MicroLife-Lab

- Entwurf Vortrag
- Nutzerhandbuch MicroLife-Lab

Entwurf Vortrag

Einleitung

Hallo in die Runde, Ich möchte euch heute meine Simulationsprojekt "Microlife-Lab" vorstellen. Dieses Projekt beinhaltet eine vereinfachte, aber biologisch akkurate Simulation über das verhalten verschiedener Mikroorganismen auf einer Oberfläche mit Nährstoffen unter verschiedenen Umweltbedingungen, welche der Benutzer selber festlegen kann mit einer anschließenden grafischen Auswertung. Ich möchte in diesem Vortrag auf die Entwicklung, näher meine Entscheidungen für die Wahl der Programmiersprache und Frameworks eingehen und anschließend die Umsetzung des Projektes kurz vorstellen anschließend mit einem Fazit meinerseits.

Ausgangslage der Projektentwicklung

Konzeptionierung

Da ich auf Webentwicklung durch meine Firma spezialisiert war, und auch im vorherigen Projekt viel Erfahrung mit Node.JS, Electron und Co. sammeln konnte, entschied ich mich darauf aufzubauen.

Zur Versionsverwaltung habe ich mich für ein Repository auf Github entschieden, um immer den Aktuellen stand abrufen zu können, und auch eine Übersicht über die schon vorgenommen Arbeiten zu haben.

Neben den vorgegebenen Rahmenbedingungen an das Projekt war es zunächst wichtig eine JavaScript-Bibliothek zu finden, welche die Animation und auch die Technischen vorgaben in der Entwicklung vereinfachen. Hier bat sich die Bibliothek p5.js an, nicht nur für die Vereinfachung des Quellcodes mit integrierten mathematischen Funktionen, als auch bei der visuellen Darstellung der Simulation.

Des weiteren was es mir wichtig, für den Code und die Projektstruktur Redundanz zu vermeiden.

Hier hat sich Bootstrap und insbesondere auch das EJS (Embedded JavaScript Templates)-Framework angeboten. Durch die vorgegebene Projektstruktur durch Node.Js, konnte der Quellcode einfach und übersichtlich gehalten werden. Insbesondere gab sich das zu erkennen bei den Wachstumsfunktionen der einzelnen Mikroorganismen, und auch bei der Berechnung der Sterbe- und Vermehrungsrate. Es ergab sich nun folgende Projektstruktur:

Ein weiterer Vorteil von Node.JS ist der integrierte Paketmanager npm. Dadurch kann dieses Projekt sehr einfach installiert, gestartet und kompiliert werden.

Wahl der Datenquelle und wissenschaftliche Plausibilität

Neben einer hohen Anzahl von Wissenschaftlichen Publikationen über das Verhalten der Mikroorganismen konnte ich auch meinen Vater zu Rate ziehen, der promovierter Fachexperte in diesem Bereich ist. Durch ihn konnte ich mich beraten lassen, Fachliteratur und Ideen einholen. Durch die hohe Anzahl der Quellen werde ich im Rahmen dieses Vortrages nicht weiter darauf eingehen.

Hier sieht man einige Beispiele.

Umsetzung des Projektes

Die Umsetzung des Projektes hat sich in verschiedene Schritte aufgeteilt. Darunter:

1. Aufsetzen des Projektes / Repository erstellen etc.
2. Benutzeroberfläche gestalten und konzeptionieren
3. Implementierung des Faktors Zeit und Geschwindigkeit
4. Implementierung der Wachstums- Teilungs- und Sterbefunktionen
5. Implementierung einer grafischen Auswertung mithilfe von eCharts.

Auf die punkte 4 und 5 möchte ich hier genauer eingehen, da sie den Wesentlichen Teil der Simulation ausmachen.

Der Faktor Zeit

Um die Simulation realistisch zu halten entschied ich mich die Zeit von Stunden in Sekunden zu skalieren. Das bedeutet das jeder Iterationsschritt für den Nutzer auf der ersten Geschwindigkeitsstufe eine Stunde in der echten Welt widerspiegelt.

Zunächst wird die eingegebene Simulationszeit als Parameter entsprechend verarbeitet und jeweils in Minuten und Stunden umgerechnet. Durch die Timescale-Variable konnten entsprechende Geschwindigkeitsstufen realisiert werden.

```
switch(activeButtonId) {
    case "normal": timeScale = 1; break;
    case "fast": timeScale = 2; break;
    case "faster": timeScale = 3; break;
}
```

Und die Geschwindigkeit, mit der die Zeit fortschreitet:

```
simulationTime += (timeScale * SIMULATION_UPDATE_INTERVAL) / 60;
```

Diese Funktion berechnet das Zeitintervall für die Simulation. Die `1000` steht hier für 1000 Millisekunden, also 1 Sekunde. Sie wird durch `timeScale` geteilt, um die Geschwindigkeit der Simulation anzupassen.

Es wird überprüft, ob es einen vorherigen Index gibt (`currentIndex > 0`). Wenn ja, wird an dieser Stelle ein Markierungssymbol ("|") gesetzt.
Dann wird geprüft, ob der aktuelle Index noch innerhalb der Zellen-Anzahl liegt.
Wenn ja, wird an der aktuellen Position ein Cursor-Symbol gesetzt, der Index erhöht und die Simulationszeit um 3600 Sekunden (1 Stunde) erhöht.
Wenn der Index das Ende erreicht hat, wird das Intervall gelöscht und die Simulation beendet.

`getIntervalTime()` wird als Argument für `setInterval` verwendet, um die Geschwindigkeit der Simulation zu steuern.

Schließlich wird `startSimulation()` aufgerufen, um die Simulation zu beginnen.

```
const getIntervalTime = () => {
    return 1000 / timeScale;
};

function startSimulation() {
    interval = setInterval(() => {
        if (currentIndex > 0) {
            cells[currentIndex - 1].innerHTML = `<span class="marker">|</span>`;
        }
        if (currentIndex < cells.length) {
            cells[currentIndex].innerHTML = `<span class="cursor"></span>`;
            currentIndex++;

            simulationTime += 3600;

        } else {
            clearInterval(interval);
            simulationActive = false;
            console.log("Simulation beendet");
        }
    }, getIntervalTime());
}
startSimulation();
```

Die Wachstumsraten:

```
baseGrowthRate = calculateGrowthRateEscherichiaColi(temperature, concentration, ph, moisture, p) *
timeScale;
```

Die Mutationen:

```
const lambda = 0.2; // Durchschnittlich 1 Mutation alle 5 Zeiteinheiten
if (randomExponential(0.2) < 0.1) {
    this.mutate();
}
```

Die Zellteilung:

```
if (this.age > DIVISION_AGE_THRESHOLD && this.size > DIVISION_SIZE_THRESHOLD && p.random() <
divisionRate * capacityFactor) {
    this.divide();
}
```

Der Tod:

```
const deathProbability = calculateDeathRate(this.age, microbes.length, ph, moisture, temperature,
microOrganism, p);
if (p.random() < deathProbability) {
    this.death();
}
```

Zusammenfassend wichtige Aspekte:

- Jede Iteration repräsentiert eine Stunde (3600 Sekunden)
- Die Simulationsgeschwindigkeit bestimmt, wie oft dieser Schritt pro Sekunde ausgeführt wird
- Alle biologischen Prozesse (Wachstum, Zellteilung, Tod) werden dynamisch mit der simulierten Zeit skaliert.

Wachstums- Teilungs- und Sterbefunktionen

Bei den ausschlaggebenden Funktionen für das Verhalten der Mikroorganismen wurden wissenschaftliche Quellen genutzt. Alle diese Funktionen wurden der Übersichtlichkeit halber in

einer separaten Datei gespeichert und an die entsprechende Hauptlogik durchgereicht.

Beispiel für die Berechnung des Wachstums von Candida-Albicans:

```
export function calculateGrowthRateCandida(temperature, concentration, ph, moisture,p) {
    let baseGrowthRate;

    if (temperature <= 0 || temperature > 50) {
        baseGrowthRate = 0;
    } else if (temperature > 0 && temperature < 5) {
        baseGrowthRate = 0.001;
    } else if (temperature >= 5 && temperature < 20) {
        baseGrowthRate = p.map(temperature, 5, 20, 0.01, 0.05);
    } else if (temperature >= 20 && temperature <= 37) {
        baseGrowthRate = p.map(temperature, 20, 37, 0.05, 0.2);
    } else if (temperature > 37 && temperature <= 50) {
        baseGrowthRate = p.map(temperature, 37, 50, 0.2, 0.01);
    }

    // Nährstoffkonzentration beeinflusst die Wachstumsrate linear (z.B. von 0 bis 100 %)
    // Konzentration in Prozent, 0 % = kein Wachstum, 100 % = volles Wachstum
    let pHFactor = 1 - Math.abs(ph - 5.5) / 5 // Optimal pH around 5.5
    pHFactor = p.constrain(pHFactor, 0.1, 1)

    let moistureFactor = p.map(moisture, 0.3, 0.9, 0, 1)
    moistureFactor = p.constrain(moistureFactor, 0, 1)

    const nutrientFactor = concentration / 100

    return baseGrowthRate * nutrientFactor * pHFactor * moistureFactor
}
```

Hier werden die Eingabeparameter alle mit berücksichtigt und jede Wachstumsfunktion spiegelt die für den entsprechenden Mikroorganismus wieder.

Hier macht sich deutlich inwiefern ich p5.js genutzt habe.

- p.abs gibt den Absoluten wert von x zurück.
- p.constrain begrenzt einen Wert auf einen bestimmten Bereich
- p.map skaliert einen Wert von einem Wertebereich

Die Funktion für den Zelltod

```
export function calculateDeathRate(age, crowding, pH, moisture, temperature, microOrganism) {

    const BASE_GROWTH_RATE = 0.0001
    const CROWDING_FACTOR = 0.001
    const MAX_MICROBES = 2000

    let baseDeathRate = BASE_GROWTH_RATE * (age/500);
    let crowdingFactor = (crowding / MAX_MICROBES) * CROWDING_FACTOR;
    let tempStress = 0;
    let pHStress = 0;
    let moistureStress = 0;

    switch(microOrganism) {
        case "candida":
            if (temperature < 20 || temperature > 45) tempStress = 0.0003;
            if (pH < 2 || pH > 10) pHStress = 0.0002;
            if (moisture < 0.8) moistureStress = 0.0004;
            break;
        case "aspergillus":
            if (temperature < 6 || temperature > 47) tempStress = 0.0004;
            if (pH < 2 || pH > 11) pHStress = 0.0001;
            if (moisture < 0.77) moistureStress = 0.0003;
            break;
        case "penicillium":
            if (temperature < 4 || temperature > 37) tempStress = 0.0003;
            if (pH < 3 || pH > 8) pHStress = 0.0002;
            if (moisture < 0.80) moistureStress = 0.0004;
            break;
        case "ecoli":
            if (temperature < 7 || temperature > 46) tempStress = 0.0005;
            if (pH < 4.4 || pH > 9) pHStress = 0.0003;
            if (moisture < 0.95) moistureStress = 0.0002;
            break;
        case "staphylococcus":
            if (temperature < 7 || temperature > 48) tempStress = 0.0004;
            if (pH < 4 || pH > 10) pHStress = 0.0002;
            if (moisture < 0.86) moistureStress = 0.0003;
            break;
    }
    return baseDeathRate + crowdingFactor + tempStress + pHStress + moistureStress;
```

```
}
```

Hier wird ebenfalls für jeden entsprechenden Mikroorganismus die Todesrate berechnet. Dazu zählen nicht nur die Umweltbedingungen, sondern auch der Crowding-Faktor welcher bei einer zu hohen Anzahl/Konzentration von Mikroorganismen die Todesrate beeinflusst.

Die Funktion für die Teilung

```
export function calculateDivisionRate(microOrganism, temperature, pH, moisture, concentration, p) {
    let baseDivisionRate = 0.005;
    let tempFactor = 1;
    let pHFactor = 1;
    let moistureFactor = 1;
    let nutrientFactor = concentration / 100;

    switch(microOrganism) {
        case "candida":
            tempFactor = p.map(temperature, 20, 37, 0.5, 1);
            pHFactor = 1 - Math.abs(pH - 5.5) / 5;
            moistureFactor = p.map(moisture, 0.3, 0.9, 0.5, 1);
            break;
        case "aspergillus":
            tempFactor = p.map(temperature, 20, 35, 0.5, 1);
            pHFactor = 1 - Math.abs(pH - 5.5) / 5.5;
            moistureFactor = p.map(moisture, 0.15, 0.85, 0.5, 1);
            break;
        case "penicillium":
            tempFactor = p.map(temperature, 15, 25, 0.5, 1);
            pHFactor = 1 - Math.abs(pH - 5.5) / 5;
            moistureFactor = p.map(moisture, 0.2, 0.8, 0.5, 1);
            break;
        case "ecoli":
            tempFactor = p.map(temperature, 30, 37, 0.5, 1);
            pHFactor = 1 - Math.abs(pH - 7) / 5;
            moistureFactor = p.map(moisture, 0.3, 0.95, 0.5, 1);
            break;
        case "staphylococcus":
            tempFactor = p.map(temperature, 25, 35, 0.5, 1);
            pHFactor = 1 - Math.abs(pH - 7) / 6;
            moistureFactor = p.map(moisture, 0.25, 0.9, 0.5, 1);
            break;
```

```
    }

    tempFactor = p.constrain(tempFactor, 0.1, 1);
    pHFactor = p.constrain(pHFactor, 0.1, 1);
    moistureFactor = p.constrain(moistureFactor, 0.1, 1);

    return baseDivisionRate * tempFactor * pHFactor * moistureFactor * nutrientFactor;
}
```

Fazit und Problemstellungen

Letztendlich war die Unterstürzung meines Vaters Fluch und Segen zugleich. Durch sein tiefgreifendes Wissen gab es Diskrepanzen zwischen dem Machbaren oder in der vorgegebenen Zeit realisierbaren.

So wurde ich oft dazu gebracht meinen gesamten Ansatz nochmal zu überdenken und ich habe verschiedene Ansätze probiert. Es ging sogar so weit, dass ich die Echtgröße z.B. 0.05 Mikrometer der einzelnen Mikroorganismen wie in der Realität skalieren wollte. Dies hätte nicht nur Performanzprobleme bereitet, hätte aber auch zu einem kompletten Umdenken der gesamten Simulation geführt. Hierfür war die Zeit nicht ausreichend. Es hat sich als viel zu Kompliziert dargestellt, und entsprechendes tiefgreifendes Wissen in der Mikrobiologie obliegt mir selber nicht. Diese Simulation bietet jetzt zwar ein auf wissenschaftlichen Fakten basiertes Modell, ist aber jedoch aus Sicht von Fachexperten nur die Spitze des Eisberges.

Was ich gerne noch eingebaut hätte ist eine konkrete Definition der Größe der Mikroorganismen, eine verbesserte Aussagekraft beim setzen des Grenzwertes (hier hätte ich mit KBE im Bereich von 10^5 bis z.B. 10^8 arbeiten müssen).

Nutzerhandbuch MicroLife-Lab

1. Einleitung

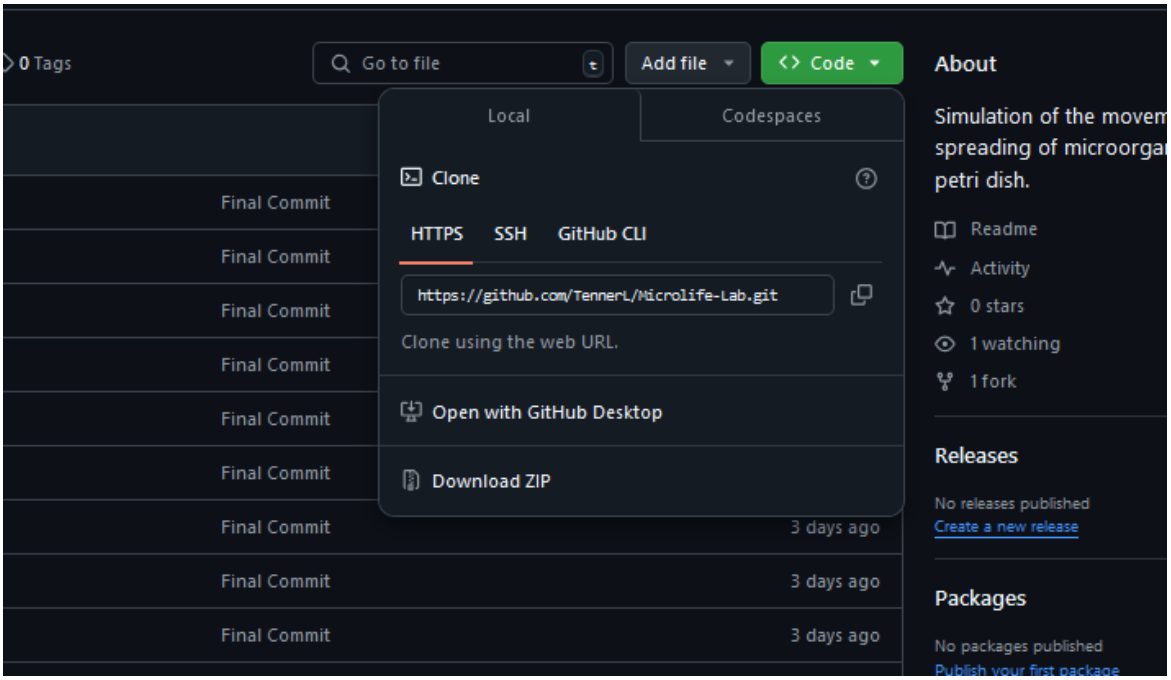
Das Ihnen vorliegende Programm erfüllt den Zweck, Wachstumsprozesse von Mikroorganismen anhand von simulierten Umweltbedingungen grafisch darzustellen.

Für dieses Programm in seiner Ausführbaren Form, sind keine Programme notwendig. Wenn Sie jedoch das Programm aus dem Quellcode ausführen und selber kompilieren wollen, benötigen Sie Node.JS auf ihrem lokalen PC.

2. Installation

Das Programm läuft auf jedem handelsüblichen Computer mit einem Betriebssystem von Windows 10 oder neuer und einem minimalen freien Speicherplatz von 1 Gigabyte.

1. Laden Sie das Repository als .zip von diesem Link herunter, und entpacken Sie dieses an einem beliebigen Ort: https://github.com/TennerL/Microlife-Lab



2. Laden Sie Node.JS unter diesem Link herunter, und installieren sie die Anwendung mit den Standardwerten: https://nodejs.org/dist/v22.14.0/node-v22.14.0-x64.msi

3. Drücken Sie jetzt die "Windowstaste" in Kombination mit der Taste "R" und geben Sie "cmd" ein. Ein Schwarzes Eingabefenster sollte erscheinen.

4. Wechseln Sie nun in das Verzeichnis, in dem sie die Anwendung entpackt haben mit dem Befehl: cd <Pfad der Anwendung> und drücken Sie die Eingabetaste.

5. Installieren Sie jetzt die Abhängigkeiten mit dem Befehl: npm install

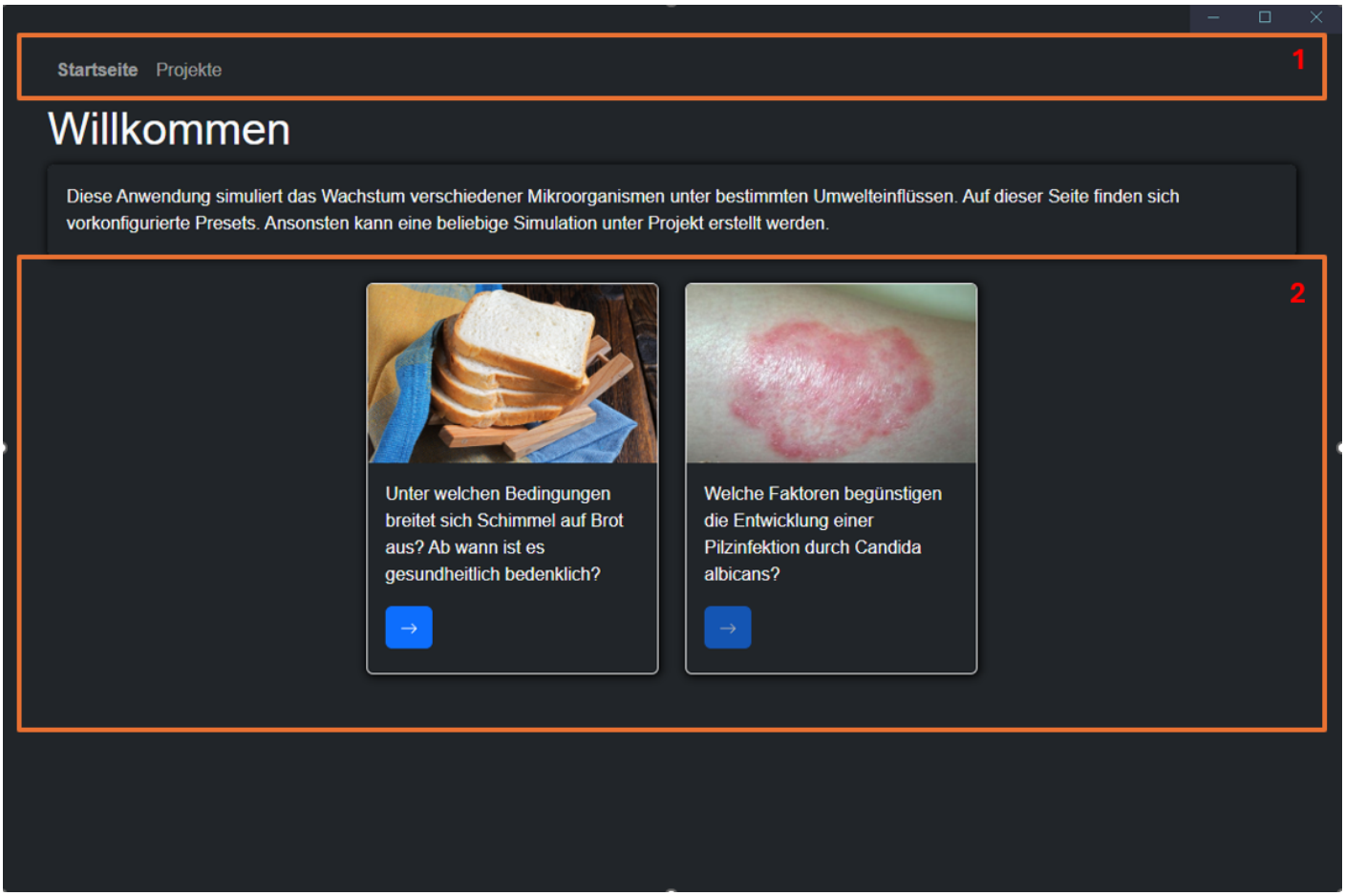
6. Anschließend können sie die Anwendung kompillieren mit dem Befehl: npm run build.

7. In dem Programmverzeichnis sollte jetzt ein Ordner namens dist sein. Hier finden Sie sowohl das Setup, falls Sie die Anwendung installieren wollen als auch die ausführbare Datei "MicroLife-Lab.exe" in dem Unterordner win-unpacked.

3. Benutzeroberfläche

Willkommens- und Konfigurationsansichten

Beim Start des Programms, werden Sie sich auf der Willkommensseite wieder finden.

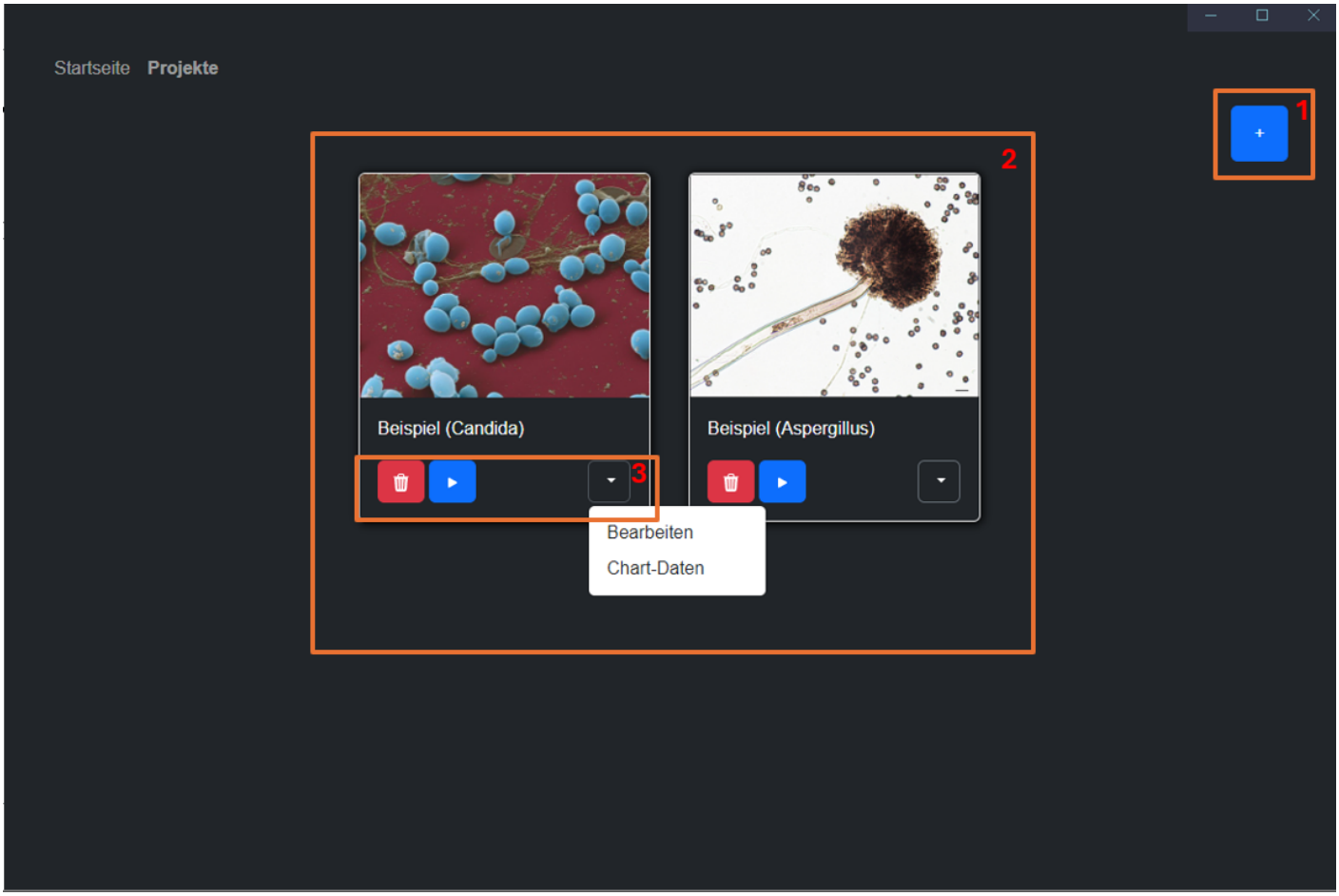


Auf dieser Seite finden Sie folgende Elemente vor:

1. Die Navigationsleiste, bei der Sie zwischen verschiedenen Ansichten navigieren können.

2. Zwei Beispiele für den einfachen Einstieg in das Programm. Sind die Buttons nicht anklickbar, ist das Projekt bereits erstellt. Beim Klick auf den Button öffnet sich ein Dialog mit einigen Informationen zu dem Mikroorganismus. Klicken Sie auf "Speichern", wenn Sie das Projekt erstellen möchten. Sie werden dann automatisch an die Projektseite weitergeleitet.

Die Projektseite besteht aus folgenden Elementen:



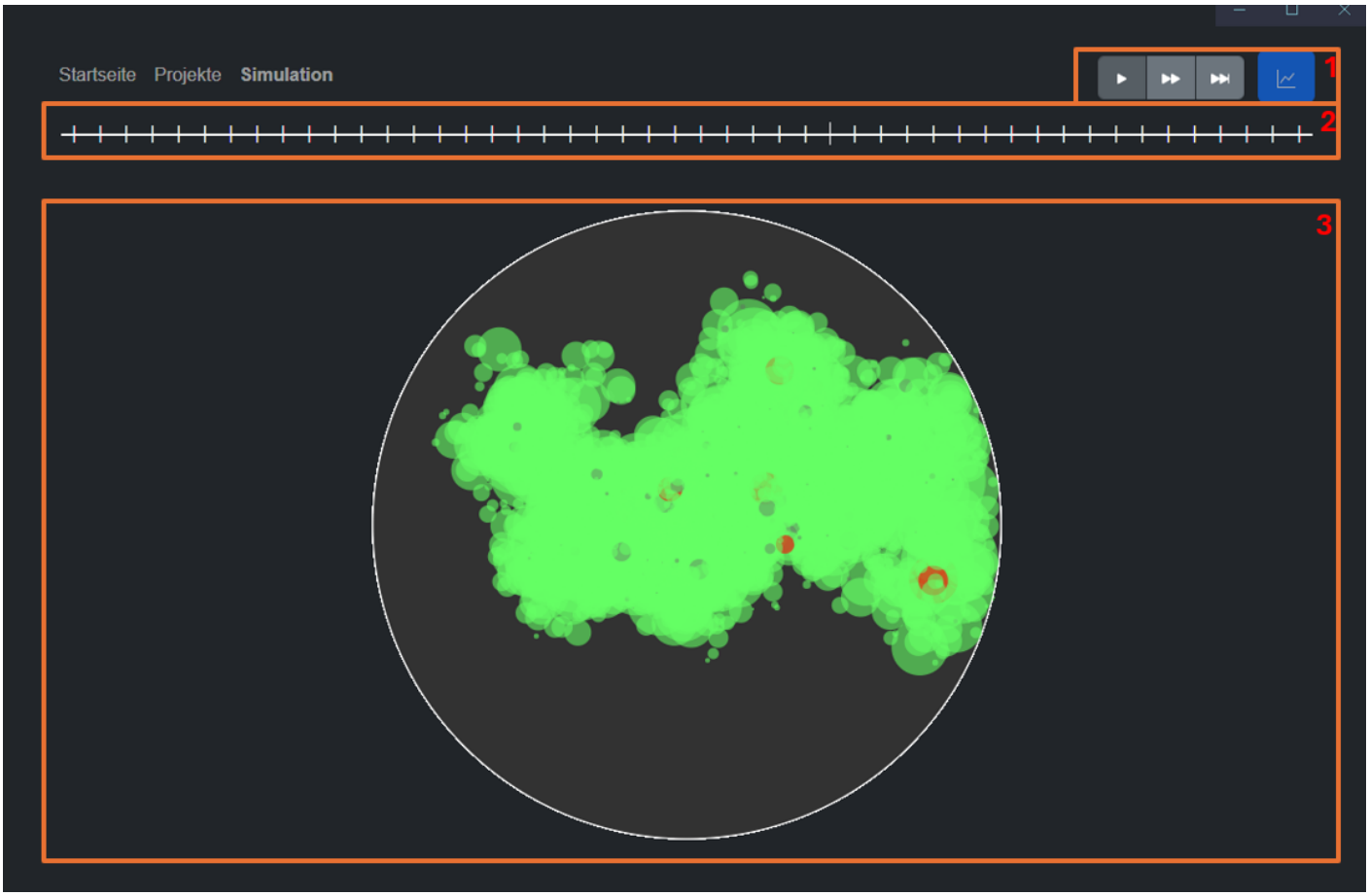
1. Bei Klick dieses Buttons öffnet sich auf der gleichen Seite ein Fenster, bei dem sie ein Projekt angelegen können.
2. Hier finden Sie alle angelegten Projekte vor. Ist kein Projekt vorhanden, wird Ihnen das auch durch einen kurzen Text mitgeteilt.
3. Hier sind die Kontrollelemente für jedes Projekt. Sie können ein Projekt starten, löschen, bearbeiten oder sich die Diagrammdaten anschauen, sofern Sie in der Vergangenheit die Simulation bereits gestartet haben.

Wenn Sie sich entscheiden ein Projekt zu erstellen oder zu bearbeiten, bekommen Sie folgende Ansicht zu sehen:

Hier können Sie die entsprechenden Parameter eintragen. Wenn sie ungültige Werte eingeben, werden Sie mithilfe eines Fensters dazu angewiesen die Werte entsprechend anzupassen. Beachten Sie, dass dieses Fenster nahezu identisch mit dem Ändern eines Projektes ist.

Simulationsansicht

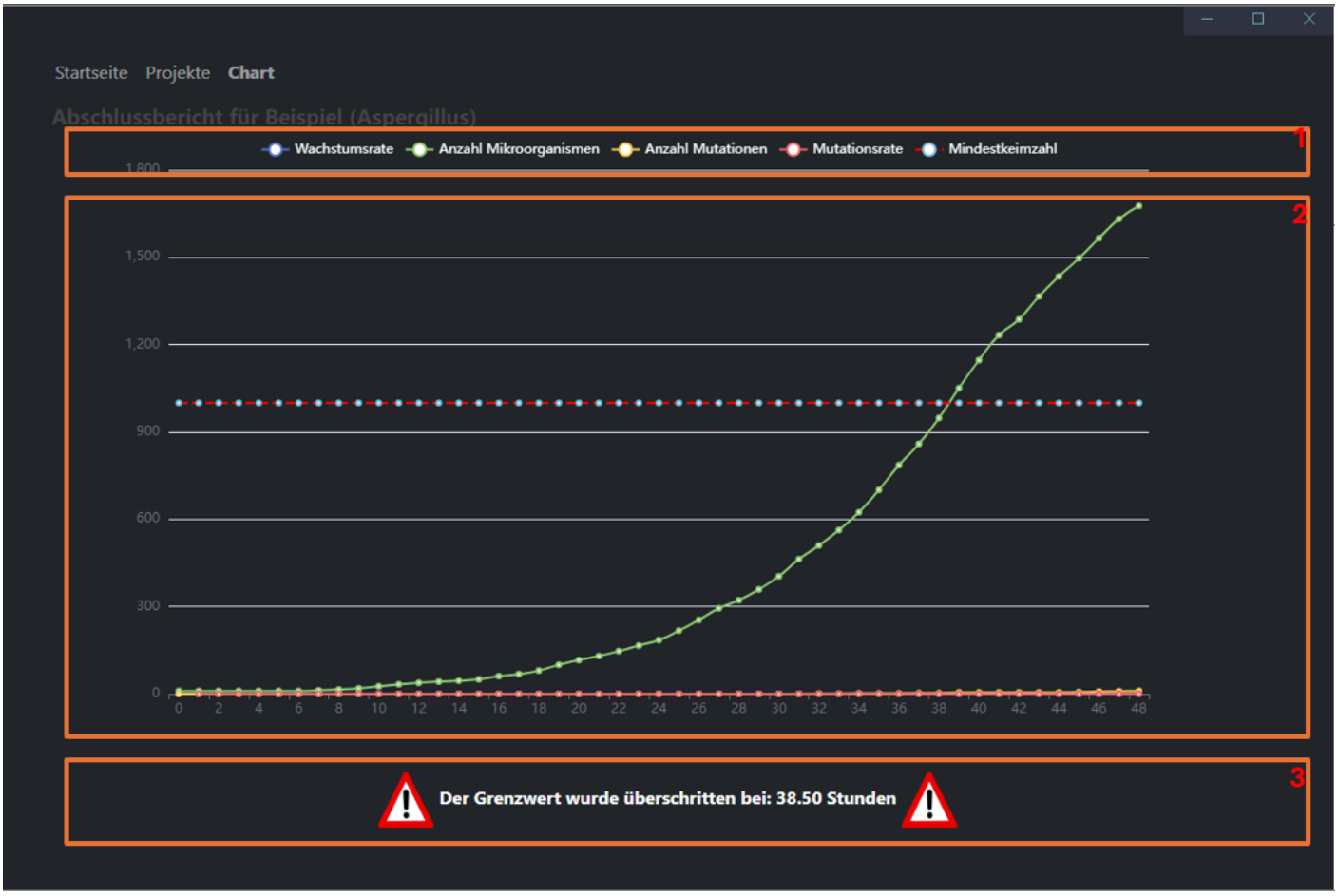
Die Simulationsansicht gliedert sich in folgende Teile:



1. Hier befinden sich die Buttons zur Kontrolle der Zeit in 3 Zeitstufen, und ein Button um sich die Auswertung anzusehen. Dieser Button wird erst aktiviert, nachdem die eingestellte Zeit abgelaufen ist.
2. Hier sehen Sie die Zeitleiste. Jeder Strich symbolisiert eine Stunde. Sie können auch erkennen, in welchem Zeitschritt Sie sich gerade befinden durch den sich bewegenden Strich.
3. Hier sehen Sie die Ausbreitung der Mikroorganismen. Rote punkte symbolisieren hierbei eine Mutation und die grauen Punkte stellten tote Zellen dar.

Auswertung

Die Auswertung erfolgt in Form eines Diagramms.



1. Hier können Sie sich einzeln die Simulationsdaten auf dem Diagramm anschauen. Es wird der Schnittpunkt mit dem eingegebenen Grenzwert deutlich, sofern dieser überschritten wurde.
2. Das Diagramm bietet Tooltips für jeden Iterationsschritt, wenn Sie Ihren Mauszeiger über einen beliebigen Punkt bewegen.
3. Hier sehen Sie wann und ob der Grenzwert überschritten wurde, und auch den exakten Zeitpunkt.