blockchain.length; i++) {
    await Block.findByIdAndDelete(this.blockchain[i]._id);
  }
  return;
};

```

Abbildung 9: Screenshot der Methoden der Blockchain

checkValid()

Die Methode checkValid überprüft, ob die Blockchain valide ist, oder ob diese an irgendeiner Stelle manipuliert wurde.

getElectionResult()

Die Methode getElectionResult ist eine Hilfsmethode, die die Blöcke der Blockchain durchgeht und für jeden Kandidaten die Anzahl der erhaltenen Stimmen berechnet. Im Anschluss gibt die Funktion eine Liste der Kandidaten und ihrer erhaltenen Stimmen zurück.

```

electionSchema.methods.checkValid = async function() {
  for (let i = 1; i < this.blockchain.length; i++) {
    const currentBlock = this.blockchain[i];
    const previousBlock = this.blockchain[i - 1];

    if (
      currentBlock.hash !== currentBlock.calculateHash() ||
      currentBlock.previousHash !== previousBlock.hash
    ) {
      return false;
    }
  }
  return true;
};

electionSchema.methods.getElectionResult = async function() {
  let result = [];
  this.candidates.forEach(candidate => {
    result.push({ candidate: candidate, vote_count: 0 });
  });
  for (let i = 1; i < this.blockchain.length; i++) {
    const index = result.findIndex(candidate => {
      return (
        candidate.candidate_id.toString() ===
        this.blockchain[i].candidate.toString()
      );
    });
    result[index].vote_count = result[index].vote_count + 1;
  }
  console.log(`Is Blockchain valid: ${await this.checkValid()}`);
  return result;
};

```

Abbildung 10: Screenshot der Methoden der Blockchain

ABLAUF

In diesem Abschnitt beschreibe ich einen typischen Ablauf, wie die Anwendung für eine Wahl genutzt werden kann. Dabei werde ich den Ablauf in sechs Schritte aufteilen:

1. Der potentielle Wähler sucht das Rathaus in seinem Bezirk auf und beantragt einen Zugang zu Wahl XY bei einem Beamten des Rathauses.
2. Der Beamte öffnet die Anwendung, wählt die Wahl aus und generiert dem Wähler einen Zugang, den sich der Wähler notieren muss.
3. Der Wähler verlässt das Rathaus und geht nach Hause, wo er sich bei der Anwendung mit seinen erhaltenen Zugangsdaten anmeldet.
4. Nach der Anmeldung wird der Wähler dazu aufgefordert sein Passwort zu ändern.
5. Nachdem der Wähler sein Passwort geändert hat, kann dieser seinen favorisierten Kandidaten für die Wahl auswählen und seine Stimme abgeben.
6. Nachdem der Wähler seine Stimme abgegeben hat, kann er den aktuellen Stand der Wahl nachverfolgen, bis das Enddatum der Wahl erreicht ist. Sobald das Enddatum der Wahl erreicht ist, kann der Zugang nicht mehr genutzt werden.

Möchte der Wähler an einer anderen Wahl teilnehmen, muss er sich einen neuen Zugang im Rathaus seines Bezirks beantragen.

UMSETZUNG DER ANFORDERUNGEN AN EIN WAHLSYSTEM

Eine Stimme darf nicht erzwungen werden

Diese Anforderung an ein Wahlsystem kann auch in der heutigen Zeit bei den heutigen Wahlen nicht vollständig verhindert werden. Wähler können immer noch bedroht werden, um ein bestimmtes Ergebnis bei der Wahl zu erzielen.

Eine Stimme darf nicht zum Wähler verfolgbar sein

Dadurch, dass für jeden Wähler ein Zugang zur Wahl generiert wird, kann der Identifier vom Wähler bei der Wahl nicht auf diesen zurückverfolgt werden.

Der Wähler muss das Ausmaß seiner Stimme einsehen können

Nachdem der Wähler seine Stimme für seinen Kandidaten abgegeben hat, kann er das Ausmaß seiner Stimme in den Diagrammen, die den aktuellen Wahlstand zeigen, einsehen.

Das System darf nicht von einer dritten Instanz beeinflussbar sein

Dadurch, dass jeder Wähler einen Zugang zur Wahl beantragen muss, kann eine dritte Instanz das System nicht beeinflussen, da es nur von dem Beamten eines Bezirks die Zugänge gibt.

Das System darf nicht von einer einzelnen Instanz kontrolliert werden

Dadurch, dass die Blockchain auf alle Bezirke des Staates verteilt ist, kann sie nicht von einem Bezirk kontrolliert werden. Zusätzlich sorgt die Eigenschaft der Mehrheitskontrolle der Blockchain dafür, dass das System nicht von der Minderheit manipuliert werden kann.

Nur autorisierte Wähler dürfen an der Wahl teilnehmen

Dadurch, dass nur Wähler, die sich einen Zugang beim Rathaus des Bezirks beantragen, wählen können, ist diese Anforderung gewährleistet.

OPTIMIERUNGSPOTENZIAL

Die Anwendung kann an mehreren Stellen optimiert werden, um mehr User Experience zu bieten und die Eigenschaften einer Blockchain noch weiter zu erfüllen. Ein Beispiel für eine solche Optimierung ist, dass jeder User sich über seine Personalausweisnummer bei der App registrieren kann und sich selbst für eine Wahl einen Zugang erstellen kann, sodass der Faktor Beantragung bei dem Bezirks Rathaus wegfällt. Des Weiteren kann ein Mining Reward hinzugefügt werden, womit die Blockchain nicht mehr bei den Bezirken des Staates verwaltet wird, sondern vollständig dezentralisiert.

REFERENZEN

"Axel Voss". 2019. De.Wikipedia.Org. https://de.wikipedia.org/wiki/Axel_Voss.

Azimdoust, Negin. 2019. "Distributed Ledger Technologie (DLT) Ist Mehr Als Blockchain". Blockchainwelt.

<https://blockchainwelt.de/dlt-distributed-ledger-technologie-ist-mehr-als-blockchain/>.

"Chart.js | Open Source HTML5 Charts For Your Website". 2019. Chartjs.Org. Accessed October 13. <https://www.chartjs.org/>.

"Express-Middleware Verwenden". 2019. Expressjs.Com. Accessed October 13. <https://expressjs.com/de/guide/using-middleware.html>.

Foundation, Node.js. 2019. "Guides | Node.js". Node.Js. Accessed October 13. <https://nodejs.org/en/docs/guides/>.

Ganji, Raghavendra, and Yatish BN. 2018. "ELECTRONIC VOTING SYSTEM USING BLOCKCHAIN". Pdfs.Semanticscholar.Org. <https://pdfs.semanticscholar.org/84c7/c5b9df300d5d282038684654e2d47998b3dd.pdf>.

"Get Started, Part 1: Orientation And Setup". 2019. Docker Documentation. Accessed October 13. <https://docs.docker.com/get-started/>.

Hebel, Christina. 2018. "Putin Gewinnt Russland-Wahl: Ein Sieg Wie Gemacht - SPIEGEL ONLINE - Politik". SPIEGEL ONLINE. <https://www.spiegel.de/politik/ausland/russland-wahl-wie-sich-wladimir-putin-seinen-sieg-gemacht-hat-a-1198704.html>.

Hjálmarsson, Friðrik Þ., and Gunnlaugur K. Hreiðarsson. 2019. "Blockchain-Based E-Voting System". Skemman.Is. <https://skemman.is/bitstream/1946/31161/1/Research-Paper-BBEVS.pdf>.

"Introduction — Vue.js". 2019. Vuejs.Org. Accessed October 13. <https://vuejs.org/v2/guide/>.

"Quick Start — Vuetify.js". 2019. V15.Vuetifyjs.Com. Accessed October 13. <https://v15.vuetifyjs.com/en/getting-started/quick-start>.

Stresing, Laura. 2019. "Weiß Die CDU, Was Sie Da Tut?". Ww.T-Online.De. https://www.t-online.de/digital/id_85443096/posse-um-artikel-13-wissen-axel-voss-und-die-cdu-da-was-sie-tun-.html.

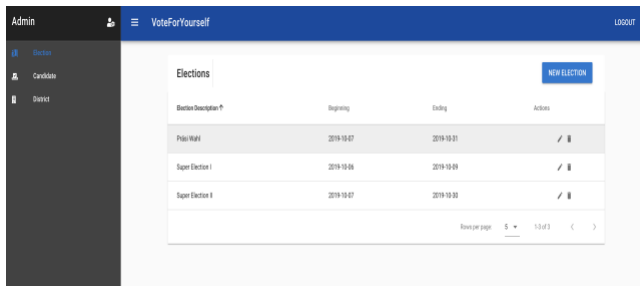
U/LongLiveAnarchyAcres. 2019. "Axel Voss On Artikel 13.". Reddit. https://www.reddit.com/r/memes/comments/aqd4py/axel_voss_on_artikel_13/.

ANHANG



A login form with a blue header bar containing the text "Login form" and a code icon. Below the header, there are two input fields: "Login" with a user icon and "Password" with a lock icon. A blue "LOGIN" button is positioned at the bottom right of the form.

Figure 1: Login-Screen der GUI

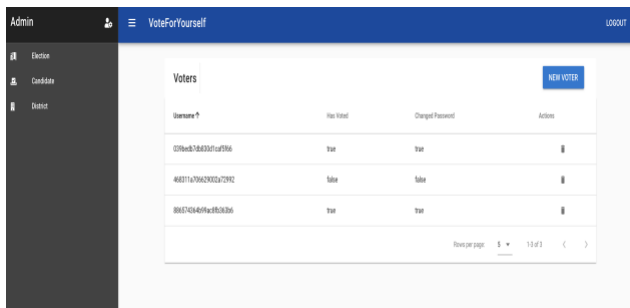


Admin View - Election Overview. The interface shows a sidebar with "Admin", "Candidates", and "District". The main content area displays a table of elections.

Election Description	Beginning	Ending	Actions
Präsi Wahl	2019-10-07	2019-10-31	/ i
Super Election 1	2019-10-06	2019-10-09	/ i
Super Election 1	2019-10-07	2019-10-30	/ i

Rows per page: 5 | 10 of 3

Figure 2: Admin-View - Election Overview der GUI

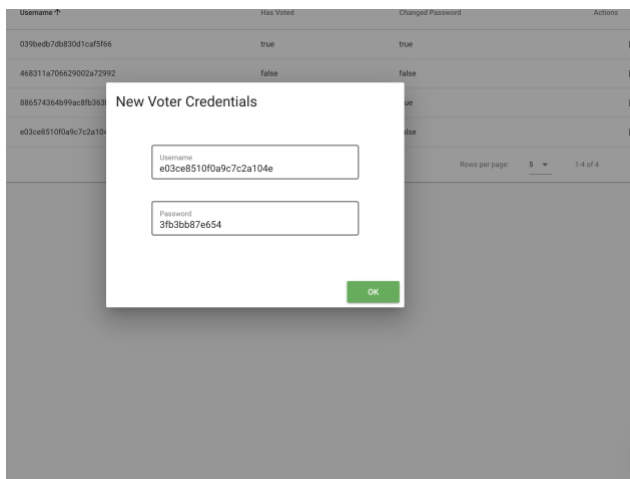


Admin View - Election Detail. The interface shows a sidebar with "Admin", "Candidates", and "District". The main content area displays a table of voters.

Username	Has Voted	Changed Password	Actions
039ebd7db830d1caf5f6	true	true	/
468311a706629002a72992	false	false	/
886574364099ac8fb363	true	true	/

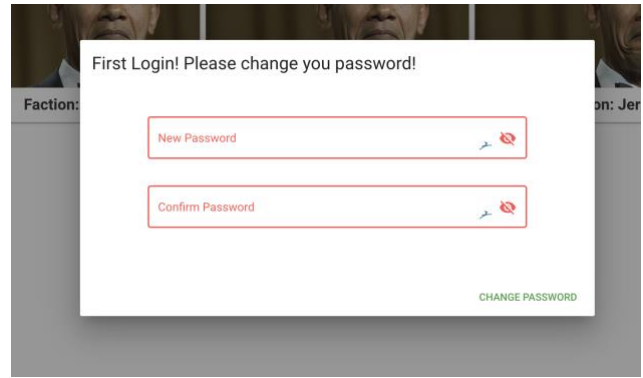
Rows per page: 5 | 10 of 3

Figure 3: Admin View - Election Detail der GUI



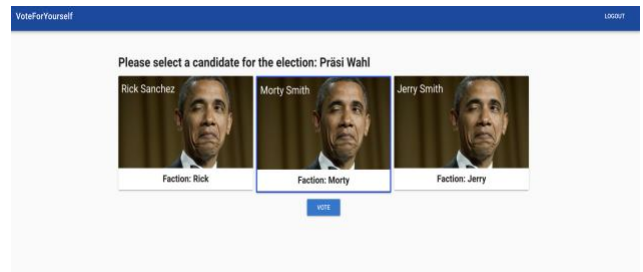
New Voter Credentials Dialog. A modal dialog box with two input fields: "Username" and "Password". The username field contains the text "e03ce8510f0a9c7c2a104e" and the password field contains "3fb3bb87e654". An "OK" button is at the bottom right.

Figure 4: Admin View - New Voter Credentials Dialog



First Login! Please change your password! A modal dialog box with two input fields: "New Password" and "Confirm Password". Both fields have a red border and a red "X" icon. A "CHANGE PASSWORD" button is at the bottom right.

Figure 5: Voter View - Change Password nach Erstlogin Dialog



Voter View - Auswahl des Kandidaten. The interface shows a sidebar with "VoteForYourself" and "Logout". The main content area displays a selection screen for the "Präsi Wahl" election. Three candidates are shown: Rick Sanchez, Morty Smith, and Jerry Smith. Morty Smith is selected, indicated by a blue border. A "Vote" button is at the bottom.

Figure 6: Voter View - Auswahl des Kandidaten

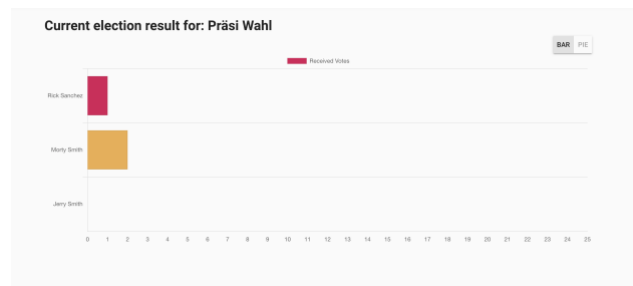


Figure 7: Voter View - Übersicht des aktuellen Wahlverlaufs als Balkendiagramm

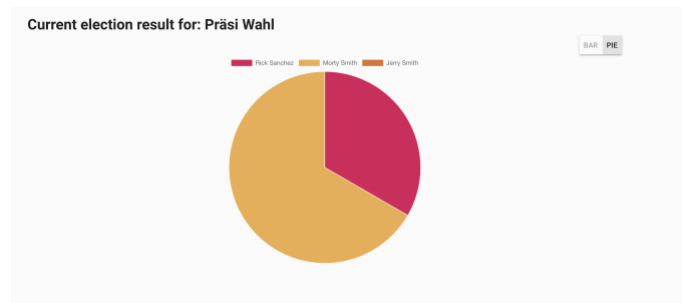


Figure 8: Voter View - Übersicht des aktuellen Wahlverlaufs als Tortendiagramm