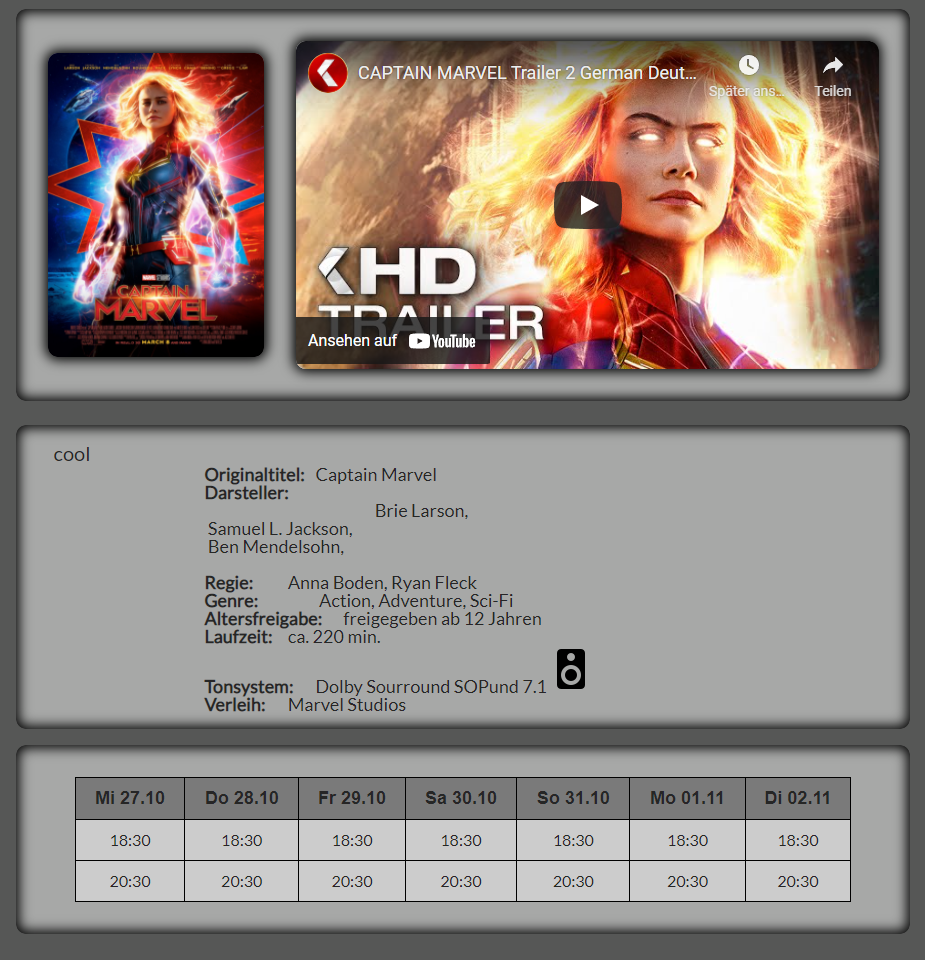
Ein Bild, das Text enthält.

Automatisch generierte Beschreibung



**Kino Webseite (aus Besuchersicht)**

**14.12.2021**

Lukas Haiber, Jakob Meyer

Inhaltsverzeichnis

[Projektauftrag 4](#_Toc90918767)

[Zusammenfassung 4](#_Toc90918768)

[Details 4](#_Toc90918769)

[Termin Vorgaben 4](#_Toc90918770)

[Projektstrukturplan 5](#_Toc90918771)

[Organisationschart 5](#_Toc90918772)

[Planung 6](#_Toc90918773)

[Zielsetzung 6](#_Toc90918774)

[Beispiele existierender Ansätze 6](#_Toc90918775)

[Ergebnisse 6](#_Toc90918776)

[Strukturlayout / Featureset 8](#_Toc90918777)

[Nutzen des Strukturplans 9](#_Toc90918778)

[Aufgabenverteilung 9](#_Toc90918779)

[Aufgabenblöcke 9](#_Toc90918780)

[Verteilung 9](#_Toc90918781)

[Durchführung 10](#_Toc90918782)

[Genutzte Technologien 10](#_Toc90918783)

[Repository 10](#_Toc90918784)

[Inhalte 10](#_Toc90918785)

[Verwendete Tags/Elemente (Beispiele) 11](#_Toc90918786)

[Design 12](#_Toc90918787)

[Color Scheme 12](#_Toc90918788)

[Flexbox 12](#_Toc90918789)

[Datenbank Anbindung 15](#_Toc90918790)

[Test-Server 16](#_Toc90918791)

[Entwicklungsumgebung 16](#_Toc90918792)

[Endpunktbeschreibung 16](#_Toc90918793)

[Startseite 16](#_Toc90918794)

[Detailseite 16](#_Toc90918795)

[Buchungsseite 16](#_Toc90918796)

[Admininterface 17](#_Toc90918797)

[Codeausschnitte 17](#_Toc90918798)

[Index 17](#_Toc90918799)

[PHP / HTML 17](#_Toc90918800)

[CSS 19](#_Toc90918801)

[Detail 20](#_Toc90918802)

[PHP / HTML 20](#_Toc90918803)

[CSS 21](#_Toc90918804)

[Contact 22](#_Toc90918805)

[CSS 22](#_Toc90918806)

[Aktivitäten 22](#_Toc90918807)

[Fazit 24](#_Toc90918808)

[Ergebnis 24](#_Toc90918809)

[Verbesserungsmöglichkeiten 24](#_Toc90918810)

[Quellen 25](#_Toc90918811)

# Projektauftrag

## Zusammenfassung

Das Projekt (HTML, CSS, PHP) soll in einer Zeitspanne von ca. 12-16 SW Stunden realisiert werden. Eine Erarbeitung auch außerhalb des Unterrichts wird angestrebt. Die Aufgabe umfasst das Projekt selbst, eine abschließende Präsentation mit Fragen sowie eine ca. 10 bis 15-seitge Ausarbeitung, die einen wissenschaftlichen Charakter haben soll. Die Struktur und der Aufbau obliegen den jeweiligen Gruppen.

## Details

Abgabe: Montag, 20.12.2021 bis 20:00

Projekt: Möglichst naturgetreue Abbildung/Realitätsnah (5-10 Seiten HTML/PHP programmiert, "statische" Erarbeitung --> keine CMS und Co, kein PHP-Framework, kein Bootstrap)

Präsentation: Dienstag, 11.01.2021, ca. 15 Minuten (+5-10 Minuten Fragen)

Bewertung: Wie eine KA(BK Note) und ggf. ergänzende PK-Note

Dokumentation: Sinnvolle Gliederung, inklusive wissenschaftlichen Charakter (Verzeichnisse, Quellen und Co), 10-15 Seiten Fließtext (ohne Titelseite, Anhänge oder Verzeichnisse gerechnet), Aufbau und Strukturierung obliegt den einzelnen Gruppen

Modellierungselemente: Grafisch in der Dokumentation, sinnvolle Methoden/Werkzeuge zur Darstellung

Vertiefungsmöglichkeit: Datenbankanbindung mit Daten, Funktionen, OOP in PHP

## Termin Vorgaben

30.11.2021 Labor, Erarbeitung in den Projektgruppen

01.12.2021 Labor, Erarbeitung in den Projektgruppen

14.12.2021 Labor, Letzter Termin zur Erarbeitung

20.12.2021 Abgabe von Dokumentation und Projekt (bis 20:00 Uhr)

11.01.2021 Halten der Präsentation

# Projektstrukturplan

Projektstrukturplan

# Organisationschart

# Planung

## Zielsetzung

Das Ziel unserer Gruppe für dieses Projekt ist es, eine Kino-Webseite zu erstellen. Sie soll die Ansicht des Kunden darstellen und diesem ermöglichen das Kino-Programm einzusehen, sowie Karten zu buchen. Des Weiteren soll zu jedem Film eine Detail-Ansicht aufrufbar sein, welche dem Kunden weitere Informationen wie Darsteller, Altersfreigabe und Laufzeit bieten kann. Während des Buchungsprozesses gibt der Kunde seine Daten an und wählt den Sitzplatz aus.

Für die gesamte Webseite gilt, dass die individuellen Daten per PHP von einer Datenbank geladen werden sollen. Dies jedoch nur statisch, da es sich nicht um ein reelles Projekt und somit um Testdaten handelt.

## Beispiele existierender Ansätze

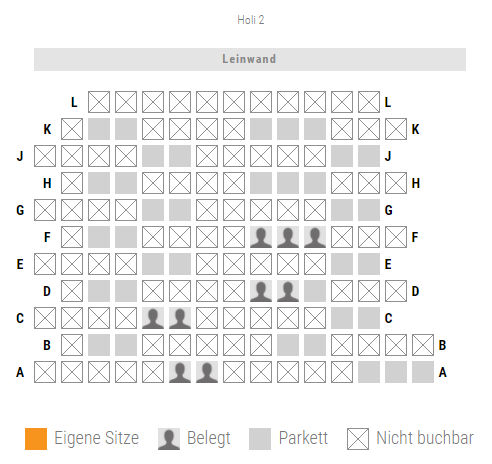
### Ergebnisse

Dank der Analyse hatten wir eine genauere Vorstellung der Abläufe auf einer typischen Kino-Webseite. Dadurch konnten wir besser Planen welche unter Seiten wir benötigen und welche Komponenten diese beinhalten sollen. Außerdem gaben uns die Beispiele Ideen für das Design und Layout unserer Webseite.

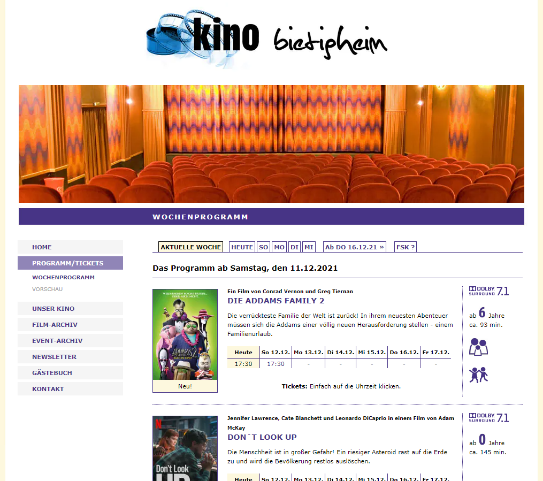
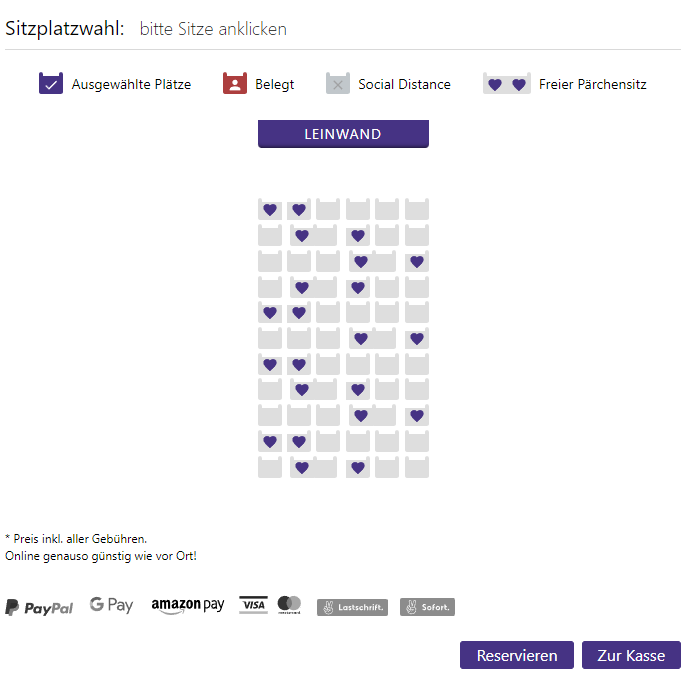
Somit haben wir uns zum Beispiel dazu entschieden den Kunden nicht direkt mit den Informationen zu den Filmen zu überhäufen, sondern erst nach Auswahl und der damit verbundenen Interessenbekundung. Des Weiteren haben wir beschlossen das Programm direkt auf der Startseite anzuzeigen, da dies der Fokus bei Kino Webseiten sein sollte.

Ein weiterer Bereich, bei dem uns das vorherige Betrachten von „Konkurrenz“ Webseiten geholfen hat, ist bei der Wahl der Interessanten Informationen wie beispielsweise auf der Detailseite die Daten zum Film.

Für die Buchungsseite haben wir uns für ein Abbild des reellen Kinosaals durch Buttons in Sitz form entschieden. Dies gibt dem Kunden eine klare Vorstellung davon, wie der Saal aufgebaut ist und dadurch, wo er sitzen möchte.

Ein Bild, das Text, drinnen enthält.

Automatisch generierte BeschreibungEin Bild, das Text, Screenshot, verschieden, mehrere enthält.

Automatisch generierte Beschreibung

Platzwahl Cinemaxx

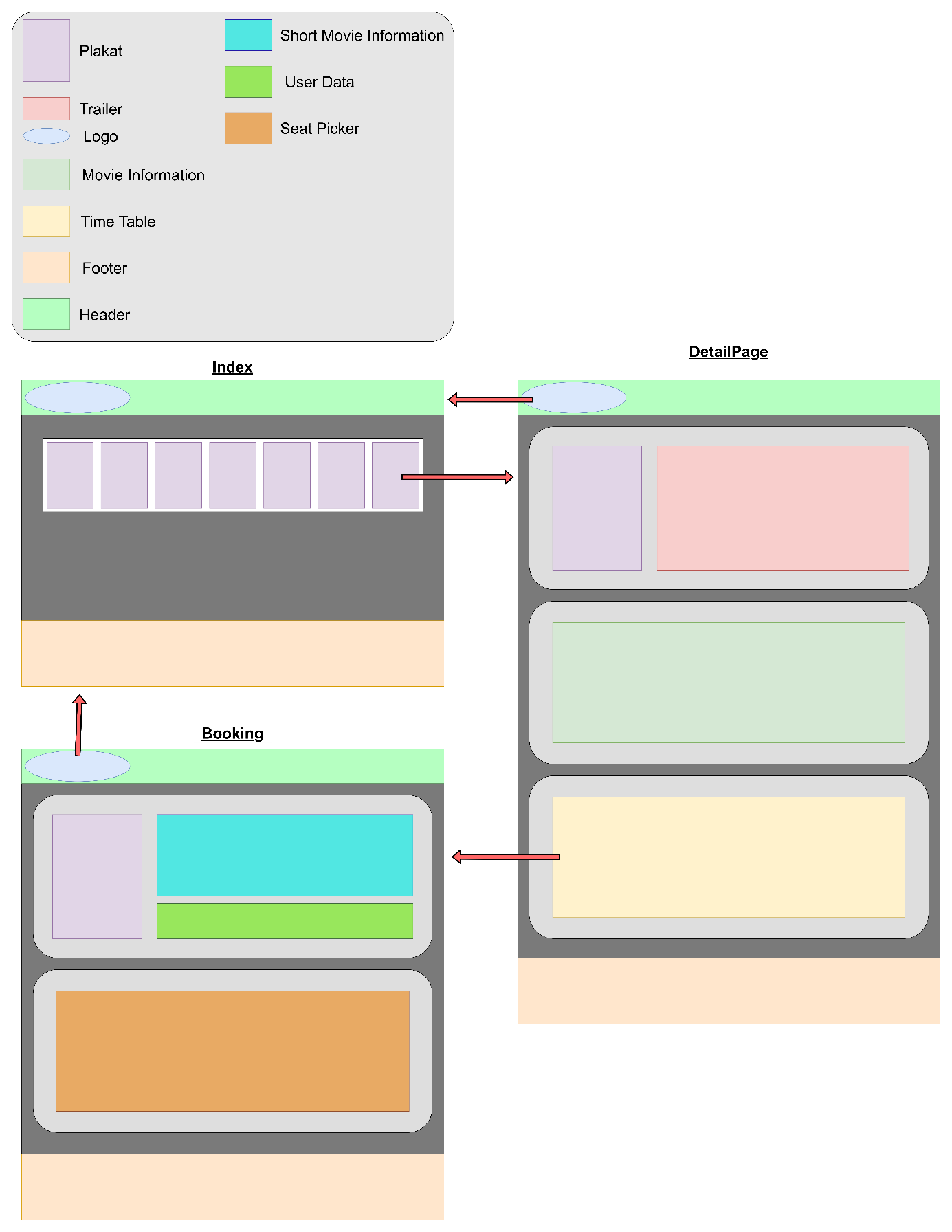
Detail Cinemaxx

Programm Cineplex

Platzwahl Kino Bietigheim

Programm Kino Bietigheim

## Strukturlayout / Featureset



### Nutzen des Strukturplans

Durch das Erstellen eines Strukturplans konnten wir schon im Voraus erarbeiten und festlegen wie unsere Webseite am Ende aufgebaut sein soll. Dies ist wichtig da man ohne Planung nicht strukturiert und effektiv an einem Projekt arbeiten kann. Außerdem kann für ein Teamprojekt die Aufgabenverteilung nur stattfinden, wenn vorher Schritte und Aufgaben definiert wurden.

## Aufgabenverteilung

### Aufgabenblöcke

Als es an die Aufgaben Verteilung ging haben wir erst einmal entsprechend dem Strukturplan das Projekt in die Seiten Index, Detail, Booking und Contact unterteilt. Zusätzlich haben wir dann noch größere Komponenten wie die Platzauswahl als extra Aufgabe definiert. Des Weiteren galt es zwischen Design und Funktionalität zu unterscheiden. Hier galt für uns, dass Design die HTML und CSS und Funktion die PHP und Datenbank umfasst. Zu guter Letzt gab es noch die Dokumentation und Präsentation. Das Festhalten des eigenen Fortschrittes haben wir nicht als eigenen Aufgabenblock definiert, da sich hier jeder selbstständig kümmern muss.

### Verteilung

Zuerst haben wir das Erstellen der einzelnen Seiten verteilt. Dadurch hat Lukas sich um Index und Booking gekümmert, während Jakob die Seiten Detail und Contact übernommen hat. Außerdem haben wir die Arbeit etwas nach Interessen verteilt, weshalb sich Lukas mehr um die Funktionalität also um Datenbank und PHP und Jakob um Design und Inhalt gekümmert hat. Des Weiteren hat Lukas den Seitenübergreifenden Header und Footer angelegt während Jakob die Platzauswahl umgesetzt hat. Im weiteren Verlauf des Projekts hat Jakob das Design der einzelnen Bestandteile der Webseite überarbeitet und Dynamik sowie Responsiveness hinzugefügt. Im gleichen Zuge hat Lukas die Daten aus der Datenbank in die Webseite eingelesen und angezeigt. Hierfür waren an manchen Stellen Änderungen am Aufbau nötig damit die PHP-Anbindung besser funktioniert. Im Ausgleich für die Datenbank hat Jakob größere Teile der Dokumentation übernommen. Insgesamt ist es uns auf diese Weise gelungen die Aufgaben gleichmäßig zu verteilen und somit gemeinsam den Gesamtarbeitsaufwand zu handhaben.

# Durchführung

## Genutzte Technologien

### Repository

Wir haben uns dazu entschieden **GitHub** für unser Projekt zu verwenden. Somit konnten wir unsere Daten sicher speichern ohne die Gefahr, dass sie aufgrund von fehlerhaften Endgeräten, Problemen der Datenträger oder anderweitigem Versagen verloren gehen. Des Weiteren konnten wir so unser Projekt ohne Probleme auf unsere Entwicklungsgeräte übertragen und aktualisieren. Der größte Vorteil, war jedoch die Versionskontrolle, durch die wir Zwischenstände erstellen und gleichzeitig am Projekt arbeiten konnten.



### Ein Bild, das Text, Schild enthält. Automatisch generierte BeschreibungInhalte

Um die Struktur, Verteilung und den Aufbau der Inhalte unserer Webseite zu definieren haben wir die Hypertext Markup Language (kurz **HTML**) verwendet. Wie der Name schon erraten lässt, handelt es sich hier nicht um eine Programmier-, sondern um eine Auszeichnungssprache.

Der Code dieser Sprache ist aus einer Reihe von Tags in der Form <öffnend> </schließend> aufgebaut. Inhalte wie Text stehen dann zwischen dem öffnenden und dem schließenden Tag. Falls es keinen Inhalt gibt, kann der schließende Tag weggelassen werden und der öffnende am Ende um einen / ergänzt werden.

Des Weiteren können diesen Tags Parameter mitgegeben werden. Dies geschieht in der Form „parameterName = wert“ und wird in den öffnenden Tag hinter dessen Namen geschrieben.

#### Verwendete Tags/Elemente (Beispiele)

##### <div>

* + Dies ist ein Container mit fließendem Inhalt und ohne genauere Bedeutung. Er wird oft zur Strukturierung und in Kombination mit CSS-Styles verwendet.

##### <h1>

* + H1 beziehungsweise H1 bis H6 sind Überschriftelemente. Wie schon gesagt gibt es die Level 1 bis 6 wobei 1 die höchste Gewichtung trägt und somit größer und deutlicher hervorgehoben wird als die nachfolgenden. Dank der Level ist es außerdem möglich automatisch ein Inhaltsverzeichnis zu generieren.

##### <img>

* + Mit diesem Tag lassen sich Grafiken in ein HTML-Dokument einbetten. Ein praktischer Parameter hierfür ist „alt“, da sich hiermit ein Alternativtext definieren lässt. Dieser wird nur dann angezeigt, wenn das Bild nicht geladen werden konnte. Außerdem ist es dieser Text, welcher vorgelesen wird, sollte ein Benutzer gebrauch der Barrierefreiheit machen (wie zum Beispiel Blinde). Dies ist mit einer der Gründe, weshalb rein dekorative Grafiken mit einem leeren „alt“ versehen werden, da diese dann beim Vorlesen übersprungen werden.

##### <iframe>

* + Der Inlineframe wird dazu verwendet eine weitere HTML-Seite in die aktuelle einzubetten. Mit dem Parameter „src“ kann die URL dieser Webseite definiert werden. In unserem Projekt verwenden wir iframe dazu um ein YouTube Video als Trailer anzuzeigen. Dank der Einbettung kann dieses direkt auf unserer Webseite abgespielt werden und ist kein Link zu YouTube.

##### <i>

* + Der i Tag soll ein Wort wie zum Beispiel einen Fachbegriff vom restlichen Text abgrenzen. Dies geschieht dadurch, dass der vom i Tag umschlossene Inhalt Kursiv dargestellt wird.

##### <b>

* + Dieser Tag hebt einen Text hervor, indem dieser Fett geschrieben wird. Er hat ansonsten keine weiteren Informationen wie es bei <strong> oder <em> der Fall wäre.

##### <p>

* + Der Paragraf (kurz p) stellt einen Absatz und somit einen zusammenhängenden Abschnitt eines Textes dar. Da er Hauptsächlich eine gruppierende Funktion hat, kann er auch für die Gruppierung anderer Elemente als Text, wie zum Beispiel Bilder verwendet werden.

##### <table>

* + Mit dem Table Tag können meistens zwei dimensionale Tabellen erstellt werden. Innerhalb können mit <thead>, <tfoot>, und <tbody> der erste, letzte und alle anderen Abschnitte angelegt werden. Jeder dieser Abschnitte kann Zeilen enthalten welche mit <tr> für Table Row definiert werden. Danach kann pro Zeile noch eine oder mehrere Spalten mit <td> definiert werden.

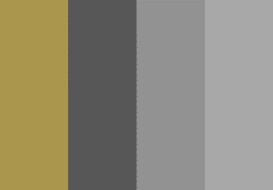
### Design



Das Cascading Style Sheet (kurz CSS) wird dafür verwendet, eine Webseite zu gestalten. Mit ihm können sämtliche Komponenten und Tags von HTML formatiert werden. Des weiteren können im HTML die Parameter is und class dazu verwendet werden zusätzliche Komponenten zu definieren. Somit kann beispielsweise eine einzelne Überschrift rot gefärbt werden indem nicht der <h1> sondern der „.überschrift“ Identifizierer verwendet wird.

#### Color Scheme

##### Base



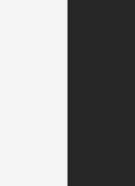
#AA964C Accent

#565756 Main Background

#929292 Element Hover

#A7A8A7 Element

##### Text



#F5F5F5 Main

#262626 Element

#### Flexbox

Wenn es darum geht die Elemente einer Webseite dynamisch zu positionieren und entsprechend des gegebenen Raumes zu skalieren ist die Flexbox eine der besten Optionen. Bei der Arbeit mit Flexboxen ist es wichtig, zwischen Container und Items zu unterscheiden. Der Container wird per „display: flex“ im CSS festgelegt. Er definiert den verfügbaren Raum, in dem die Items transformiert werden. Außerdem können im Container die folgenden Eigenschaften definiert werden, welche Auswirkungen auf Position, Anordnung und Größe haben.

* flex-direction
  + Diese Eigenschaft definiert die Richtung, in der die Items angeordnet werden. Die möglichen Werte sind „column“ (oben nach unten) und „row“ (links nach rechts). Zusätzlich gibt es diese jeweils als „reverse-“ also invertiert.
* flex-wrap
  + Hier wird mit „wrap“ oder „nowrap“ definiert, wie sich die Items im Bezug auf den verfügbaren Raum verhalten sollen. Standardmäßig ist nowrap aktiv was bedeutet, dass die Items selbst skalieren und so beispielsweise schmaler werden, wenn nicht genug platz im Container ist. Wrap hingegen bedeutet, dass Items automatisch in einer neuen Zeile landen und so nicht kleiner werden, wenn in einer Zeile zu wenig Platz ist.
* flex-flow
  + Der flex-flow ist nichts anderes als eine Arbeitsersparnis, da sich mit ihm die flex-direction und flex-wrap Eigenschaften auf einmal setzen lassen.
* justify-content
  + Mit dieser Eigenschaft werden die Items als Gruppe auf horizontaler Ebene positioniert. Die Werte „flex-start“, „flex-end“ und „center“ sind an sich selbsterklärend und halten den Mindestabstand zwischen den Items ein. Danach gibt es noch „space-between“ für gleichmäßige Verteilung über den gesamten Bereich, „space-around“ für gleich große Abstände auf beiden Seiten der Items (Lücken zwischen zwei Items sind dadurch zwei Abstände breit) und zu guter letzt „space-evenly“ was dafür sorgt, dass alle Abstände gleich sind.
* align-items
  + Diese Eigenschaft dient ebenfalls der Positionierung, jedoch diesmal auf vertikaler Ebene. Wie bei justify-content gibt es hier auch die Werte „flex-start“, „flex-end“ und „center“. Wichtig zu verstehen ist hierbei jedoch, dass die Items jeweils bestmöglich positioniert werden. Das bedeutet bei start und end jeweils so weit wie möglich an die entsprechende Kante des Containers und bei center auf die horizontale Mittellinie. Der Wert „baseline“ kann ähnlich aussehen wie center, orientiert sich jedoch nicht am kompletten Item, sondern an der Baseline also der gedachten Linie, auf der der Text innerhalb der Items steht. Als letztes gibt es „stretch“, welcher ganz einfach die Item Höhe auf die des gegebenen Raums im Container setzt.
* align-content
  + Die align-content Eigenschaft ermöglicht es die Zeilen von Items in einem Container vertikal zu positionieren. Es gibt wieder die Werte „flex-start“, „flex-end“, „center“, „stretch“, „space-between“ und „space-around“. Wichtig ist hier, dass der Effekt nur eintritt wenn es sich um einen multi-line Container mit gesetztem „flex-wrap“ handelt.
* gap
  + Der „gap“ definiert den Mindestabstand zwischen den Items. Es kann entweder ein Wert für alle oder einer für horizontal und einer für vertikal gesetzt werden. Außerdem gibt es noch die Varianten des „row-gap“ und „column-gap“, welche diese beiden Werte noch einmal in einzelne Werte unterteilen.

Auch für die Items im flex Container gibt es eine Reihe von Eigenschaften, die folgende Effekte haben.

* order
  + Mit der Eigenschaft „order“ kann unabhängig von der Reihenfolge, in der die Items im HTML-Code angelegt worden sind eine Ordnung innerhalb des Containers hergestellt werden.
* flex-grow
  + Diese Eigenschaft gibt an, dass und wie schnell ein Item wächst um den Container zu füllen. Wenn zum Beispiel Item1 einen grow Wert von 1 und Item2 einen von 4 hat, so wird Item2 am ende vier mal so viel Platz einnehmen wie Item1. Wichtig ist hier, das es sich nicht um feste Breiten oder Höhenangaben handelt, sondern um ein Verhältnis. Der Wert 0 bedeutet das ein Item nicht wächst.
* flex-shrink
  + Dies ist im Prinzip das Gegenteil von „flex-grow“ und definiert das Verhältnis wenn zu wenig Platz im Container für die Items ist, wodurch diese schrumpfen müssen.
* flex-basis
  + Mit der „flex-basis“ kann die Basisgröße eines Items definiert werden. Es wird immer versuchen dies einzuhalten und somit nicht weiterwachsen. Ist jedoch zu wenig Platz vorhanden, so wird es ganz normal schrumpfen.
* Flex
  + Dies ist wieder eine Arbeitsersparnis, da mit ihr grow, shrink und basis in einem definiert werden können.
* align-self
  + Die „align-self“ Eigenschaft überschreibt den align-items Wert des Containers für das einzelne Item.

### Datenbank Anbindung

Zur Umsetzung unserer Datenbankanbindung haben wir uns für die in der Webentwicklung weit verbreitete Skriptsprache PHP entschieden. PHP stand ursprünglich für „Personal Home Page (Tools)“ und dient als Hypertext Preprocessor. Mit seiner an C und Perl angelehnten Syntax wird PHP dazu verwendet, dynamische Webseiten zu erstellen. Dies geht unteranderem dadurch, dass in ihren Code HTML eingebettet und somit Webelemente per PHP angelegt werden können. Dies haben wir zum Beispiel für die Film Plakate auf der Index Seite verwendet. Außerdem bietet sie eine umfangreiche Datenbankunterstützung, sowie Erweiterbarkeit durch zahlreiche Funktionsbibliotheken.

Um PHP innerhalb eines HTML Codes zu verwenden, muss lediglich der <?php INHALT ?> Tag verwendet werden. Mit dem echo Befehl können dann Inhalte wieder zurück in HTML gegeben werden. Auf diese Wiese konnten wir beispielsweise Werte aus unserer Datenbank wie Film Name auf der Webseite ausgeben.

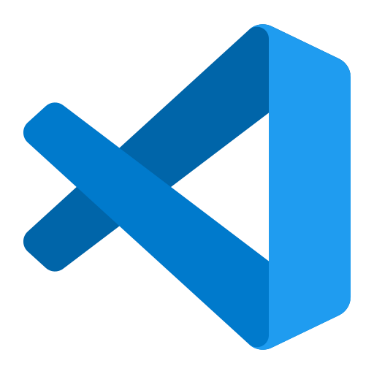
Ein weiterer Vorteil von PHP war für uns die in XAMPP integrierte PHP-Funktionalität für MySQL Datenbanken, wodurch wir aus der Kombination der unterschiedlichen Technologien eine gute Umsetzung eines simulierten Webservers bauen konnten.

### Test-Server



Da wir uns dazu entschieden haben PHP für die Einbindung der Daten aus unserer Datenbank zu verwenden, lag es nahe XAMPP (nach eigener Aussage die beliebteste PHP-Entwicklungsumgebung) zu verwenden. Mit diesem Tool war es uns möglich einen Lokalen Apache Testserver zu starten und auf diesem unsere Webseite zu testen. Ein weiterer Pluspunkt für XAMPP ist, dass es Betriebssystem übergreifend kostenlos zur Verfügung steht und somit ideal für private Projekte ist.

### Entwicklungsumgebung



Für die Entwicklung selbst haben wir die Entwicklungsumgebung Visual Studio Code verwendet. Grund dafür ist einerseits das es diese für jedes Betriebssystem sowohl als App als auch als Webseite kostenlos gibt. Des Weiteren kann diese dank unzähligen Plugins und Modulen kompatibel mit jeglicher Programmiersprache gemacht werden. Außerdem und dadurch hat sie ein sehr gutes Highlighting und eine gute Autocompletion was die Arbeit grundsätzlich erleichtert.

## Endpunktbeschreibung

Startseite (<http://localhost/index.php>)

Die Startseite zeigt die Übersicht der aktuell laufenden Filme, außerdem ist eine kurze Auflistung der Schritte zum Reservieren von Tickets angezeigt.

Detailseite (<http://localhost/details.php?id=1>)

Die Detailseite enthält das Filmplakat, einen Trailer, eine kurze Filmbeschreibung, Daten und Fakten zum Film, wie Darsteller, Regie, Genre. Der Letzte Abschnitt zeigt eine Tabelle mit den Vorstellungen des jeweiligen Films, mit einem Klick auf eine der Uhrzeiten gelangt man zur Buchungsseite.

Buchungsseite (<http://localhost/booking.php?id=1&date=27-10&time=18-30>)

Bereits gebuchte Sitze werden rot dargestellt, noch zu buchbare Sitze sind in Weiß dargestellt. Wurde ein, oder mehrere Sitze ausgewählt, kann nach ausfüllen der Formular Felder ein Sitz gebucht werden.

Admininterface (<http://localhost/admin/admin.php>)

Das Admininterface ermöglicht es Buchungen zu löschen. Außerdem können Filme angelegt oder gelöscht werden.

## Codeausschnitte

### Index

#### PHP / HTML

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Der Head unserer Index.php übernimmt wie auch bei den restlichen Seiten unseres Projekts, die Aufgabe, das grundlegende Setup einzurichten. Neben den Metadaten, welche dem Browser grundlegende Informationen bieten, werden im Head auch noch die Stylesheet-Links angegeben. Dies ist nötig. Damit das in den CSS-Dateien definierte Styling auch auf die Webseite angewandt wird. Grundsätzlich muss das Styling nicht immer in extra Dateien ausgelagert werden, da dieses auch als Style-Parameter in die HTML-Tags geschrieben werden kann. Da wir jedoch für unsere Webseite viel CSS Styling verwenden ist es allein schon wegen der Übersichtlichkeit sinnvoll dies auszulagern. In unserem Projekt verwenden wir außerdem meistens mehrere Stylesheets, da beispielsweise die „globalen“ Variablen für das Design somit nicht in jedem einzelnen angegeben werden müssen. Außerdem verwenden wir noch ein Stylesheet, welches aus dem Internet geladen wird und Schriftarten enthallt. Zu guter Letzt setzen wir im Head noch den Titel und ein Icon, welche auf dem geöffneten Tab angezeigt werden.

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Im ersten PHP tag jeder unserer Seiten werden Variablen und die Verbindung zu unserer Datenbank hergestellt. Da wir auf der Index Seite die laufenden Filme und dafür deren Namen, Poster und Altersfreigabe anzeigen, laden wir dafür die Tabelle „movieTable“ aus der Datenbank.

Für den Body unserer Seiten gilt, er beginnt immer mit unserem Header und endet mit dem Footer und den Skripten, welche für den Import nötig sind. Zwischen diesen kommt der eigentliche Inhalt der jeweiligen Seite, welcher bei uns mittels des div Tags in Abschnitte unterteilt ist. Dies hilft einerseits der logischen Strukturierung und des Überblicks im HTML-Code, andererseits ermöglicht dies detaillierte Style Anpassungen mittels der Classen im CSS umzusetzen.

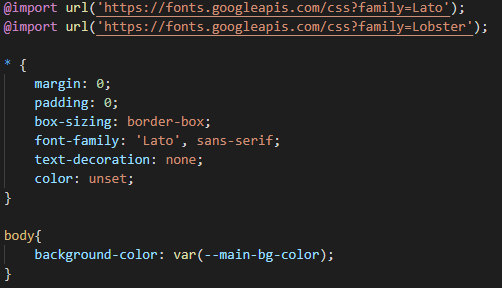
Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

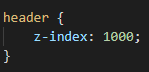
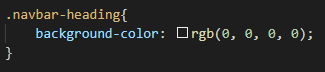
Es ist möglich in einem PHP-Tag HTML Code zu erstellen und diesen dann über den echo Befehl auszugeben, wodurch dieser dann auf der Webseite erscheint. Diese Funktion verwenden wir beispielsweise um passend zur liste der laufenden Filme deren Poster und Daten anzuzeigen. Nach ein paar Abfragen, um sicherzugehen, dass überhaupt daten aus der Datenbank ausgelesen werden, geben wir mittels einer For-Schleife zu jedem Eintrag den nötigen HTML Code zurück. Zum Schluss muss nur noch die nun nicht mehr benötigte Verbindung zur Datenbank getrennt werden.

#### CSS

Unser Main Styling und das für die Index Seite ist folgendermaßen aufgebaut.



Zu beginn werden verwendete Schriftarten importiert und danach im, die komplette Webseite betreffenden, \* Tag mit weiteren Parametern angewandt. Danach wird noch dem Design-Scheme entsprechend die Hintergrund Farbe gesetzt. Die Farbe wird aus einer variable gelesen, welche in einer weiteren CSS-Datei mit allen variablen für das Design steht und somit in allen Designsheets wieder verwendet werden kann. Außerdem erlaubt dieser Ansatz, die nachträgliche Änderung des Color Schemes über einen einzelnen Wert.

Ein Bild, das Text enthält.

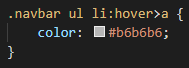
Automatisch generierte BeschreibungDank der im HTML-Code definierten Classen und IDs können für die einzelnen Bestandteile jeweils die Style Parameter gesetzt werden.

Hierbei gibt es drei Arten, „#name“ für IDs, „.name“ für Classen und „name“ für die HTML Elemente an sich. Durch Kombinationen dieser und weiterer Keywords kann das Design spezifisch umgesetzt werden.

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Hier wird zum Beispiel allen Listenelementen „li“ jeder unsortierten Liste „ul“ welche in der Classe „.navbar“ liegen die Textfarbe für Elemente zugewiesen.



Ein weiteres, von uns viel verwendetes Keyword ist „hover“. Diese werden mittels : an eine Auswahl angehängt. Das Styling des hover Effekts wird angewandt, wenn der Benutzer mit der Maus über das Element fährt. Wenn es nötig ist innerhalb eines solchen Effekts ein bestimmtes Element anzusprechen so kann dies mit > angehängt werden. Dank dieser Funktionen ändert sich in diesem Beispiel die Schriftfarbe aller A-Tags, sobald der Mauszeiger über dem Listenelement ist.

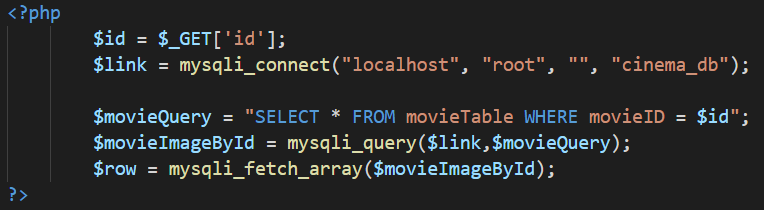
### Detail

#### PHP / HTML

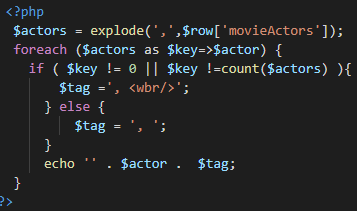
Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Mit dem Iframe-Tag konnten wir ein YouTube Video mit dem Trailer des Films in unsere Webseite einbauen. Um die Detail Seite dynamisch auf den ausgewählten Film abzustimmen, wird die eigentliche URL aus der Datenbank geladen.



Dies ist der Code, in dem die Datenbankanbindung und die dafür nötigen Variablen angelegt werden.



Bei den Informationen zum Film konnte bei den meisten der Wert direkt ausgegeben werden, da es jedoch mehrere Darsteller gibt, mussten wir hier eine Logik erstellen, um diese für die Ausgabe zu trennen. Dafür haben wir über die enthaltenen Filmdarsteller iteriert.



Danach folgt noch die Tabelle mit den Laufzeiten des Films, welche den Benutzer dann an die Buchungsseite weiterleiten. Wichtig war hier die „movieID“, sowie Datum und Uhrzeit mitzugeben, damit die Buchungsseite auch richtig Buchen kann.

#### CSS

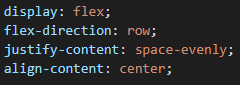
Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Um die einzelnen Abschnitte der Detailseite genauer zu betonen haben wir diese mittels Schatten mit einer Art 3d Effekt versehen. Hierfür muss zunächst der Rahmen festgelegt werden, da dieser gleichzeitig die Kontur des Elements bestimmt. Beim Box-Shadow gibt es einige Optionen, um zu bestimmen, wie hart und weit der Schatten sein soll. Außerdem kann mit Inset bestimmt werden, ob der Schatten vom Rahmen aus nach außen oder innen verlaufen soll. Dadurch konnten wir es so wirken lassen als wären die einzelnen Abschnitte Vertiefungen aus denen Elemente wie die Zeittabelle wieder herausstehen.

### Contact

#### CSS



Auf der Contact Seite haben wir viel mit Flex-Boxen gearbeitet, da mit ihnen Objekte dynamisch positioniert werden können. Sie skalieren und positionieren sich somit automatisch entsprechend des verfügbaren Platzes. Dies hat viele Vorteile, da einerseits nicht jedes Anzeigegerät dieselbe Größe hat und andererseits so besser eine UI mit dynamisch geladenen Inhalten umgesetzt werden kann.

## Aktivitäten

|  |  |  |
| --- | --- | --- |
| 26.10.21 | Beide | Uns wurde die Aufgabe gegeben eine Kino-Webseite aus Benutzer Sicht zu erstellen.  Wir haben das Git-Repository aufgesetzt und jeweils eine lokale Kopie unserer Branches erstellt. Des Weiteren haben wir uns Gedanken über Funktion und Aufbau unserer Webseite gemacht. Nach Betrachtung von Beispielen konnten wir unser Grund Layout und Feature Set definieren. |
| Lukas | Lukas hat die Index.php angelegt und mit Informationen gefüllt. Außerdem hat er das basisdesign umgesetzt für die Index Page umgesetzt. Des Weiteren hat er einen Header und Footer erstellt und in der Startseite eingebunden. Danach hat er zum Test, Daten in einem Array angelegt. |
| Jakob | Jakob hat die Detail Page angelegt und mit Beispiel Informationen gefüllt. Zu diesen gehören unter anderem Titel, Darsteller Altersfreigabe und Laufzeit. Des Weiteren hat er ein Plakat und Trailer eingebunden. Danach hat er das Basis Design erstellt und dafür das Color Scheme definiert. |
| 30.10.21 | Lukas | Lukas hat die Datenbank angelegt und per PHP an die Webseite angebunden. Dann hat er den Header und Footer noch mal überarbeitet. Außerdem hat er die Detail Page zu PHP umgewandelt damit die Daten der Datenbank eingebunden werden können. |
| Jakob | Jakob hat weitere Änderungen am Design vorgenommen. Danach hat er durch die Verwendung von Schatten den Elementen einen 3D Effekt verliehen. Zu guter Letzt hat er Style Variablen angelegt damit diese konsistent im restlichen Stylesheet angewandt werden können. |
| 16.11.21 | Lukas | Lukas hat dafür gesorgt, dass auf der Detail Seite Darsteller und weitere Informationen dynamisch aus der Datenbank geladen werden. Außerdem hat er die Buchungsseite erstellt und angefangen diese einzurichten. Des Weiteren hat er eine Admin Seite angelegt, welche zum Löschen von Filmen und Buchungen verwendet werden kann. Für die Buchungsseite hat er die Spielplan PHP-Parameter angelegt. |
| Jakob | Jakob hat die Styles der Detailpage so angepasst, dass die Elemente dynamisch skalieren und sich so an die aktuelle Größe des Browserfensters anpassen. Außerdem hat er dafür gesorgt, dass die Website responsive ist und dem Benutzer verdeutlicht, wenn er mit ihr interagiert. Danach hat er noch die Time Table mit den buchbaren Film Zeiten auf der Detailview eingefügt. |
| 20.11.21 | Jakob | Jakob hat die Sitzauswahl für die Buchungsseite angelegt |
| 30.11.21 | Jakob | Jakob hat die Styles des Headers an das zuvor erstellte Designschema angepasst. außerdem hat er dafür gesorgt, dass sich die Sitzauswahl automatisch zentriert und in manchen Punkten dynamisch skaliert. |
| 01.12.21 | Lukas | Lukas hat die Sitzauswahl in die Buchungsseite eingebunden. |
|  | Jakob | Jakob hat eine Kontaktpage angelegt und diese mit Beispieldaten gefüllt. Danach hat er dafür gesorgt, dass sie dynamisch skaliert. zu guter Letzt hat er ein Kontaktformular erstellt. |
| 11.12.21 | Lukas | Lukas hat weitere Anpassungen an der Buchungsseite vorgenommen. |
| Jakob | Jakob hat sich um die Ausarbeitung der Dokumentation gekümmert und kleinere Korrekturen an der Webseite vorgenommen. |
| 14.12.21 | Beide | Wir haben unser Projekt fertiggestellt und zusammengeführt. Vor 18:00 Uhr haben wir unser Projekt selbst und dessen Dokumentation abgegeben. |
| 15.12.21 | Beide | Wir haben unsere Präsentation gehalten und damit diese Projektarbeit abgeschlossen. |

# Fazit

## Ergebnis

Insgesamt sind wir mit dem Projekt zufrieden. Wir haben unser Ziel beziehungsweise unsere Vorstellung davon wie unsere Webseite am Ende aussehen soll gut umgesetzt bekommen und haben alle wichtigen Inhalte eingebaut. Natürlich gibt es auch einige Stellen, an denen wir noch länger arbeiten und diese verbessern könnten, mit Blick auf den gegebenen Zeitraum haben wir jedoch einen guten Endstand abliefern können. Die Arbeit am Projekt hat uns ermöglicht einen guten und in bestimmten Bereichen auch tiefen Einblick in HTML, CSS und PHP zu erlangen. Damit konnten wir wichtige Grundkenntnisse der Webentwicklung sammeln, welche auch in modernen Frameworks wie Angular und React immer nötig sind.

## Verbesserungsmöglichkeiten

Wenn wir nochmal ein solches Projekt umsetzen, werden wir zu Beginn gemeinsam Dinge wie das Design festlegen und auch gleich von Anfang an mittels Variablen, damit dieses direkt überall richtig angewandt wird. Dadurch fängt die „richtige“ Arbeit am Projekt zwar erst später an, es wird dadurch jedoch auch viel Aufwand erspart bleiben. Des Weiteren werden wir in Zukunft während der Arbeit am Projekt mehr Zeit in Aufschriebe, Dokumentation und Projekttagebücher stecken, damit Dokumentation und Präsentation am Ende weniger Aufwand werden. Für den Aufbau der Webseite werden wir uns von nun an im Vorhinein mehr Gedanken zur Datenbank zu machen, da hierdurch, später nötige Änderungen für die Umsetzung der Datenanbindung, im Vorhinein vermieden werden können.

# Quellen

* <https://www.kino-bietigheim.de/programm>
* <https://www.cinemaxx.de/film/ammonite>
* <https://www.cineplex.de/neckarsulm/>
* <https://www.toptal.com/designers/colourcode/>
* <https://www.freeiconspng.com/downloadimg/15142>
* <https://github.com/logos>
* <https://freeiconshop.com/icon/html-icon-flat/#google_vignette>
* <https://freeiconshop.com/icon/css-icon-flat/>
* <https://developer.mozilla.org/de/docs/Learn/Getting_started_with_the_web/HTML_basics>
* <https://www.w3schools.com/html/>
* <https://www.w3schools.com/css/>
* <https://www.w3schools.com/php/>
* <https://developer.mozilla.org/de/docs/Web/CSS/box-shadow>
* <https://css-tricks.com/snippets/css/a-guide-to-flexbox/>
* <https://developer.mozilla.org/de/docs/Web/CSS/CSS_Flexible_Box_Layout/Basic_Concepts_of_Flexbox>
* <https://www.w3schools.com/css/css3_flexbox.asp>
* <https://icon-icons.com/de/download/94513/PNG/512/>
* <https://icon-icons.com/de/download/130084/PNG/512/>
* <https://icon-icons.com/de/download/130266/PNG/512/>