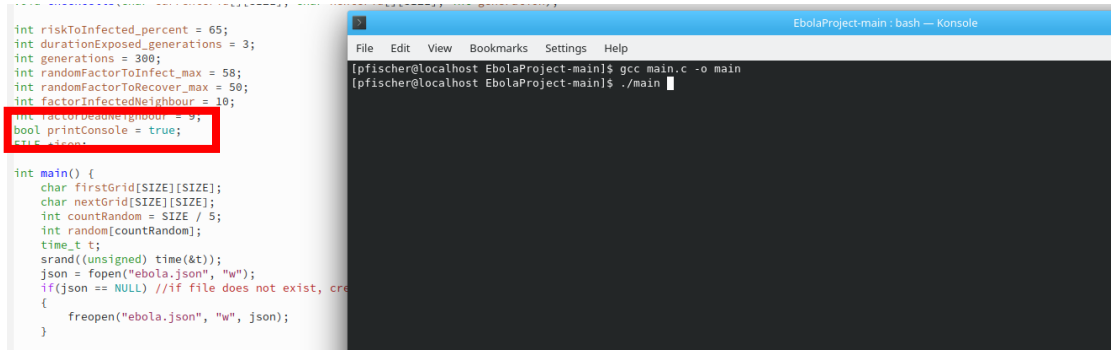
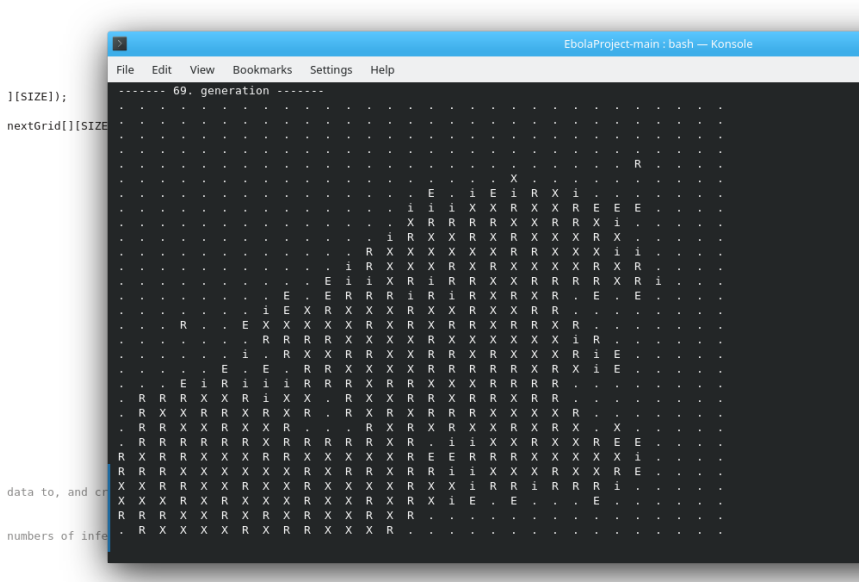
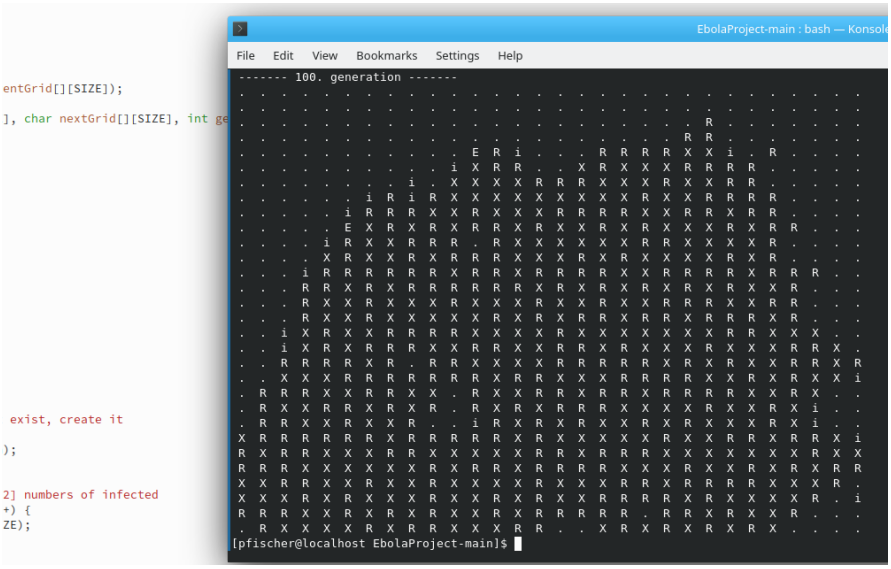
SERIAL AND VISUALISATION NOTES

Peter kept us updated with the Serial and Visualisation while coding this. Peter would talk about the code and we all have this document that peter had kept for us to keep up to date and talk about the with the Serial and Visualisation.

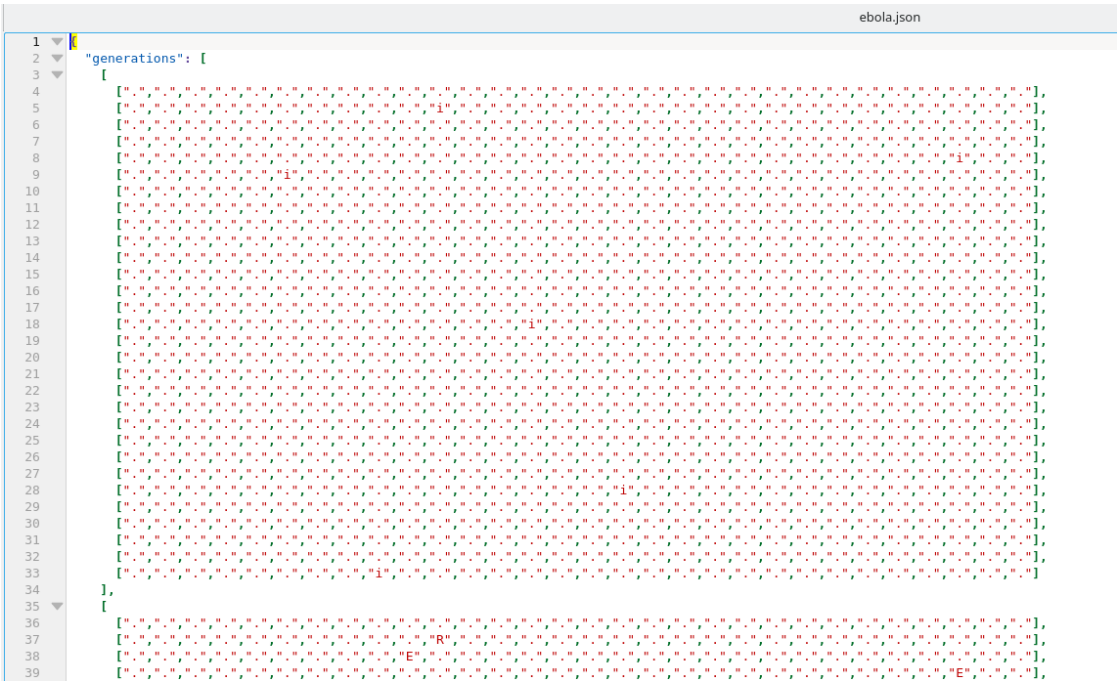


Compiling the C file using the command “`gcc main.c -o main`” creates an executable called “main”. If the variable “printConsole”.





The screenshot is one generation of the cells as JSON object. json in this format way of visualising data, as the file visualise the state of each generation itself. Therefore, it is very easy to read and to understand, but also needs more space because of many quotation marks and square brackets. Moreover, we needed this format for the visualisation program, which is built using a JavaScript framework.



The gnuplot.dat file is a data file for gnu-plotting the data. The numbers represent the format:

```
generation susceptible exposed infected recovered dead
```

gnuplot.dat

```
0 894 0 6 0 0
1 892 2 0 3 3
2 892 2 0 3 3
3 892 2 0 3 3
4 892 0 2 3 3
5 890 2 2 3 3
6 890 2 2 3 3
7 889 1 2 3 5
8 887 3 2 3 5
9 885 5 2 3 5
10 885 0 5 3 7
11 880 5 5 3 7
12 878 7 5 3 7
13 874 4 7 4 11
14 869 9 7 4 11
15 865 13 7 4 11
16 860 5 13 9 13
17 852 13 13 9 13
18 850 15 13 9 13
19 844 6 15 14 21
20 838 12 15 14 21
21 832 18 15 14 21
22 826 6 18 22 28
23 816 16 18 22 28
24 812 20 18 22 28
25 805 7 20 27 41
26 794 18 20 27 41
27 785 27 20 27 41
```

VISUALISATION PROGRAM

Peter wrote the visualisation program using the JavaScript framework Vue JS. It allows him to create dynamically rendered html he told us . The IDE WebStorm from the company JetBrains was used. Basically, the program reads a JSON file using an Input field. After loading the file, a grid is created using Vue's integrated for loops for html code. One loop iterates over the rows, one inner loop iterates over the columns and creates Elements with the colour depending on the current Cell value. This is repeated for every generation, which creates the effect of watching a video of the ebola spreading process.

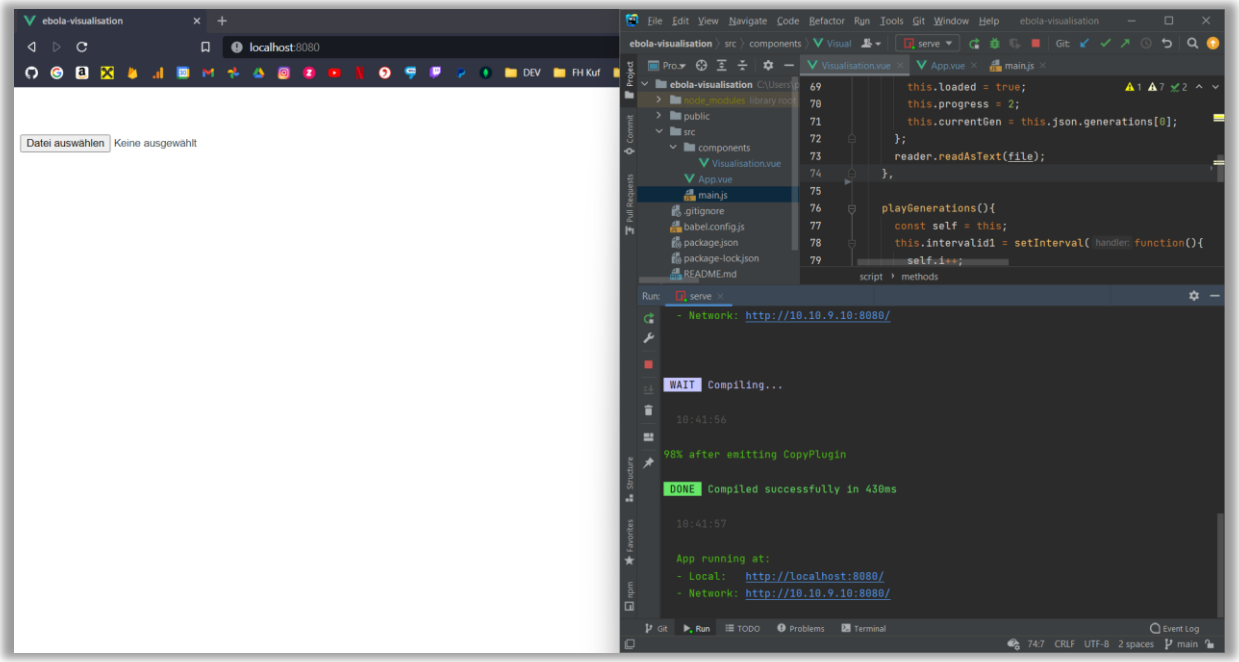
Visualisation.vue

```
<template>
  <div class="hello">
    <input type="file" accept="application/json" @change="onFileChange">
    <div v-if="progress === 1">
      <div class="lds-grid">
        <div></div>
        <div></div>
        <div></div>
        <div></div>
        <div></div>
        <div></div>
        <div></div>
        <div></div>
        <div></div>
      </div>
    </div>
    <div v-if="loaded">
      <button @click="playGenerations">start</button>
      <button @click="activePlay = false">stop</button>
      <div :style="`width: ${json.generations[0].length*sizeOfCell_px}px; height: ${json.generations[0].length*sizeOfCell_px}px`">
        <div v-for="(row,indexRow) in json.generations[i]" :key="indexRow"
             style="display: flex"
             :style="`width: ${json.generations[0].length*sizeOfCell_px}px;height: ${sizeOfCell_px}px`">
          <div
              v-for="(cell,indexCol) in json.generations[i][indexRow]"
              :key="indexCol"
              :style="{'background': colors[cell], 'height': sizeOfCell_px+'px', 'width': sizeOfCell_px+'px'}"
          >
          </div>
        </div>
```

```
      </div>
    </div>

  </div>
</template>

<script>
export default {
  name: 'HelloWorld',
  data() {
    return {
      json: {},
      loaded: false,
      sizeOfCell_px: 10,
      colors: {
        'i': '#F00',
        '.': '#0F0',
        'X': '#000',
        'R': '#00F'
      },
      progress: 0,
      currentGen: [],
      i: 0,
      intervalid1: null
    }
  },
  methods: {
```

```
    onFileChange(e) {
      let files = e.target.files || e.dataTransfer.files;
      if (!files.length) return;
      this.readFile(files[0]);
      this.loaded = false;
      this.progress = 1;
    },
    readFile(file) {
      let reader = new FileReader();
      reader.onload = e => {
        this.json = JSON.parse(e.target.result);
        this.loaded = true;
        this.progress = 2;
        this.currentGen = this.json.generations[0];
      };
      reader.readAsText(file);
    },

    playGenerations(){
      const self = this;
      this.intervalid1 = setInterval( handler: function(){
        self.i++;
      }, timeout: 500);
    },

    stopGenertions(){
      clearInterval(this.intervalid1);
    }
  }
}
```

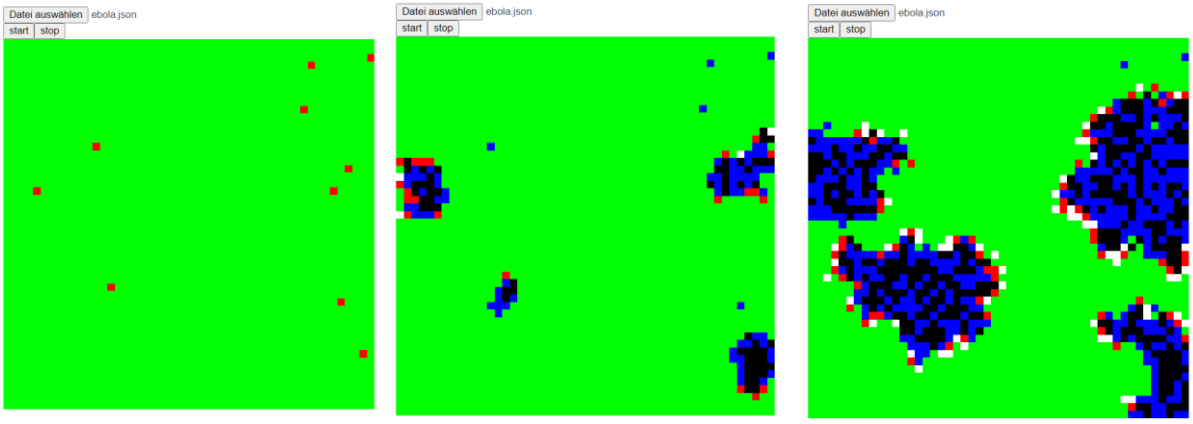
On the left is the result of running the Vue Js application on my localhost. On the right is the IDE.



When a file is uploaded and currently being processed by JavaScript, a loading animation gets displayed. “Datei auswählen” is German for “choose file”.

The buttons “start” and “stop” are starting / stopping the animation. The generation gets refreshed every 0,5 seconds. The following screenshots show the progress of the spread with coloured cells.

Green= susceptible, white: exposed, red: infected blue: recovered, black: dead



Inspecting the site with Chrome’s developer tools indicates, that every cell is a square div in another div. The background colour depends on the cells current state and gets updated every 0.5 seconds. So, every generation is displayed 0.5 seconds, before it gets replaced by the next one.