Inhaltsverzeichnis: LIESMICH-Datei

Inhaltsverzeichnis: LIESMICH-Datei 1
Einführung 2
Dokumentation 2
Datenbank-Dump & Spiel-Assets 3
Spiel-Teaser 3
Systemanforderungen 3
    Lokale Datenbank mit Docker einrichten (optional) 5
    Empfohlener Editor und Extensions 6
Einsatz auf einem Server (mit CapRover) 6
Betreiben des Servers von einem Laptop im lokalen Netzwerk 10
    Über den Betrieb eines lokalen Servers 10
    Erste Einrichtung des Routers 10
    Server-Installation 11
    Server Starten 13
    Auf dem Windows-Client (Desktop/Tablet) 13
    Auf Nicht-Windows-Clients 13
NPM-Befehle 14
Umgebungsvariablen 14
Wer war beteiligt? 19

Einführung

Adamara: Harsh Waters ist ein kollaborativer Echtzeit-Spiele-Editor und ein isometrisches RPG/Abenteuerspiel. In Kombination werden sie in Workshops zur politischen Bildung mit Jugendlichen eingesetzt:

1. Die Teilnehmenden spielen das Spiel.
2. Anschließend diskutieren sie ihre Erfahrungen im Spiel mit der Workshop-Gruppe.
3. In den nächsten Tagen des Workshops organisieren sie sich in kleinen Teams und erstellen mit dem Editor neue Erfahrungen im Spiel (entweder als Zusatz zum Spiel oder als eigenständiges Spiel), die dann von anderen gespielt werden können.

Weitere Informationen zum Projekt finden Sie hier: https://www.cultures-interactive.de/en/the-project.html. (Hinweis: "Call of Prev" war der Arbeitstitel des Projekts und wird möglicherweise noch auf der Website erwähnt).

Dokumentation

Diese LIESMICH-Datei gibt Entwicklern einen Überblick darüber, wie sie die Anwendung für die Entwicklung oder den Einsatz einrichten können.

Zusätzliche Dokumentation in englischer Sprache:

- A quick overview of the different parts of the editor
- Understanding the code
  - Folder structure
  - Technologies used
  - Real-time collaboration
  - Undo/redo

    - User roles
    - Optimizations on the client
    - Optimizations on the server
    - Isometric depth ordering
- More information about the project

Zusätzliche Dokumentation auf Deutsch:

- Workshop-Ablauf und Editor-Benutzung (PDF)
- Mehr Informationen zum Projekt

Datenbank-Dump & Spiel-Assets

Ein Dump der Datenbank (der das komplette Spiel und die meisten Spielinhalte enthält) ist hier verfügbar:

- https://drive.google.com/file/d/1Ugfe04A2NthHCSJajXdDPXv1MPDJBqZd/view (Lizenz: CC BY 4.0)

Die Spiel-Assets (Artwork, Animationen, Sounds etc.) sind separat erhältlich unter:

- https://drive.google.com/drive/folders/1a-X0hVuhpUyYIpwt0f7bFZegkuWGLIWU (Lizenz: CC BY 4.0)

Spiel-Teaser

Der Teaser zum Spiel kann hier angesehen werden:

- https://www.youtube.com/watch?v=EdVFUJPmzWE

Systemanforderungen

Server/Entwicklung:

- RAM: 16 GB empfohlen (der Server benötigt 4 - 6 GB RAM oder mehr, je nach Anzahl der Assets)

- Festplattenspeicher: 20 GB freier Festplattenspeicher (4 GB für den Code und die statischen Assets, 1 GB für die Datenbank; mehr für Docker, CapRover und ähnliche Tools, Caching und den Deployment-Prozess)

Clients (zum Öffnen des Editors/der Webanwendung des Spiels):

- Browser: Google Chrome (andere Browser funktionieren möglicherweise, sind aber nicht getestet)
- RAM: 8 GB (das Spiel/der Editor benötigt 2-3 GB)
- Am besten getestet auf unserem Zielgerät: Microsoft Surface Pro 6 (2018)
- Zum Zeitpunkt der Erstellung dieser LIESMICH-Datei hat iOS einen Fehler, der den Editor daran hindern könnte, sich mit dem Server zu verbinden.

Entwicklungs-Setup

1. Installieren Sie die in package.json angegebene Node-Version (derzeit 14.17.1), z.B. über einen Versionsmanager wie NVM Windows / NVM (for Linux/macOS) (empfohlen) oder direkt von https://nodejs.org/dist/v14.17.1.
2. Laden Sie dieses Projekt runter.
3. Navigieren Sie in Ihrem bevorzugten Terminal (z.B. in Ihrer IDE) zum Projektordner und führen Sie `npm install` aus.
4. Installieren Sie eine MariaDB oder MySQL Datenbank, z.B. über mariadb.org oder mit Hilfe der Anweisungen unter Setup Local Database via Docker (optional).
5. Erstellen Sie mit einem SQL-Administrationstool (z.B. HeidiSQL oder phpmyadmin) eine neue Datenbank `adamara` mit dem Benutzer `adamara`.
6. (Optional) Laden Sie einen Datenbank-Dump herunter und kopieren Sie ihn in Ihre lokale Datenbank, um einen guten Ausgangspunkt zu haben.
    - Dies geschieht am besten über die Kommandozeile: `mysql -u root -p adamara < dump.sql`

- Wenn Sie eine andere Methode verwenden, stellen Sie sicher, dass alle Sonderzeichen (z.B. im Feld `name` in der Tabelle `tile_asset`) korrekt importiert wurden. (Ich habe keine Möglichkeit gefunden, die Kodierung während des Imports in HeidiSQL korrekt einzustellen, daher bevorzuge ich den Weg über die Kommandozeile).

7. Kopieren Sie `.env.template` nach `.env`.
8. Passen Sie den Platzhalter `DB_URL` in `.env` an Ihre lokale Datenbank an.
9. Führen Sie `npm run dev` aus.
10. Öffnen Sie http://localhost:3000.

Lokale Datenbank mit Docker einrichten (optional)

1. Docker installieren
2. Speichern Sie einen Datenbank-Dump vom Server in:

   `./dumps/<some name>.sql`

   (Hinweis: alle SQL-Dateien in diesem Verzeichnis werden importiert)
3. Legen Sie Umgebungsvariablen für das Betriebssystem an:

   `ADAMARA_DB_ROOT_PASSWORD`
   `ADAMARA_DB_USER`
   `ADAMARA_DB_USER_PASSWORD`
   `ADAMARA_DB_NAME`
4. Starten:

   `docker-compose up` zum Starten der Datenbank
   `docker-compose stop` zum Stoppen der Datenbank
5. Stellen Sie sicher, dass `DB_URL` in `.env` dem Benutzer/Passwort/Namen entspricht, den Sie in Schritt 3 festgelegt haben.

Empfohlener Editor und Extensions

Es wird empfohlen (ist aber nicht zwingend erforderlich), mit Visual Studio Code zu arbeiten, einem funktionsreichen, aber schlanken Editor, der für alle gängigen Systeme kostenlos verfügbar ist. Der Hauptvorteil besteht darin, dass die Formatierung des Codes mit minimalem Aufwand an das Projekt angepasst werden kann. Zusätzlich erleichtern die unten aufgeführten Erweiterungen die Arbeit.

VSCode Extensions (können direkt in VSCode über das Fenster "Extensions" installiert werden):

- Zwingend notwendig:
  - ESLint für korrekte Linting Integration.
  - i18n Ally für die fantastische Unterstützung unserer Übersetzungsdateien.
  - vscode-styled-components (Autor: "Styled Components") für Syntax-Highlighting für styled-components CSS-in-JavaScript.
- Situational (sobald Sie mit zugehörigen Dateien arbeiten):
  - Jest Test Explorer um eine eingebaute Unit-Test-UI zu erhalten.
  - WebGL GLSL Editor, wenn Sie Shader im Projekt ändern.
- Weitere nette optionale Dinge:
  - Color Picker zur Farbauswahl im Editor.
  - NPM Audit, um einen visuellen Bericht über den npm audit Bericht zu erhalten (einschließlich empfohlener Maßnahmen).

Einsatz auf einem Server (mit CapRover)

1. Installieren und konfigurieren Sie CapRover (bis einschließlich Schritt 3) auf einem Server mit mindestens 16 GB RAM.
   - Am einfachsten ist dies wahrscheinlich mit der im Dokument erwähnten DigitalOcean One-Click App.

- Obwohl 16 GB RAM empfohlen werden, kann es auch mit 8 GB RAM funktionieren, wenn Sie eine Auslagerungsdatei erstellen. Wenn Sie Probleme bei der Erstellung haben, sollten Sie auf 16 GB aufrüsten und es erneut versuchen.

2. Öffnen Sie das CapRover Frontend (https://captain.your-url.com).
3. Erstellen Sie eine MariaDB-Instanz:
   A. "Apps" -> "Create new App-> "One-Click Apps/Databases“: “MariaDB”
      - App Name: `adamara-mariadb`
      - Passwort: Geben Sie etwas ein
      - Kann als Voreinstellung belassen werden:
        - MariaDB Version: 10.4
        - Standard-Server-Zeichensatz: utf8mb4
        - Standard-Serversortierung: utf8mb4_unicode_ci
   B. Wechseln Sie zu Ihrer Datenbank unter "Apps" -> "adamara-mariadb-db" und öffnen Sie den Reiter “Deployment”.
   C. Scrollen Sie nach unten zu "Method 4: Deploy plain Dockerfile" und fügen Sie folgendes ein:
      - `FROM mariadb:10.4`
      - `CMD ["--max-allowed-packet=100M", "--character-set-server=utf8mb4", "--collation-server=utf8mb4_unicode_ci", “--skip-character-set-client-handshake"]`
   D. Klicken Sie auf "Deploy Now”.
4. Erstellen Sie die Datenbank:
   E. Fügen Sie "phpmyadmin" als CapRover-Anwendung hinzu.
   F. Melden Sie sich bei phpmyadmin an und erstellen Sie eine neue Datenbank "adamara" mit dem Benutzer “adamara”.
   G. Optional, wenn Sie unsere Datenbank verwenden wollen:

- Loggen Sie sich mit einem SSH-Client, z.B. PuTTY, auf dem Server ein.
- Laden Sie den Datenbank-Dump herunter und legen Sie ihn als `dump.sql` in `/var/lib/docker/volumes/captain--app-mariadb-db-data/_data` ab.
- `docker ps`
- Überprüfen Sie den Namen der aktuellen Datenbank-App, z.B. `srv-captain—app-mariadb-db.1.ystl2yu4ygiiq1qsn8bjs6fgr`
- `docker exec -it srv-captain--app-mariadb-db.1.ystl2yu4ygiiq1qsn8bjs6fgr /bin/bash`
- `mysql -u root -p adamara < /var/lib/mysql/dump.sql`

5. Erstellen Sie den adamara-Server:

H. "Apps" -> "Create new App”

- Name: `adamara`
- Hat persistente Daten: Ja

I. Klicken Sie "Create New App”.

J. Wechseln Sie zu Ihrer App unter "Apps" -> “adamara".

K. Unter "HTTP Settings”:

- Klicken Sie "Enable HTTPS”
- Aktivieren Sie "Websocket Support”
- Klicken Sie "Save & Update"

L. Unter "App Configs”:

- Umgebungsvariablen: Bulk Edit, fügen Sie den Inhalt von .env.deploy.template ein und tragen Sie die Werte ein.
  - DB_URL: `mysql://adamara:YOUR_ADAMARA_USER_PASSWORD@srv-captain--app-mariadb-db:3306/adamara`

- SESSION_SECRET: Eine zufällige UUID
- SERVICE_API_KEY: Eine weitere zufällige UUID
- ACCESS_CODE: Das Admin-Passwort, mit dem Sie sich bei der Adamara-App anmelden möchten
- SERVER_BASE_URL: `https://app.YOUR-URL-GOES-HERE.com`

- Dauerhafte Verzeichnisse:
  - Pfad in der App: `/logs`, Bezeichnung: `app-logs`
  - Pfad in der App: `/temp`, Bezeichnung: `app-temp`
- Klicken Sie "Save & Update"

M. Unter “Deployment":

- Konfigurieren Sie "Method 3: Deploy from Github/Bitbucket/Gitlab”, um auf dieses (oder Ihr) Repository zu verweisen, und stellen Sie es bereit.
- Prüfen Sie "View Build Logs" um zu sehen, ob der Build korrekt abgeschlossen wurde.
- Prüfen Sie "View App Logs" um zu sehen, ob die Anwendung korrekt startet. Der Startvorgang ist abgeschlossen, wenn Sie sehen: `[info] App listening on port 80!`

Wenn alles richtig gelaufen ist, sollten Sie nun einen laufenden Server unter https://app.YOUR-URL-GOES-HERE.COM zur Verfügung haben.

Es gibt ein paar Tools, die ebenfalls empfohlen werden, aber nicht zwingend notwendig sind:

- **NetData**: Eingebaute Überwachung, verfügbar unter "Monitoring" im CapRover-Hauptmenü.
- **mysql-backup**: Automatische Backups
  1. Erstellen Sie eine App mit persistenten Daten.
  2. Einsatz über Image-Name: `databack/mysql-backup:latest`
  3. Erstellen Sie ein persistentes Verzeichnis `/backup` mit der Bezeichnung `app-mysql-backup`
  4. Setzen Sie Umgebungsvariablen:

    - DB_SERVER: srv-captain—app-mariadb-db
    - DB_USER: `root`
    - DB_PASS: Ihr Datenbank-Root-Passwort
    - DB_DUMP_TARGET: `/backup`
5. Klicken Sie "Save & Update"

- CapRover-Apps, die Sie hinzufügen können:
    1. **Dozzle**: Zeigt die Log-Ausgabe für alle Docker-Container.
    2. **Portainer**: Verwaltet Docker-Container.
    3. **phpmyadmin**: Prüft den Inhalt Ihrer Datenbank.
- **Sentry.io**: Fehlerüberwachung (über Umgebungsvariablen in der App konfigurieren)

Betreiben des Servers von einem Laptop im lokalen Netzwerk

Über den Betrieb eines lokalen Servers

Um mit Adamara zu arbeiten, ist eine stabile und schnelle Internetverbindung erforderlich. Leider ist dies in Ihrer Umgebung möglicherweise nicht der Fall. Wenn Sie den Server über einen Laptop in einem von Ihnen kontrollierten lokalen Netzwerk betreiben, können Sie einen Workshop auch in Situationen durchführen, in denen das Internet möglicherweise langsam ist oder in denen nicht klar ist, ob die drahtlose Verbindung stark genug ist.

Leider können die Daten nicht zwischen Webserver und lokalem Server zusammengeführt werden.

Erforderliche Ausrüstung:

- Ein Laptop mit mindestens 16 GB RAM wird empfohlen. (Für das Starten eines Servers werden je nach Umfang der Assets 4 - 6 GB RAM oder mehr benötigt).
- Ein Router.

Erste Einrichtung des Routers

1. Verbinden Sie den Server-Laptop mit einem Router.

2. (Optional) Stellen Sie sicher, dass dem Laptop vom Router immer die gleiche IPv4 zugewiesen wird, damit Sie auf den Clients Lesezeichen setzen können.

Server-Installation

1. Installieren Sie die in package.json angegebene Node-Version (unter `engines` -> `node`; derzeit 14.17.1), zum Beispiel über einen Versionsmanager wie NVM Windows / NVM (for Linux/macOS) (empfohlen), oder direkt von https://nodejs.org/dist/v14.17.1.
2. Laden Sie dieses Projekt herunter.
3. Navigieren Sie in Ihrem bevorzugten Terminal zu dem Projektordner und führen Sie `npm install` aus.
4. Installieren Sie eine MariaDB- oder MySQL-Datenbank, z. B. über mariadb.org oder mit Hilfe der unter Setup Local Database via Docker (optional) beschriebenen Anweisungen.
5. Erstellen Sie mit einem SQL-Verwaltungstool (wie HeidiSQL oder phpmyadmin)eine neue Datenbank `adamara` mit dem Benutzer `adamara`.
6. Laden Sie unseren Datenbank-Dump oder Ihren eigenen von Ihrem Hauptserver herunter und laden Sie ihn in die Datenbank.
   - Dies geschieht am besten über die Kommandozeile: `mysql -u root -p adamara < dump.sql`
   - Wenn Sie eine andere Methode verwenden, stellen Sie sicher, dass alle Sonderzeichen (z.B. im Feld `name` in der Tabelle `tile_asset`) korrekt importiert wurden. (Ich habe keine Möglichkeit gefunden, die Kodierung während des Imports in HeidiSQL korrekt einzustellen, daher bevorzuge ich den Weg über die Kommandozeile).
7. Kopieren Sie `.env.local-server.template` nach `.env`.
8. `.env`:
   - `DB_URL`: Ersetzen Sie den Platzhalter durch Ihre Datenbankzugangsdaten.
   - `SESSION_SECRET`: Füllen Sie den Platzhalter mit einer zufälligen Zeichenfolge (mindestens 30 Zeichen, z. B. eine UUID)

- `SERVICE_API_KEY`: Füllen Sie den Platzhalter mit einer zufälligen Zeichenfolge (mindestens 30 Zeichen, z. B. eine UUID)
- `ACCESS_CODE`: Geben Sie hier das Passwort ein, das die Benutzer eingeben müssen, um die Seite zu öffnen. Bitte verwenden Sie **nicht** das gleiche Passwort wie das des Hauptservers.

9. Generierung von SSL-Stammzertifikaten:

   - Auf Windows

   1. Führen Sie tools\root-ca-generation-windows\generate-and-install-root-CA.bataus. Lassen Sie das Fenster für den nächsten Schritt geöffnet.
   2. Kopieren Sie `SSL_GENERATE_ROOT_CERT` und `SSL_GENERATE_ROOT_KEY` nach `.env` (überschreiben Sie die Zeilen #8 und #9).

   - Auf anderen Systemen

   3. Installieren Sie mkcert.
   4. Find your root certificate path via `mkcert -CAROOT`.
   5. In the `.env` file:
      - Point `SSL_GENERATE_ROOT_CERT` to your `rootCA.pem` in the root certificate path.
      - Point `SSL_GENERATE_ROOT_KEY` to your `rootCA-key.pem` in the root certificate path.
         1. Create a folder Root CA Installation Package.
         2. Copy `mkcert.exe` into that folder.
         3. Copy `install-root-CA.bat_` into that folder an rename it to `install-root-CA.bat`.
         4. Copy `rootCA.pem` into that folder.

10. Kopieren Sie den Ordner `tools\root-ca-generation-windows\Root CA Installation Package` auf einen USB-Stick.

Server Starten

1. Starten Sie den Server, indem Sie `npm-run-dev.bat` ausführen.

Auf dem Windows-Client (Desktop/Tablet)

1. Verbinden Sie den Client mit dem Router.
2. Installieren Sie Google Chrome.
3. Schließen Sie den USB-Stick mit dem Root-CA-Installationspaket an.
4. Öffnen Sie den USB-Stick und führen Sie `install-root-CA.bat` aus.
5. Trennen Sie den USB-Stick ab.
6. Starten Sie Google Chrome.
7. Öffnen Sie die URL: `https://[server-ip-configured-in-the-router]:3000 (e.g. https://192.168.178.55:3000).`

Wenn Sie sich nicht an die von Ihnen konfigurierte IP-Adresse erinnern, suchen Sie in der Serverkonsole nach der Zeile `If you want to access this server from your local network, one of the following URLs should work in Google Chrome:.`

Wenn Sie einen lokalen Server verwenden, denken Sie daran, diese URL auch beim Drucken von Workshop-PDFs zu verwenden.

Auf Nicht-Windows-Clients

Sie können die obigen Anweisungen befolgen, aber die Schritte 3.-5. werden durch die hier beschriebenen Schritte ersetzt.

NPM-Befehle

- `npm run dev` - Client und Server befinden sich im Überwachungsmodus mit Source Maps. Öffnen Sie http://localhost:3000, sobald es läuft, um die Anwendung zu sehen.
- `npm run test` - Führt Jest-Tests aus
- `npm run lint` - Führt es-lint aus
- `npm run lint:fix` - Führt es-lint --fix aus
- `npm run build` - Der Ordner dist enthält alle benötigten Dateien, sowohl für den Client (Bundle) als auch für den Server.
- `npm start` - Startet einfach `node ./dist/server/server.js`
- `npm start:prod` - setzt NODE_ENV auf production und führt dann `node ./dist/server/server.js` aus. (Umgehung des Webpack-Proxys)
- `npm run migrate-up` - Führt alle Datenbankmigrationen aus, die noch nicht ausgeführt wurden. (Wird auch automatisch ausgeführt, wenn der Server z.B. über `npm run dev` oder `npm start` gestartet wird).
- `npm run migrate-down` - Ruft die „down"-Methode für die zuletzt ausgeführte Migration auf und macht sie rückgängig. Kann mehrere Male aufgerufen werden, um mehrere Migrationen rückgängig zu machen.

Umgebungsvariablen

- `SERVER_BASE_URL`: Wird im Docker-Build verwendet, um die aktuell laufende Anwendung auf dem Server vor dem Build herunterzufahren (`SERVER_BASE_URL/api/service/shutdown` wird über curl aufgerufen). Beispielwert: `https://adamara-server.com`
- `SESSION_SECRET`: Das Sitzungsgeheimnis. „Sollte nicht leicht von einem Menschen geparst werden können und am besten eine zufällige Zeichenfolge sein". Beispiel (das nicht auf einem

Server verwendet werden sollte, da es jetzt öffentlich ist): `87e13481-9b3f-4e16-be2f-186858feabf2`

- `BASE_PATH`: Der URL-Pfad vom Server-Stammverzeichnis für den React-Build. Beispielwert: `/`
- `DB_URL`: Der URL-String der Datenbank. Beispielwert/Format: `mysql://database_username:user_password@localhost:3306/database_name`
- `SERVICE_API_KEY`: Eine sichere Zeichenfolge, die als API-Schlüssel für die /api/service-Routen verwendet wird. Eine UUID ist hier in Ordnung. Beispiel (das nicht auf einem Server verwendet werden sollte, da es jetzt öffentlich ist): `7125a0bc-80e6-4811-bba2-e07477bd99fb`
- `ACCESS_CODE`: Das Passwort, das für den Zugriff auf den Editor verwendet wird (bis wir richtige Benutzerkonten haben). Beispiel: `somepassword`
- `DEBUG_SERVER_COLOR`: Optional. Eine CSS-Farbe für den Hintergrund des Hauptmenüs, um zu verdeutlichen, dass Sie sich auf einem speziellen Server befinden, z. B. dem Feature-Test-Server, damit Sie dort nicht versehentlich wichtige Arbeiten durchführen. Beispielwerte: `rot`, `#8B0000`
- `THUMBNAIL_FOLDER`: Der Ordner für die Erzeugung von Miniaturbildern. Beispielwert: `temp/thumbnails`
- `NETWORK_DIAGNOSTICS_EXTERNAL_PING_URL`: Optional. Wird im Popup-Fenster der Netzwerkdiagnose verwendet. Eine externe URL, an die eine GET-Anfrage gesendet wird, um zu sehen, ob eine Internetverbindung besteht, auch wenn unser App-Server aus irgendeinem Grund nicht erreichbar ist. Muss das Protokoll enthalten (z. B. https://). Die angefragte Ressource muss CORS-Header haben, die unsere Anfrage akzeptieren und einen positiven Statuscode zurückgeben. Standardwert: `https://dragonlab.de/projects/adamara/available.php`
- `SESSION_COOKIE_MAX_AGE`: Optional. Legt die Zeit in Millisekunden fest, bis der Session-Cookie abläuft. Standardwert: `31536000000` (ein Jahr)

- `HELM_DEAKTIVIEREN`: Optional. Setzen Sie diesen Wert auf 1, um die Helmet-Sicherheit zu deaktivieren. Dies kann notwendig sein, wenn Sie von anderen Geräten im lokalen Netzwerk auf den Dev-Server zugreifen möchten (npm run dev). Funktioniert nur, wenn `NODE_ENV` nicht auf `production` gesetzt ist. Beispielwert: `1`
- `DEFAULT_PLAYER_NAME`: Optional, am besten leer lassen (das ist der Standardwert). Nützlich für die Entwicklung, wo man das Spiel oft neu startet, ohne jedes Mal einen Namen eingeben zu wollen. Beispielwert: `Spieler`
- `CLIENT_FORCE_PRODUCTION_MODE`: Optional; Standardwert ist leer. Setzen Sie diesen Wert auf `1`, wenn Sie lokale Debugging-Hilfen im Client-Code entfernen wollen, ohne NODE_ENV auf production zu setzen, z.B. wenn Sie einen lokalen Server im Netzwerk betreiben. Beispielwert: `1`
- Protokollierung:
  - `LOG_FOLDER`: Optional. Der Ordner, in den die Protokolldateien geschrieben werden sollen. Wenn nicht festgelegt, werden keine Protokolldateien geschrieben. Beispielwert: `logs`
  - `LOG_LEVEL`: Optional. Die Standardprotokollebene. Standardwert ist `info`, wenn er nicht gesetzt ist. Werte: `error`, `warn`, `info`, `http`, `verbose`, `debug`, `silly`
  - `LOG_LEVEL_CONSOLE`: Optional. Überschreibt `LOG_LEVEL` für die Konsolenausgabe.
  - `LOG_LEVEL_FILE_COMBINED`: Optional. Überschreibt `LOG_LEVEL` für die `combined.log`-Datei.
  - `LOG_LEVEL_NEW_RELIC`: Optional. Überschreiben Sie `LOG_LEVEL` für Protokolle, die an NewRelic.
- Sentry-Integration. Nur erforderlich, wenn Ausnahmen an Sentry gesendet werden sollen:
  - `SENTRY_ENV`: Der Name der Sentry-Umgebung, unter dem Berichte von diesem Server abgelegt werden sollen. Beispielwert: `feature-test-app`

- `SENTRY_DSN`: Die DSN-URL, an die dieser Build Bericht erstatten soll. Example value: `https://d1054791c29a4a6793ac2b9e293ae8cc@o1133894.ingest.sentry.io/6180776`
- `SENTRY_AUTH_TOKEN`: Das Token einer internen Integration (Teameinstellungen -> Entwicklereinstellungen -> Neue interne Integration) mit Lese-/Schreibzugriff auf das Projekt. Wird für das Hochladen der Source Maps in Sentry verwendet. Example value: `143dfagh148944fcb4be46634ca6c9a234037135038c496040f6606848e91fae`
- `SENTRY_ORG`: Slug der Sentry-Organisation. Wird für das Hochladen der Quellkarten in Sentry verwendet. Beispielwert: `cultures-interactive-ev`
- `SENTRY_PROJECT`: Sentry-Projekt-Slug. Wird für das Hochladen der Source Maps in Sentry verwendet. Beispielwert: `adamara`

- Atlas-Erstellung. Wird nur benötigt, wenn die Atlas-Generierung verwendet werden soll (was vielleicht nie ratsam ist - wir haben dies einmal getestet, aber es führte tatsächlich zu schlechterer Leistung und Abstürzen bei größeren Karten):
  - `ATLAS_FOLDER`: Der Ordner, in dem die generierten Atlanten gespeichert werden. Wenn sich in diesem Ordner keine Datei mit dem Namen „delete_this_file_to_regenerate_atlases_on_start“ befindet (die am Ende der Atlasgenerierung automatisch erstellt wird), werden beim Start der App neue Atlanten generiert. Die Atlasgenerierung funktioniert nur, wenn auch `THUMBNAIL_FOLDER` gesetzt ist. Beispielwert: `temp/atlases`
  - `ATLAS_SIZE`: Optional. Wie groß jede Seite des Atlasses in Pixeln sein soll. Sollte eine Potenz von 2 sein. Standardwert: `2048`
  - `MIN_ATLAS_FILE_COUNT_PER_BATCH`: Optional. Wie viele Atlasdateien pro Stapel mindestens gefüllt werden sollen. Niedrigere Werte führen zu einem weniger optimalen Ergebnis, da insgesamt mehr Atlanten erzeugt werden, führen aber zu einem geringeren Speicherverbrauch während des Erzeugungsprozesses. Um hier einen optimalen Wert zu finden, setzt man ihn am besten auf mehrere Werte und beobachtet die Ergebnisse in

Bezug auf die erzeugten Atlanten, die RAM-Nutzung und die Erzeugungszeit. Der Standardwert sollte für eine ATLAS_SIZE von 2048 sinnvoll sein. Standardwert: `20`

- Feature Switches (entweder `0` zum Ausschalten oder `1` zum Einschalten):
  - `SKIP_CULLING_UNTIL_FIRST_RENDER` Standardwert: `1`
- Anwendungsmetrik Dashboard
  - `APP_METRICS_DASHBOARD:` Wenn `APP_METRICS_DASHBOARD` auf 1 gesetzt ist, wird ein einfaches Metrik-Dashboard unter /appmetrics-dash verfügbar sein. Wenn NODE_ENV=production ist, werden auch Benutzername und Passwort (siehe unten) benötigt. Beachten Sie, dass dies nur funktioniert, wenn die optionale Abhängigkeit appmetrics-dash korrekt installiert wird - falls nicht, installieren Sie entweder das, was auf Ihrem System fehlt (wahrscheinlich node-gyp-Abhängigkeiten: https://github.com/nodejs/node-gyp#installation), oder setzen Sie den Wert nicht auf 1. Voreinstellung: `0`
  - APP_METRICS_DASHBOARD_BENUTZERNAME: Benutzername für das Dashboard. Auth wird ausgeschaltet, wenn Benutzername/Passwort nicht gesetzt sind. Beispielwert: user
  - APP_METRICS_DASHBOARD_PASSWORD: Passwort für das Dashboard. Auth ist ausgeschaltet, wenn Benutzername/Passwort nicht gesetzt sind. Beispielwert: my-auth-password
- NewRelic APM-Integration
  - `NEW_RELIC_LICENSE_KEY:` Der NewRelic-Lizenzschlüssel für Ihr Konto. Zu finden unter `User->API keys`, geben Sie „INGEST - LICENSE“ ein, `...->Copy Key`. Beispielwert: `eu01xx3c417ca3f57d924d11595e351c3e14NRAL`
  - `NEW_RELIC_APP_NAME:` Der Name, unter dem die Anwendung erscheinen soll. Beispielwert: `Adamara (Staging)`
  - Andere `NEW_RELIC_*`-Parameter: Siehe die Dokumentation zur Konfiguration des NewRelic Node.js-Agenten.
  - `LOG_LEVEL_NEW_RELIC:` Optional. Überschreiben Sie `LOG_LEVEL` für Protokolle, die an NewRelic gesendet werden.

- SSL:
  - Geben Sie das SSL-Zertifikat manuell an. Beide müssen festgelegt werden:
    - `SSL_CERT`: Wahlweise. Der Pfad zu dem SSL-Zertifikat, das für die Verwendung eines HTTPS-Servers benötigt wird. Beispiel: `cert/localhost+1.pem`
    - `SSL_KEY`: Wahlweise. Der Pfad zum Schlüssel des SSL-Zertifikats, das für die Verwendung eines HTTPS-Servers benötigt wird. Beispiel: `cert/localhost+1-key.pem`
  - Erzeugt automatisch ein SSL-Zertifikat für localhost und die aktuelle lokale IP im Netzwerk. Funktioniert nur, wenn `SSL_CERT/SSL_KEY` nicht gesetzt ist. Beide müssen gesetzt sein:
  - `SSL_GENERATE_ROOT_CERT`: Optional. Der Pfad zum SSL-Root-Zertifikat, um ein SSL-Zertifikat zu erzeugen. Beispiel:

    `C:\Benutzer\someuser\AppData\Local\mkcert\rootCA.pem`
  - `SSL_GENERATE_ROOT_KEY`: Optional. Der Pfad zum SSL-Root-Zertifikatsschlüssel, um ein SSL-Zertifikat zu generieren. Beispiel:

    `C:\Benutzer\someuser\AppData\Local\mkcert\rootCA-key.pem`
- Lokal werden diese Variablen am besten über die `.env`-Datei gesetzt, die nicht in Git eingecheckt wird. .env.template ist eine Datei, die mit den oben genannten Standard-/Formatwerten vorbelegt ist.

Wer war beteiligt?

Das Spiel Adamara entstand im Rahmen des Projekts Call of Prev, einem Modellprojekt von cultures interactive e.V. Das Projekt wurde vom Bundesministerium für Kultur und Medien gefördert.

- **Christian Kirschner:** Produktionsleitung
- **Silvia Weiß:** Ko-Produktionsleitung
- **Sven Vössler:** Ko-Produktionsleitung
- **Stefanie Ritter:** Ko-Produktionsleitung

- **Tobias Wehrum:** Hauptprogrammierer (Portfolio, LinkedIn)
- **Lena Siess:** Programmiererin (Twitter)
- **Luca Hoffmann:** Programmierer
- **Jendrik Johannes:** Programmierer (GitHub)
- **Kevin Blank:** Spieldesign (Portfolio)
- **Elham Nizam:** Spieldesign (LinkedIn, Instagram, TikTok, YouTube, Twitch, Website)
- **Jaqueline Martin:** Lead Artist (Website, LinkedIn)
- **Alexander Günther:** Game Artist / 2D-Umgebung & Requisiten (Portfolio))
- **Gastón Santibáñez Engemann:** Artist & Animator (Portfolio, LinkedIn)
- **Ilgin Özcelik:** Artist & Animator (Website, LinkedIn)
- **Kerstin Rilke:** Animation & Bewegungsdesign (Website, LinkedIn)
- **Fritz David Thiel:** Game Artist, Asset-Erstellung, Asset-Eingabe (ArtStation, LinkedIn)
- **Lena Falkenhagen:** Narrative Lead (LinkedIn, Twitter, Instagram, TikTok)
- **Ivonne Vaziri-Elahi:** Narrative Designerin / Autorin (Website, LinkedIn)
- **Ingelis Wipfelder:** Narrative Designerin / Autorin (Website, LinkedIn)
- **Jannik Jentsch:** Level-Design (LinkedIn)
- **Miriam Oumar:** Content Design (ArtStation, Portfolio)
- **Katharina Bär:** Sound Design (Website)
- **Isi Wieja:** Testerin (Twitter)
- **Marie Jäger:** Wissenschaftliche Mitarbeiterin
- **Csongor Baranyai:** UX-Designer (Website, Twitter, Mastodon)

(Fehlen Sie in der Liste? Oder möchten Sie etwas geändert haben? Schicken Sie eine E-Mail an tobias.Wehrum@dragonlab.de oder ändern Sie es selbst über einen Pull Request hier).