Entwickler Dokument

Code

Formatting

Damit eine einheitliche Formatierung gewährleistet ist, wurde im Projekt mit „Prettier“ gearbeitet. Prettier ist ein Codeformatierer, welcher Code in einen einheitlichen Stil formatiert und häufig in der Webentwicklung verwendet wird. Es wurde eine Konfigurationsdatei „.prettierrc“ im Root Verzeichnis erstellt, in der Regeln zur Formatierung festgelegt wurden. Zusätzlich konnte jeder Entwickler der VSCode als Editor verwendet hat die Prettier Extension installieren, welche beim Speichern den Code direkt formatiert. Die Extension sucht automatisch nach einer Konfigurationsdatei im Projekt Kontext und verwendet die dort festgelegten Regeln oder nutzt die Standard Regeln in in der Extension Config festgelegt wurden.

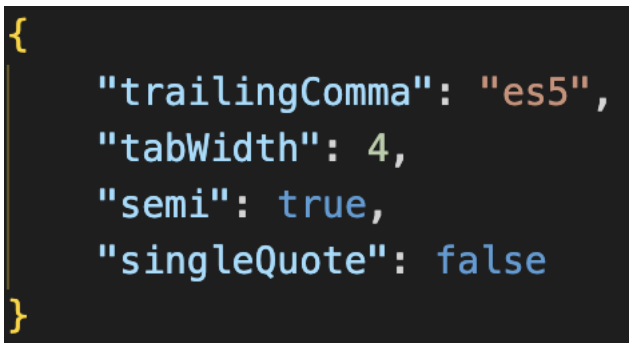

Abbildung 1".prettierrc" im Root Verzeichnis

Statische Codeanalyse

Da wir das Framework NextJS verwenden wurde „ESLint“ automatisch mit installiert. ESLint ist ein Werkzeug zur statischen Codeanalyse für JavaScript-Code. Es ermöglicht Codequalitätsregeln zu definieren und anzuwenden, um potenzielle Fehler, Stilprobleme und ineffiziente Codestrukturen zu identifizieren. Wir haben das vorgefertigte Regelset von NextJS verwendet, um sicherzustellen, dass der Code den Best Practices entspricht und konsistent ist. Es gibt auch hier eine VSCode Extension die wir genutzt haben um Echtzeit Feedback im Editor zu erhalten.

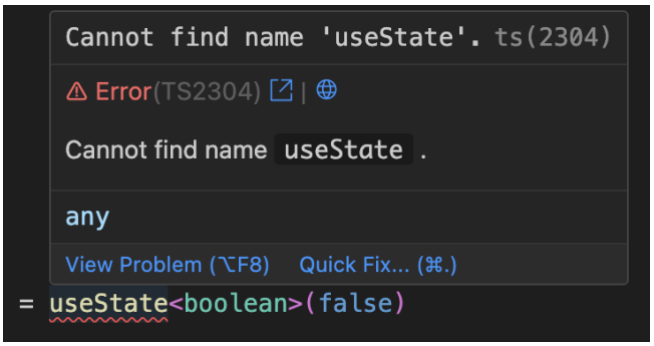

Abbildung 2 Editor Fehlermeldung

Verzeichnisstruktur

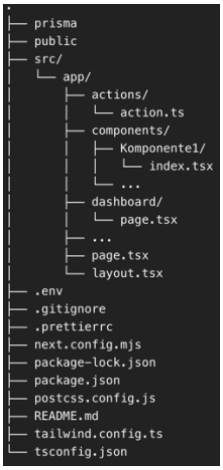

Abbildung 3 Unsere Grund Verzeichnisstruktur

Unser Framework NextJS legt viel fest, wie unser Verzeichnis auszusehen hat. Das liegt zu einen daran das NextJS die Ordnerstruktur für das Routing mappt. Im Root Verzeichnis befinden sich Konfigurationsdateien und Ordner für andere Systeme, wie z.B. unser ORM-Prisma. Damit ein Ordner als Route angesehen wird muss dieser im app Directiory liegen und eine Datei „page.tsx“ haben, die eine React Komponente exportiert welche bei Aufruf der Seite gerendert wird. Komponenten liegen bei uns auch in app/components. Für weitere Informationen, z.B weitere reservierte Dateinamen und Next Funktionen, kann man die NextJS Dokumentation lesen.

Programing Guide

Da wir React in Kombination mit Typescript verwendet haben, wurde vorher festgehalten, wie eine React Komponente „aussehen“ soll, damit wir die Vorteile von Typescript haben wie z.B. Intellisense oder Linting. Eine Komponente soll nach dem folgendem Schema erstellt werden.

Anleitung Komponentenerstellung:

1. Ordner in „src/app/components“ mit einer „index.tsx“, welche die Haupt Komponente exportiert
2. Ein Interface anlegen, mit allen „props“ der React Komponente
3. Eine Variable (bennant nach der Komponente) der eine Funktionale Komponente mit Arrow Function Syntax zugewiesen wird, die das erstellte interface nutzt und die props als Parameter destructured.
4. Die Variable als default export

```tsx
import React from 'react'

interface ITestComponent {
  prop1: string
}

const TestComponent : React.FC<ITestComponent> = ({prop1}) => {
  return (
    <div>{prop1}</div>
  )
}

export default TestComponent
```

Abbildung 4 Beispiel Komponente

Versionskontrolle und Workflow

Für die Versionskontrolle wurde GitHub genutzt. Wir haben mit einem leicht angepassten Git Workflow gearbeitet, der sich „Common Flow“ nennt.

Im Prinzip gibt es einen „main“ Branch dieser hat immer den aktuellen „release“. Wenn ein Feature implementiert werden soll, wird ein Branch der vom Main abzweigt erstellt. Auf diesem neuen Branch werden alle Änderungen, die zur Umsetzung notwendig sind, in Form von „commits“ gesammelt. Bei fertiger Implementation des Features wird ein Pull request erstellt, anschließend wird ein Review gemacht. Wenn der Main bereits fortgeschritten ist, muss noch rebased werden bevor der Feature Branch in den Main Branch zurück gemerged wird.

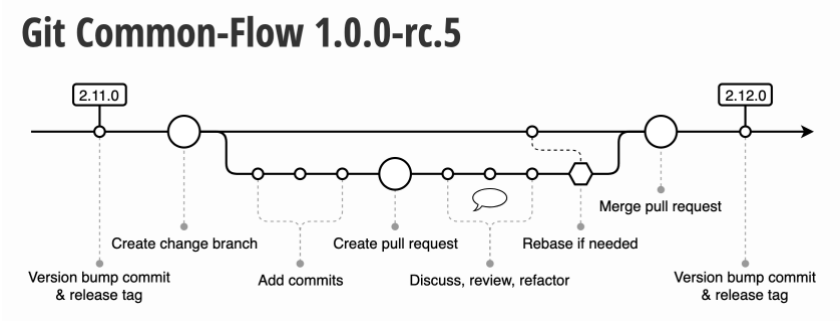

Abbildung 5 Common Flow von commonflow.org

Auf unser Projekt bezogen:

Bei Kleinen Anpassungen / Fixes
Bei kleineren Änderungen kann direkt auf dem Main gearbeitet werden

Bei einem Feature

1. Branch erstellen (Namen soll kurze Beschreibung vom Feature haben durch Bindestriche getrennt z.B. „user-authentification“ oder „connect-database“)
2. Änderung vornehmen und committen, bis Feature umgesetzt
3. Pull Request erstellen
4. Review mit Team Mitglied
5. (Rebase und) In den Main mergen

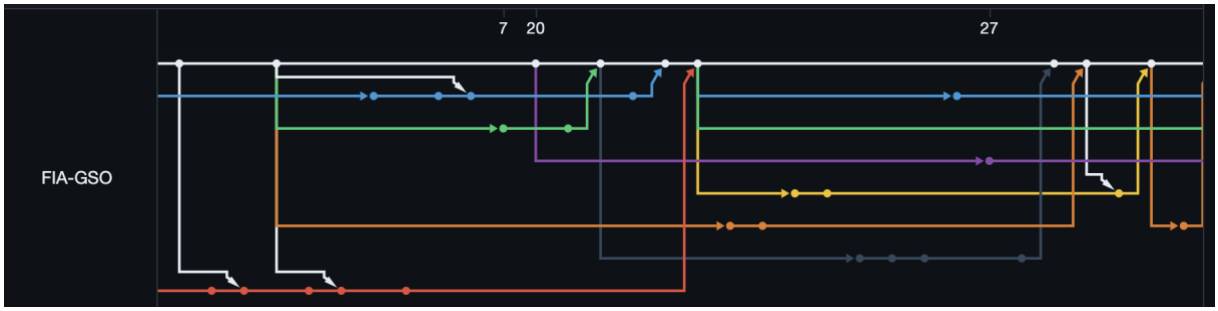

Abbildung 6 Unser Network Graph auf Github

In Abbildung 6 kann man gut sehen, wie wir uns an unseren Workflow gehalten haben. Die weiße Linie ist der Main Branch und man sieht, wie bei einem Feature immer vom diesem abgebranched wird (verschiedenfarbige Linien) und irgendwann wieder auf diesen zurückführt wird.