BACHELORPROJEKT 2019

FLOW LINK

ROBIN BITTLINGER
SEBASTIAN SCHUCHMANN

HALLO!

Bevor es mit dem Lesen losgeht, würden wir Dir gerne einen Rat geben. Damit Du beim Lesen auch stets hoch konzentriert bleibst, solltest Du Dir **FlowLink** herunterladen. Auf was wartest Du?

Also Herunterladen, Aufgabe eintragen, Timer stellen, Kopfhörer auf und los gehts!

Falls Du die Dokumentation digital liest, klicke einfach hier! Wenn Du analog liest dann scanne entweder den QR-Code unten oder tippe die URL manuell ein.

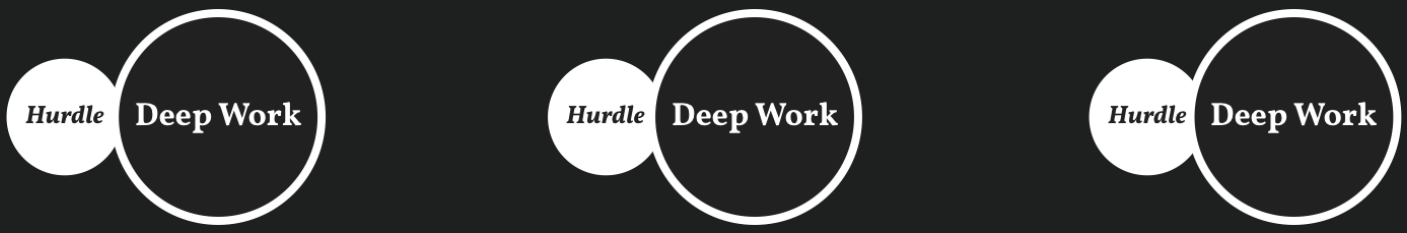
Die URL (in kurz): https://bit.ly/2Yjzf0D

FlowLink FlowLink FlowLink FlowLink FlowLink FlowLink FlowLink FlowLink FlowLink FlowLink FlowLink FlowLink FlowLink FlowLink FlowLink FlowLink FlowLink FlowLink FlowLink FlowLink

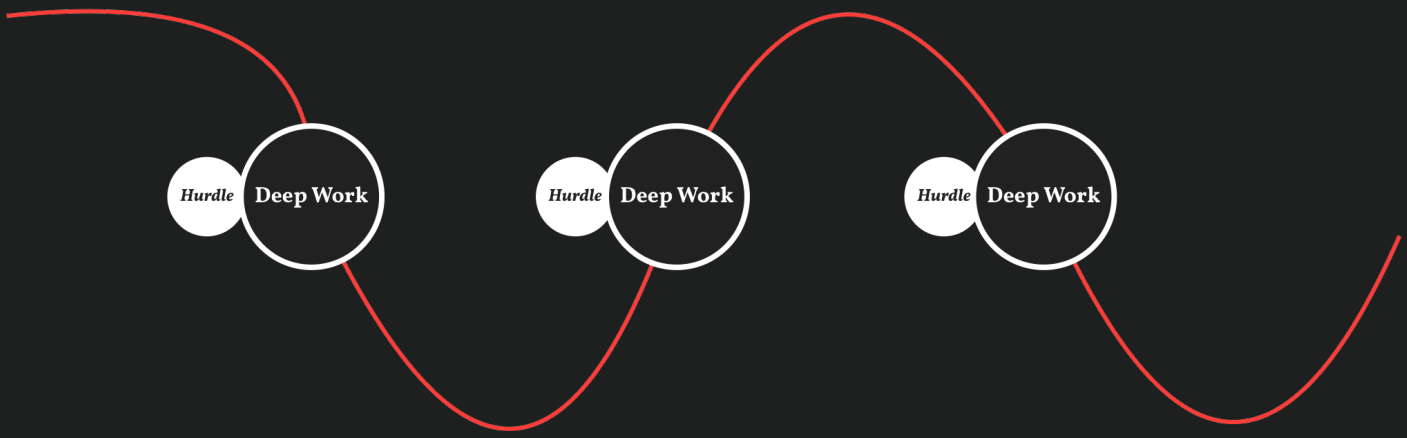
TL;DR

FlowLink is a tool that seamlessly integrates into the chaotic schedule of Knowledge Workers. By linking the context of separate flow phases during the workday, it eliminates the hurdle to enter into a flow state.

The target group of FlowLink consists of Knowledge Workers and more concretely scientists. Being prosperous in the 21. Century means doing work that requires your undivided attention. This type of work is termed Deep-Work. The Scientists we interviewed all agreed that Deep-Work is exciting but it also requires being “in the zone”. This perception of being “in the zone” is usually called flow and many would characterize it as the ideal working state. Unfortunately, getting into the zone requires overcoming an obstacle, which can lead to procrastination. That’s what we want to tackle.

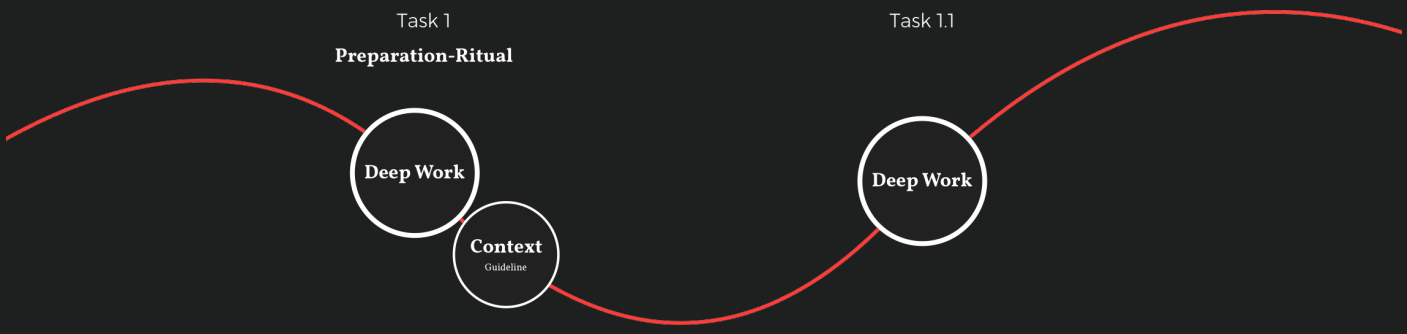


Overcoming this obstacle is a challenging and often time-consuming process. For instance, if the plan is to write an article, a respondent would describe how he would stroll around the room for 30 minutes before typing a single word into the computer. To overcome that, we developed the idea of the red thread.

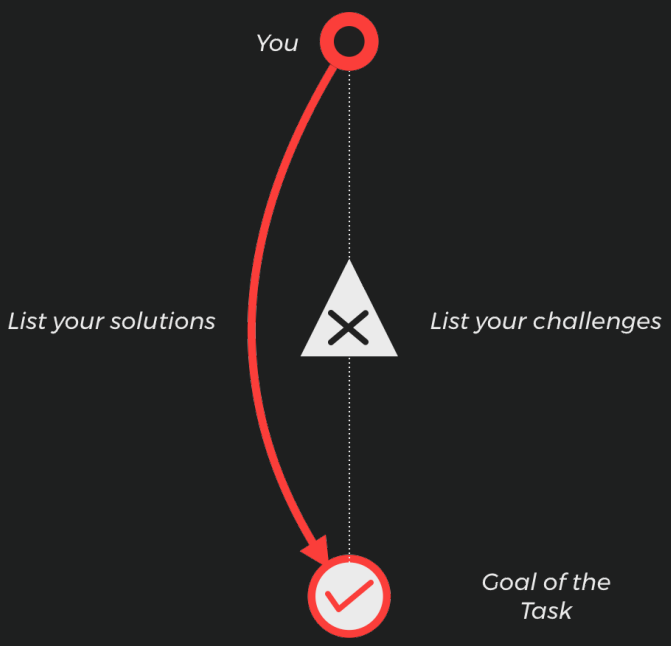


FLOW LINK

The idea is to link the Deep-Work phases of the same or similar tasks and thereby eliminate the hurdle to entry. This means, when a Deep-Work session is completed the context gets preserved. Then when you continue the task, you are right there where you left off. The song picks up where it ended the last time and you can continue your task as you have never even left in the first place.



The essential component that makes this process work is the Draft.
To complete the Draft at the end of a Deep-Work session,
you have to answer a few questions to conserve the state of mind you are in.



When finishing a task, the user is asked to state his upcoming goal, as well as the associated challenges and any potential solutions to those challenges.

This serves as a guideline when continuing the task. You quickly know where you left off and what you have to do to finish the task. We included aspects of the Hero's Journey to create the questions in the Draft used to build the Guideline.

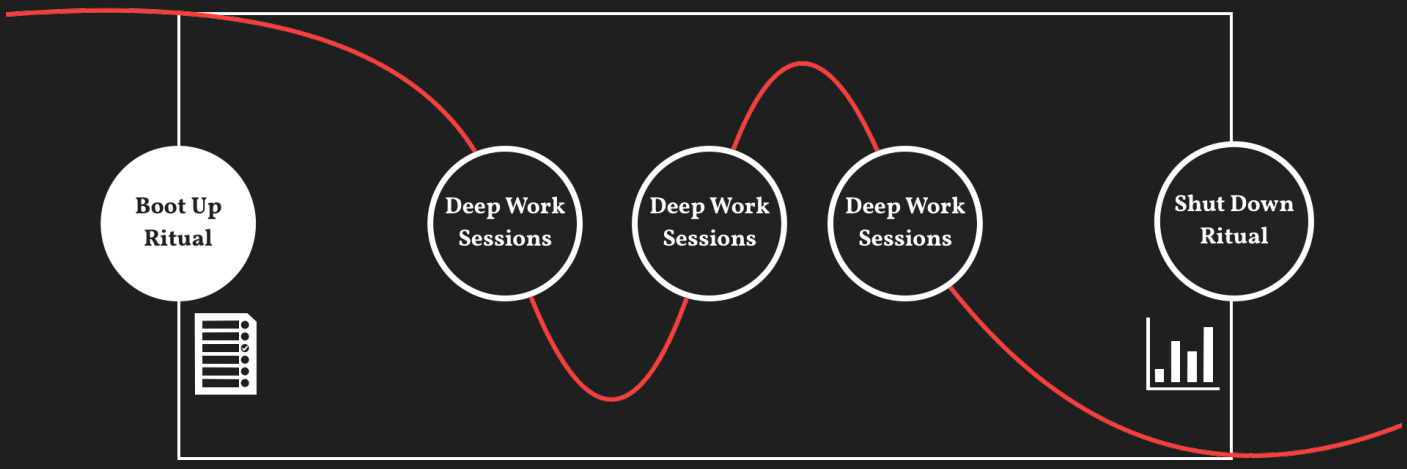
Every challenging task can be regarded as a dragon you have to fight, the main goal in sight.

To aid the concept of the Red Thread, additional structure is required. The Boot-Up and Shut-Down ritual provide this supporting structure. The system guides the user to schedule the working days for the coming week:

When is the Boot-Up and when is the Shut-Down ritual?

How much Deep-Work would you like to do?
In the Shut-Down ritual, you clarify for yourself that the workday has ended and optionally tweak the plan for tomorrow.

In the Boot-Up ritual, you declare the start of your workday and are presented with your outline for the day. Additionally, questions to promote self-reflection are asked in the Shut-Down ritual to further optimize your work.



Coming back to our original statement

Flow Link maximizes the Deep-Work necessary for a successful and well-lived working life in the 21th Century.

We, therefore, hope to provide a tool for those interested in pursuing the road of meaningful work like scientists.

FLOW
LINK

INHALTSVERZEICHNIS

INTRO 10

Die Menschen hinter FlowLink 10
Formalien 10
Der Anfang 11
Inconvenient Design 11

RESEARCH 14

Der Weg zur Zielgruppe 14
Die Interviews: Methode 14
Die Zielgruppe: Wissenschaftler 16
Die Interviews: Findings 16
User Journey 16
Was motiviert unsere Zielgruppe bedeutsame Arbeit leisten zu wollen? 18
Was findet sich direkt in der Anwendung wieder? 18
Persona 19
Das Problem 20

NACHFORSCHUNG 22

Arbeit 22
Deep-Work 23
    Kann Deep-Work dabei helfen ein Experte zu werden? 23
    Wie man Deep-Work macht 24
Das Flow-Prinzip 24
    Messbarkeit von Flow 25
    Fazit 25
Gewohnheiten 26
Prokrastination 27

DIE LÖSUNG: FlowLink 30

Value Proposition 30
So inconvenient wie nötig 30
Charakter der Anwendung 31
Die erste Einrichtung 32
Die Deep-Work-Phase: Der Rote Faden 33
    Das Interface 34
    Das Draft-Ritual 36
    Der Leitfaden 39

Der Rahmen 40
Das Boot-Up-Ritual 41
Das Shut-Down-Ritual 42
Das Recap 42
Das Wochen-Recap 45
Das Framing-Ritual 46
Sound-Konzept 50
Interaktions-Sounds 50
Die Deep-Work-Playlist 51
Iterative Optimierung 52
Alternative Interfaces 53
Visuelles Konzept 54
So sieht das aus. 54
Die Farbwahl 54
Schriften 56
Icons und Sonstiges 58

SO FUNKTIONIERTS 60

Der Prototyp von FlowLink 60
Prinzipien des Prototypen 60
Technology Stack 61
Electron 65
Plattformübergreifend 65
Nativ vs. Plattformübergreifend 65
Nachteile von Electron 68
Test Driven Development 70
Spectron und Ava 72
Electron Builder 74
Code Signierung 74
Musik Streaming 75
Analytics und Tracking 76
Deployment und Hosting 77
Aufbau der Electron Anwendung 78
Programmablauf 78
Verantwortungsbereiche und Aufteilung 80
WindowManager 82
EventHandler 82
InputManager 83
Timer 83
Ein paar Worte zu Modularität 84
Interne Abläufe (Sequenzdiagramm) 84
Technik Ausblick 86
Das Zukunftsbild 87

INTRO

Die Menschen hinter FlowLink

Das Team hinter FlowLink besteht aus zwei jungen Designern ...

„Wir stellen uns einfach selbst vor.
Wir heißen Sebastian Schuchmann und Robin Bittlinger. Derzeit studieren wir Interactive Media Design (IMD) an der Hochschule Darmstadt.

IMD ist ein interdisziplinärer Studiengang bestehend aus Informatik, Management und Design. Dieses Dokument ist Teil unserer Bachelorabgabe. Unser Schwerpunkt liegt auf Design und Informatik. Wir sind eher unkonventionelle und direkte Menschen. So ist auch diese Dokumentation etwas persönlicher und so, auf den Punkt, wie möglich.

Wir wünschen viel Spaß und einen guten Lesefluss."

Formalien

Die Bachelorarbeit besteht aus der Forschungsarbeit und dem Bachelorprojekt. Letzteres muss nicht auf der Forschungsarbeit aufbauen. Die Abgabe für das Bachelorprojekt besteht aus einer gedruckten Dokumentation, welche Du gerade liest, einer interaktiven Dokumentation, einem Prototypen, dem dazugehörigen Trailer und einer Abschlusspräsentation.

Der zeitliche Rahmen für das Projekt spannt sich vom 10. Mai bis zum 29. August 2019. Die Abgabe für dieses Dokument ist am 05. August und stellt das Ende der Konzeptphase des Projektes dar. Was Du nun lesen wirst, ist eine von uns erdachte Lösung für ein von uns herausgearbeitetes Problem, basierend auf einer von uns gewählten Zielgruppe.

In der folgenden Darstellung ist unsere zeitliche Einteilung hin zur finalen Abgabe zu sehen.

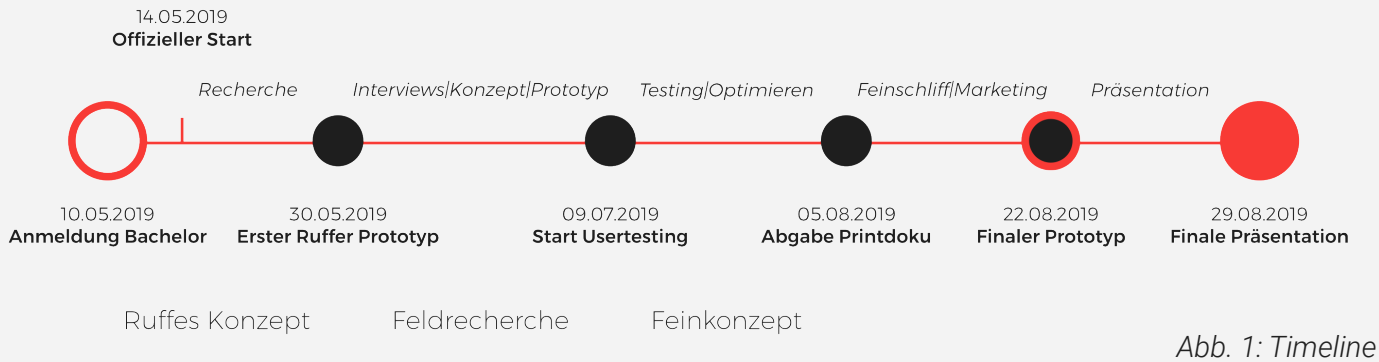


Abb. 1: Timeline

Doch was war unsere Motivation als wir den Bachelor angemeldet haben.
Hat sich diese seit dem offiziellen Arbeitsstart verändert?

Der Anfang

„Entgegen dem Trend von Amazon und Instagram, Interfaces so bequem wie möglich und mit so wenig Klicks wie nötig, zu gestalten, wollen wir den Nutzer (in den richtigen Situationen) herausfordern."

Das war der Satz mit dem das Bachelorprojekt ins Leben gerufen wurde. Er diente als rebellischer Leitsatz und Motivator neue Methoden und Techniken zum Herantreten an die menschliche Psyche zu explorieren. Schnell entstand der Titel Inconvenient Design.

Inconvenient Design

Generell lässt sich Inconvenient Design in die Oberkategorie Behavioural Design einordnen. Dabei geht es um die Analyse menschlichen Verhaltens und um das gezielte (Aus)nutzen von Verhaltensmustern, sowie um das Anzuknüpfen an Gewohnheiten.

Convenient Design bedeutet die Nutzerführung so reibungslos wie möglich zu gestalten. Mit anderen Worten geht es darum, so viele Nutzer-Operationen, wie möglich zu eliminieren und Aufgaben zu automatisieren.

> "a good user interface is believed to minimize human labor and increase efficiency."
>
> (Rekimoto und Tsujita, 2014)

Man könnte sagen, dass der Mensch fanatisch versucht, jeglichen Widerstand und Reibung aus seinen Systemen zu verbannen.

Es ist ein Optimierungsprozess, für maximale Effizienz. (vgl. Weaver, 2019).

Dass der Mensch jedoch von komfortablem Design nicht unbedingt glücklich wird, zeigen die letzten 10 Jahre. Amazon, Instagram, AirBnB, Uber etc. hatten hier ihren großen Aufstieg und haben den Markt mit Convenient Interfaces überschwemmt.

Man könnte das als das „golden age in the war against friction" bezeichnen.

Umfragen haben gezeigt, dass der Mensch im Durchschnitt trotzdem nicht glücklicher mit seinen Anwendungen ist (Weaver, 2019).

	Traditionelle Interaktionen	„Inconvenient Interactions“
Design Goal	Reduziere menschliche Operationen	Ermutige zur menschlichen Operationen
Self-awareness	Nicht so wichtig	Verbessere die Selbstwahrnehmung
Long-term benefit	Nicht so wichtig	Erhöhe den Langzeitwert
Evaluation criteria	Effizienz	Beteiligung

Abb. 2: Inconvenient Interactions

Convenient

Nutzer-Operationen eliminieren, Aufgaben automatisieren, Reibung eliminieren.

Inconvenient

Aufforderung expliziter menschlicher Teilnahme, verbunden mit Anstrengung schafft Reibung.

> “Nature understands the power of friction while we have become blind to it.”
>
> (Weaver, 2019)

Generell lässt sich Inconvenient Design als die Aufforderung expliziter menschlicher Teilnahme, verbunden mit geringer Anstrengung, um ein Nebenziel zu erreichen, beschreiben. Diese Interaktionen werden inconvenient interactions genannt. (vgl. Rekimoto und Tsujita, 2014)

Es kann nämlich wichtig sein, eine weniger bequeme Interaktion zu erschaffen. Mögliche Anwendungsfälle wären, wenn man

ein Langzeitziel erreichen will

Das System bietet dabei nicht unbedingt einen direkten Vorteil durch die vermehrte Anstrengung während der Nutzung, sondern zahlt sich auf lange sicht aus (vgl. Rekimoto und Tsujita, 2014). Ein plakatives Beispiel für eine solche Anwendung wäre eine Kaffeemaschine bei der man zehn Kniebeugen machen muss, um einen Kaffee zu erhalten. Der Prozess erhöht die körperliche Fitness über die Zeit und übernimmt gleichzeitig den Effekt für den der Kaffee ursprünglich gedacht war, nämlich ein Wachheitsgefühl am Morgen.

den Nutzer vor sich selbst schützen muss

Wir brauchen Interaktionsdesign, um menschliches Verhalten zu verändern. Beim Punkt Selbstschutz geht es darum den Nutzer vor schädlichem Verhalten zu schützen. Ein gutes Beispiel für ein Interface, dass zu convenient ist, stellt der Amazon-Dash-Button da, welcher den Kaufprozess zu sehr vereinfacht und so zu Impulskäufen verleitet. Der Button kann überall in der Wohnung angebracht werden und ein einzelner Knopfdruck reicht aus, um eine Bestellung auszulösen. Deshalb wurde der Amazon-Dash-Button vom deutschen Verbraucherschutz verboten.

die Wertschätzung eines Prozesses steigern möchte

Manche Menschen mögen umständliche Handhabung. Das zeigt sich in der Präferenz zwischen einem Automatikgetriebe und einem manuellen. Außerdem steigert es den gefühlten Wert etwas selbst zu erschaffen, wie zum Beispiel beim Kochen oder bei handwerklicher Betätigung. Auch umständliche Architektur kann arbeitsfördernd sein, wenn sie ein Langzeitziel verfolgt oder einfach nur durch eine schwer zu öffnende Tür eine geborgene Atmosphäre erschafft (vgl. Rekimoto und Tsujita, 2014).

„Too much friction destroys value. But so does too little. We need to reassess our relationship with friction. We reduce the likelihood of value, purpose, and satisfaction when we focus on smoothing increasingly benign inconveniences and ignore the significant friction holding back much of the world.
[...]
All friction is not created equal. If we are designing products for human value, we can't treat all problems the same way. We need to understand which problems are worth solving because they truly hold people back and which problems may not actually be problems at all. The nuance of this difference, just as we see in nature, is key to maximizing a product's value to humanity."
(Weaver, 2019)

Ein gewisser Widerstand ist wichtig, um ein Produkt wirklich wertschätzen zu können. Ein gutes Beispiel dafür ist der Ikea-Effekt. Ein Experiment bei dem zwei Probandengruppen fertig zusammengebaute Möbel und die Selbstbau-Variante von Ikea bekamen, wurden gefragt welches Möbelstück sie mehr wertschätzen. Die Antwort fiel eindeutig auf die selbst zusammengebaute Variante von Ikea. Daraus lässt sich die Friction-vs-Value-Curve ableiten: die richtige Menge an Widerstand bringt die maximale Wertschätzung (vgl. Weaver, 2019).

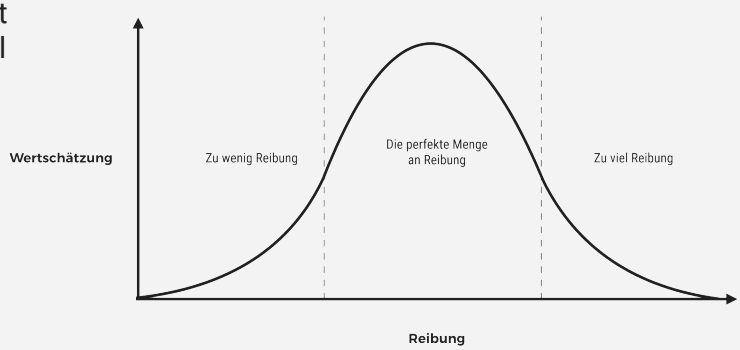


Abb. 3: Value vs Friction

Guides für gute Inconvenience

Bei der Entwicklung eines unbequemen Systems kann man sich gut an der Spieleindustrie oder menschlichen Traditionen orientieren.

Gamification ist ein gutes Mittel, um ein inconvenient Interface durchzusetzen. So folgt ein Computerspiel immer einem Ziel. Zudem gibt es Feedback-Systeme sowie die freiwillige Teilnahme an der Geschichte. Über das Flow-Prinzip wird die Aufmerksamkeit des Spielers erlangt und über den Verlauf des Spiels aufrecht erhalten.

Auch Rituale sind mit extra Aufwand verbunden, geben dafür einer Tätigkeit besondere Bedeutung, da eine passende Atmosphäre geschaffen wird. Es geht um die Verbundenheit der Teilnehmer und ein Fokus auf die Aufgabe selbst. Ein gutes Beispiel für ein erfolgreiches inconvenient Interface wäre die japanische Teezeremonie .
(vgl. Rekimoto und Zujika, 2014).

um den Nutzer zu manipulieren, etwas bestimmtes zu tun.

Inconvenient Design wird häufig unbewusst oder unter anderem Namen eingesetzt. So wird man in Supermärkten gezwungen durch den ganzen Laden zu gehen, um an die Kasse zu gelangen oder der Ausgang eines Stores ist doppelt so groß wie der Eingang. Auch in der Softwareentwicklung ist inconvenience ein längst verbreitetes Stilmittel, um gewisse Interessen durchzusetzen. So stoppt bei Spotify die Werbung, wenn man die Lautstärke zu weit herunterdreht oder der Prozess, sich aus einer App auszuloggen wird künstlich verkompliziert (vgl. Ramirez, 2017).

Dabei ist es wichtig, zwischen inconvenience und dark pattern zu differenzieren:

"A Deceptive UX Pattern is an unethical method or mechanic that forces users through an unexpected path to later profit from them. [...] In contrast, a Designed Inconvenience is simply a design choice that can annoy users but would rarely hurt them."
(Ramirez, 2017)

Der Weg zur Zielgruppe

Durch den Start mit Inconvenient Design tat sich die Frage auf, was sich denn nun durch mehr Reibung verbessern ließe. Es wird in verschiedene Richtungen recherchiert und Parallelen zwischen dem Ikea-Effekt und dem Flow-Prinzip entdeckt. Kurzgesagt entsteht durch die richtige Menge an Anstrengung ein Zustand maximaler Konzentration, Flow. Diese richtige Menge an Anstrengung ist vergleichbar mit der „Friction-Value-Curve“ des Ikea-Effekts. Das Flow-Prinzip findet einerseits bei Computerspielen, aber auch beim Leisten schwer reproduzierbare Arbeit, Einsatz.

Die Idee kommt auf den Arbeitsprozess über gezielte Inconvenience zu optimieren. Es wird das Konzept von Deep-Work entdeckt, welches das Erledigen von schwer reproduzierbarer Arbeit fördert und somit die perfekte Ergänzung zu der vorherigen Recherche darstellt. Schnell wird Deep-Work der Kern des Konzeptes. So grenzt sich die Zielgruppe auf Knowledge Worker ein. Die Relevanz verschiedener Knowledge Worker wird evaluiert und man kommt zu dem Schluss, dass bei niemandem schwer reproduzierbare Arbeit so elementar für den generellen Fortschritt der Menschheit ist wie bei Wissenschaftlern.

Die Interviews: Methode

Die geführten Interviews mit der Zielgruppe fanden ausschließlich über Skype statt. Um das Gespräch flexibel zu halten und sich gegebenenfalls dem Gesprächspartner anzupassen, sind die Fragen stichpunktartig formuliert und in keiner festen Reihenfolge. Der Links angezeigte Leitfaden diente als eine Checkliste für den Interviewer. So konnte er stets überprüfen, welche Fragen noch zu beantworten sind.

Die Gespräche wurden mit Einverständnis der Befragten, anonymisiert auditiv aufgezeichnet, unter der Bedingung bei direkten Zitaten vorher noch mal Absprache zu halten. Alle Informationen wurden sinngemäß transkribiert und werden als reine Datenquelle behandelt, um die Zielgruppe genauer zu definieren. Dabei werden keine personenbezogenen Daten oder Zitate verwendet. Alle verwendeten Zitate wurden anonymisiert.

Bachelorprojekt Interview Leitfaden Wissenschaftler

MAIN GOAL

Probleme identifizieren die Wissenschaftler haben, die sie daran hindern, um bedeutsame Arbeit zu leisten durch eine Customer Journey.

Anrede: _____ Titel: _____

Datum: _____

EINLEITUNGSFRAGE

„Betreiben Sie derzeit wissenschaftliche Forschung und wenn ja, woran arbeiten Sie gerade?“

INTERVIEWLEITFADEN

Zum Abkotzen bringen (Trigger-Words: Zeitdruck, Finanzierung)

CHECKLISTE AN PUNKTEN DIE BEANTWORTET SEIN SOLLTEN

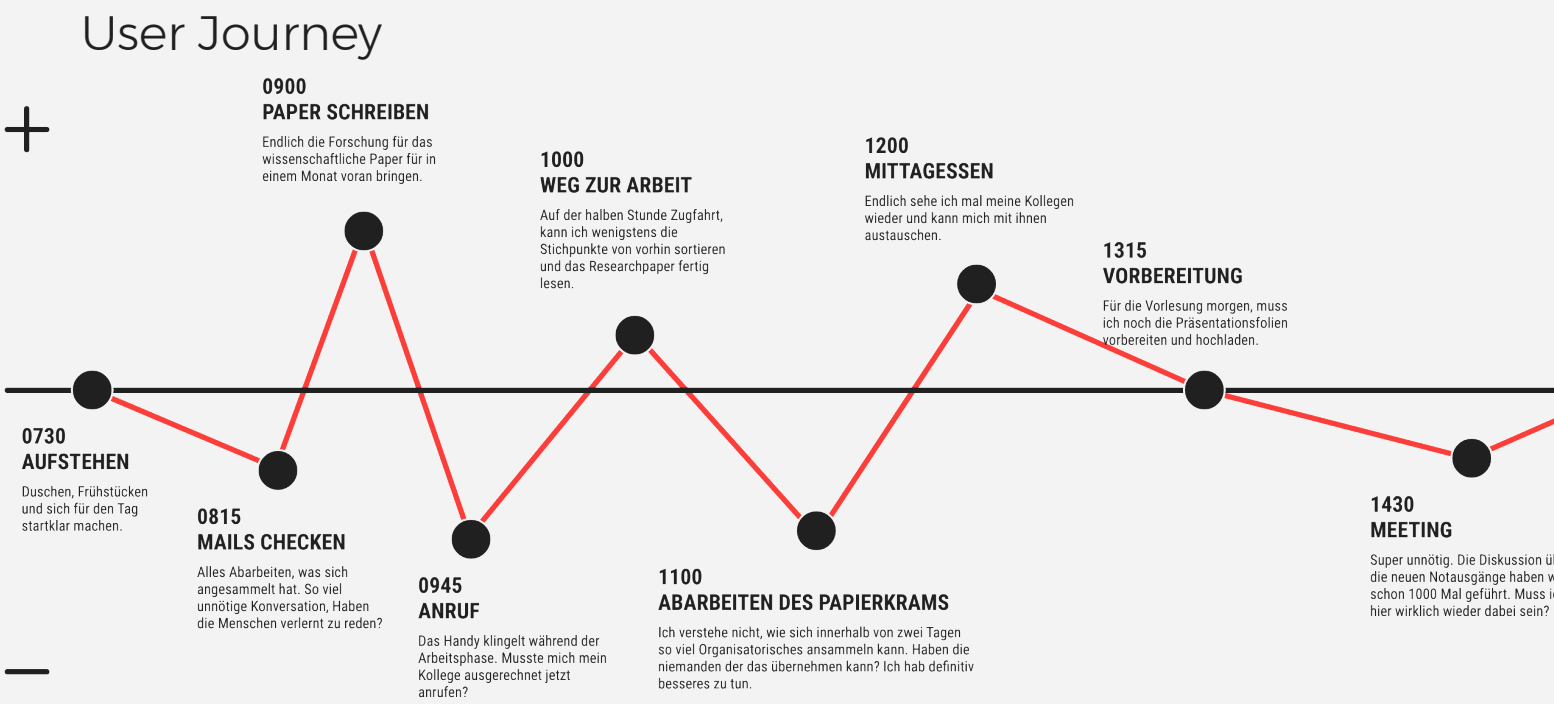
- ☐ Welche verschiedene Arbeitsphasen?
- ☐ Tagesablauf innerhalb dieser Phasen?
- ☐ Worauf freuen Sie sich? …warum?
- ☐ Worauf haben Sie gar keinen Bock? …warum?
- ☐ Welche Aktivitäten würden Sie als besonders relevant einstufen?
- ☐ Welche Aktivitäten sind unwichtig?
- ☐ Haben Sie bestimmte Gewohnheiten/Rituale vor, nach oder während der Arbeit?
- ☐ Wie verbringen Sie ihre Pausen?
- ☐ Wie gut sind Sie mit anderen Wissenschaftlern vernetzt/ Teamarbeit?
- ☐ Welche Tools verwenden Sie?
- ☐ Wie finanzieren Sie ihre Forschung?
- ☐ Ist Zeitdruck ein relevantes Problem?
- ☐ Warum sind sie Wissenschaftler geworden?
- ☐ In wiefern dürfen wir ihre Daten weiterverarbeiten?

Abb. 4: Interview-Leitfaden

Die Zielgruppe: Wissenschaftler

Die Interviews: Findings

In den Interviews wurden ausschließlich akademische Wissenschaftler befragt. Die Persona wurde auf Basis der Befragungen erstellt und durch gezielte Recherche ergänzt. Da die Zielgruppe keinen geregelten Tagesablauf hat, sollte die User Journey eher als ein Beispiel, für einen möglichen Tag im Leben eines akademischen Wissenschaftlers, gesehen werden.



Zusammenfassend lässt sich sagen, dass die Zielgruppe aktiv wissenschaftliche Forschung betreibt. Es gibt keine verschiedenen Arbeitsphasen über das Jahr verteilt. So wie auch der Arbeitsalltag keineswegs routiniert abläuft, sind die Aufgaben meist nicht lange im Voraus planbar. „Wissenschaftler zu sein ist kein Nine to five Job." (Müller, 2019) Der Arbeitstag ist dabei meist länger als acht Stunden. Eine von wenigen Konstanten ist das gemeinsame Mittagessen mit Kollegen, welches jedoch gelegentlich übersprungen wird, wenn viel Arbeit ansteht. Neben dem akademischen Betrieb wird versucht die Forschung voranzutreiben. Da hat jeder seine eigenen Methoden und Zeiten. Meistens wird versucht, bevor es zur Universität geht, eine Arbeitssession einzulegen. Manche Arbeiten sogar auf dem Weg dorthin, „um sich schonmal in Arbeitsstimmung zu versetzen".

Im Universitätsalltag und der Forschung sind die Höhepunkte der entweder virtuelle oder physische Austausch mit Kollegen und Fachleuten, die in einem ähnlichen Themengebiet tätig sind. Das Erfreuliche daran ist einerseits die soziale Interaktion, andererseits das Weitergeben des eigenen Wissens, der Zugewinn neuen Wissens und das Feedback zu bestehenden Projekten. So macht auch das Begutachten neuer, gelungener Projekte Spaß. Besonders gefreut wird sich jedoch auf das sogenannte „konzentrierte Hineinarbeiten in ein bestimmtes Themengebiet, gerne auch über mehrere Tage".

Was hingegen als unangenehm empfunden wird, sind vor allem Aufgaben, „die eigentlich jemand anderes übernehmen könnte" oder „alles was der Forschung im Weg steht". Dazu gehören Nachfragen aus der Verwaltung, Reisekostenabrechnungen, Finanzkalkulation, E-Mails beantworten etc., kurz Organisatorisches. Einer der größten Vorteile des wissenschaftlichen Alltags, nämlich die Flexibilität, bringt auch eines der größten Probleme mit sich: die Einstiegshürde in diese hoch konzentrierten Arbeitsphasen, kurz in den Flow. Um diesen Zustand zu erreichen, haben wir verschiedene Ansätze gehört. Musik wird während der Arbeit gehört, interessant dabei ist jedoch, dass es weniger um die Musik selbst zu gehen scheint, sondern eher um den Akt

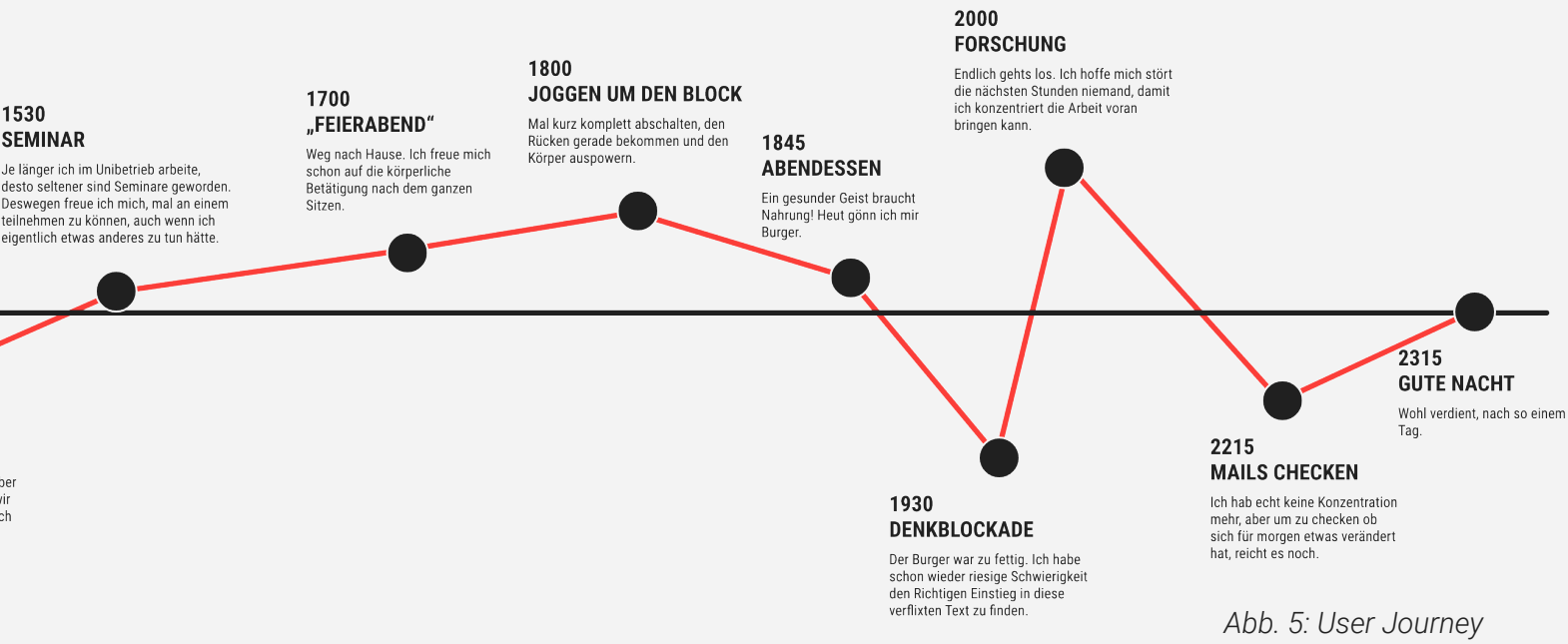


Abb. 5: User Journey

des Kopfhöreraufsetzens als Trigger-Element. So reicht allein das Aufsetzen schon aus, um die Gewohnheit hoch konzentriert zu arbeiten, auszulösen. Eine weitere Angewohnheit ist das Herumlaufen, während dem Denkprozess. Wird eine neue Aufgabe angefangen, wird vorher einmal um den Block gelaufen, um das Gehirn neu zu starten oder den Arbeitsweg schon zu nutzen, um bei Ankunft direkt im Arbeitsmodus zu sein.

Nach der Universität wird meist versucht einen Ausgleich zur Arbeit zu schaffen. So wird Sport getrieben, ein Instrument praktiziert oder eine ähnlich körperlich oder geistig beanspruchende Tätigkeit ausgeübt. Generell scheint die Zielgruppe kein Problem mit dem Abschalten nach der Arbeit oder der Gestaltung der Pausen zu haben.

Die Pausen werden meisten nicht mal eingeplant und dann genommen, wenn sie als nötig empfunden werden. Sie beinhalten meistens einen Spaziergang an die frische Luft oder zur Kaffeemaschine und einen kleinen Plausch mit Kollegen. Somit sind fast alle Kriterien für eine erholsame Pause gegeben: der Kontextwechsel, die Bewegung und der soziale Austausch (vgl. Linhart, 2012). Viele der Befragten verzichten auch auf das Abschalten nach der Arbeit: „ich möchte Forschung und Freizeit nicht trennen“.

Deadlines sind kein Problem für die Zielgruppe. Diese seien früh genug bekannt und Projekte, die zu knapp seien, könne man einfach ablehnen. Tatsächlich sind Deadlines sogar gewünscht oder selbst auferlegt, um noch mehr Arbeit in Form von wissenschaftlichen Arbeiten und Artikeln in Magazinen fertig zu bekommen.

Die Finanzierung stellt sich ebenfalls als nicht problematisch dar, da die meisten Projekte bereits von einer Stiftung, Fonts oder der Regierung im Vorfeld finanziert und dann nur noch ausgeschrieben werden. Dabei ist die Zielgruppe gut vernetzt. Jedoch eher innerhalb so genannter Research-Gruppen. Diese bestehen meist aus 20 – 70 Menschen, innerhalb des deutschsprachigen Raums, plus einzelnen Fachleuten und Koryphäen. ResearchGate und LinkedIn sind zwar ein Begriff, werden jedoch nicht aktiv benutzt.

Was motiviert unsere Zielgruppe bedeutsame Arbeit leisten zu wollen?

Besonders viel Wert wird auf das Eigenmanagement in Verbindung mit der Selbstverantwortung gelegt. Dabei ist sowohl der Prozess spannend, wie neues Wissen generiert wird, als auch die Weitergabe des generierten Wissens. Letzteres macht vor allem dann Spaß, wenn man Begeisterung auslösen kann und das Gefühl hat dieser Person geholfen zu haben. Das Unbekannte erforschen, komplexe Systeme auf atomare Prozesse zurückführen, das „Warum“ beantworten und sich mit anderen darüber Austauschen sind alles grundlegende Motivationen für viele der Befragten. Die Wichtigkeit von Anerkennung für das Geleistete ist sehr personenabhängig. Gerade bei jungen Wissenschaftlern ist Anerkennung wichtig, um ein gutes Resume aufzubauen und um an relevante Projekte zu kommen. Dass was gearbeitet wird, soll Hand und Fuß und somit nachweislich Sinn haben.

Was findet sich direkt in der Anwendung wieder?

Über die Interviews konnten wir die Zielgruppe besser verstehen. Zuerst vermutete Probleme, wie Finanzierung oder Zeitdruck konnten ausgeschlossen werden und wir konnten die Einstiegshürde in den Flow als das zentrale Problem identifizieren.

Bei mir ist es so, wenn ich einmal den Anfang gefunden habe, dann geht das schnell. Die Barriere liegt am Anfang. […] weil ich meistens schon weiß, worauf es hinausläuft, ich brauche dann nur den Aufhänger, wo man mit beginnt.“ (Moller, 2019)

„Forschungsalltag ist kein Routinejob“ (Kaiser, 2019)

Auch wie flexibel die Anwendung aufgrund des Tagesablaufes sein muss, hat die Anwendung maßgeblich beeinflusst und viele Ideen rausgeschmissen und vereinfacht.

Wir hatten Glück, was die befragten Wissenschaftler angeht. So befindet sich unter ihnen ein Arbeitspsychologe, welcher direkt noch Tipps zu Arbeit und Selbstreflexion geben konnte.

„Ich weiß nicht, wie die Leute sich ändern sollen, ich erarbeite das eher mit den Leuten in Gruppen.“ (Ruben, 2019).

„Ich mache mir selbst Druck, weil ich gerne mehr Schaffen würde.“ (Ruben, 2019)

Auch der Fokus auf den selbst zu stellenden Timer in der Pomodoro-Technik wurde durch die Interviews noch einmal bestätigt.

Generell wurde während des Konzeptes immer wieder mit den Interviews und der daraus resultierenden Persona abgeglichen und zusammen mit den Usertests das Konzept und der Prototyp iterativ angepasst. Trotzdem ist FlowLink aufgrund der flexiblen Anforderungen der Zielgruppe von nahezu allen Knowledge Workern verwendbar.

Persona

Beim nutzerzentrierten Design geht es darum, das System an die Bedürfnisse des Nutzers anzupassen. Damit das funktioniert, darf man als Designer nicht von den eigenen Bedürfnissen ausgehen, sondern muss sich in die Zielgruppe hineinversetzen. Wie wenn man empathisch mit anderen Menschen ist, geht es auch hier darum, sich in die Schuhe des anderen hineinzustellen (Bittlinger, 2019).

Ein Tool, damit dies gelingt ist die Persona. Die Persona gibt den erarbeiteten Fakten ein Gesicht und lässt die Zielgruppe somit real werden. Diese personifizierten Daten helfen in Diskussionen sich zu einigen, weil stets die Persona, wie ein realer Mensch, befragt werden kann. So stellt sich auch beim Erarbeiten dieses Konzeptes immer wieder die Frage: „Könnte Charlez damit etwas anfangen?“

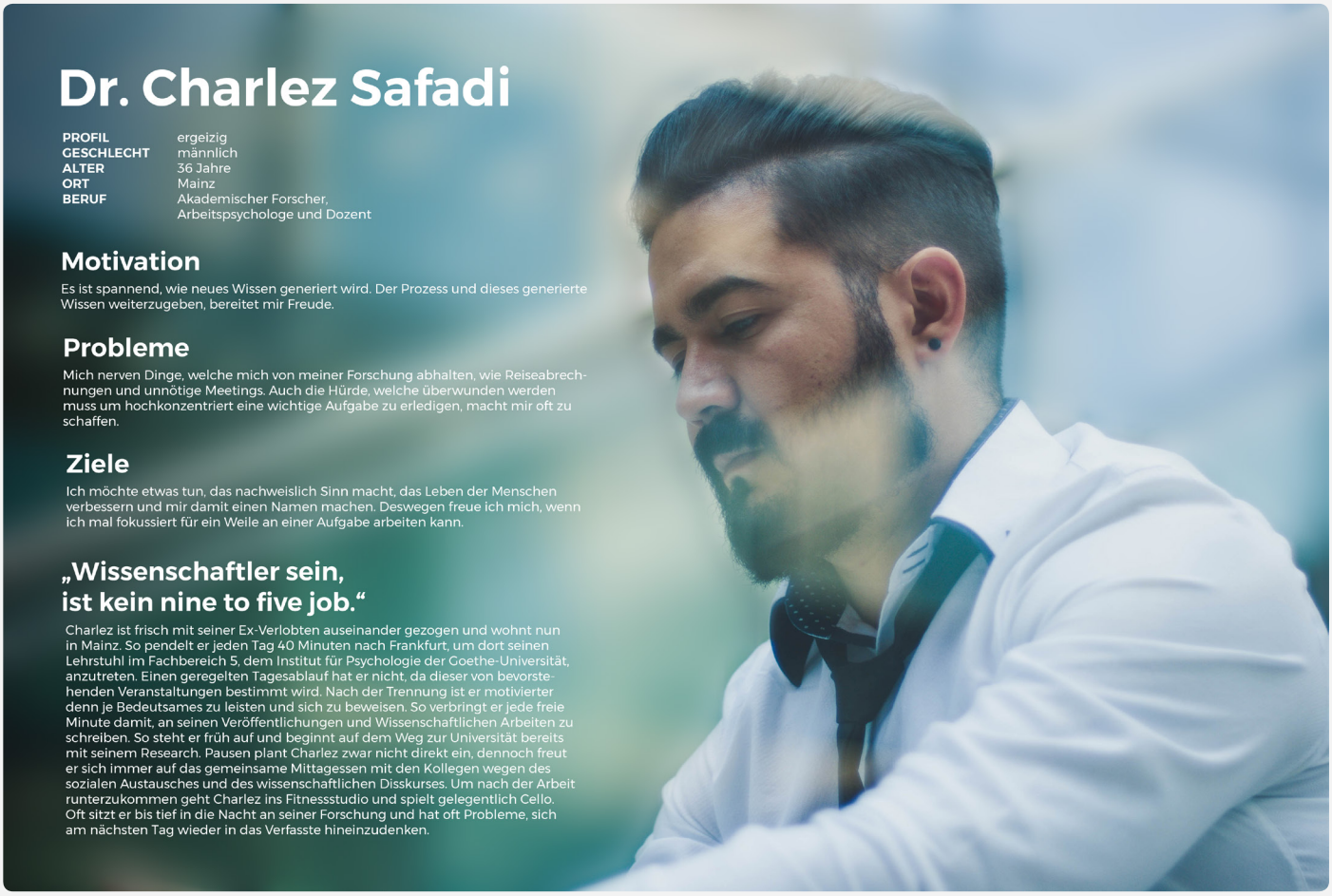


Abb. 6: Persona Charlez

Das Problem

In den Interviews mit der Zielgruppe hat sich ein Problem für alle Befragten als relevant herausgestellt. Jeder der Beteiligten ist gerne Wissenschaftler und genießt die hoch konzentrierte Arbeit an komplexen Problemen. Alles, was die Wissenschaftler davon abhält, stellt deswegen automatisch ein Problem da.

Nicht nur, weil es darum geht die Produktivität zu maximieren. Die Wissenschaftler genießen ihre Arbeit und haben das Bedürfnis, ihre Neugierde zu stillen. Interessanterweise waren dabei externe Störfaktoren für die Befragten kein Problem. Alle berichteten vielmehr von einem schwierigen und häufig zeitintensiven Prozess, den sie benötigen, um in den Flowzustand zu gelangen. Wenn zum Beispiel geplant ist einen Artikel zu schreiben, schilderte ein Befragter, wie er vorher 30 Minuten lang im Raum umher läuft, bevor auch nur ein Wort in den Computer getippt ist. Dieses Phänomen nennen wir die Hürde, die es bedarf in den Flow Zustand zu gelangen. Dabei ist dies aber keineswegs nur verschwendete Zeit. Sie wird nämlich dafür genutzt ein Gedankenkonstrukt aufzubauen.

Es wird ein Rahmen abgesteckt, der markiert, wie der Artikel enden soll, was die Kernaussage ist und wie sinnvoll zu dem nächsten Text übergeleitet werden kann. Erst durch das Abstecken des Rahmens kann ein Schreibfluss effektiv erzeugt werden. Gleiches gilt natürlich für jede andere Aufgabe, die die komplette kognitive Aufmerksamkeit benötigt auch.

Trotz der Notwendigkeit der Hürde sehen wir hier Optimierungsbedarf. Denn wenn die Aufgabe unterbrochen wird, muss das komplette Gedankenkonstrukt wieder Stück für Stück mühsam aufgebaut werden, obwohl es schon einmal stand. Des Weiteren kann der unstrukturierte Aufbau des Gedankenkonstruktes schnell zu Prokrastination führen, weil man keinen sinnvollen Anfangspunkt findet. Das größte Potenzial sehen wir aber in der Flow-Phase selbst. Da in der Flow-Phase auch die Arbeit an komplexen Problemen spielend leicht fällt, ist das die perfekte Gelegenheit, um etwas dieser wertvollen Denkmasse abzugreifen und sie so zu verarbeiten, dass der Wiedereinstieg einfacher gelingt.

Ein erleichterter Einstieg in den Flow würde bei Erfolg einige Vorteile mit sich bringen. Natürlich bleibt den Wissenschaftlern so wertvolle Zeit erspart, die in andere Arbeit oder wertvolle Pausen oder Freizeit investiert werden kann. Doch das ist nicht alles. Zudem würden kognitive Ressourcen geschont werden, weil ein Teil der intensiven Denkarbeit wegfällt. Dies sorgt für mehr Deep-Work Kapazität an einem Tag. Darüber hinaus sind vor allem die Flow-Phasen jenige Arbeitszeiten, die den Wissenschaftlern viel Freude bringen und somit würde ein erleichterter Einstieg in diese Phasen auch das Wohlbefinden steigern.

NACHFORSCHUNG

Bevor wir uns in das Konzept und eine mögliche Lösung für das Problem stürzen konnten, musste erst mal eine Grundlage an Informationen geschaffen werden. Da unser Leitsatz die Optimierung des Arbeitsprozesses ist, muss zunächst die Arbeit generell untersucht werden und in dem Kontext Deep-Work, eine Methode, welche speziell für Knowledge Worker entwickelt wurde, um schwer reproduzierbare Arbeit zu leisten. Knowledge Worker sind auch der Überbegriff für unsere Zielgruppe, weshalb diese im gleichen Zuge behandelt werden.

Da das in den Interviews festgestellte Hauptproblem der Zielgruppe, der Einstieg in den Flow ist, darf das Flow-Prinzip in der Recherche nicht fehlen. Das Konzept beschäftigt sich viel mit dem Schaffen von Gewohnheiten und dem Vorbeugen von Prokrastination. Somit schließen sich diese beiden Punkte noch an die Recherche an. Wie wir das oben beschriebene Problem gelöst haben, seht ihr nach diesem Kapitel. Wer die Recherche überspringen möchte:

Die Lösung beginnt auf Seite XY

Arbeit

Der Begriff der Arbeit erhält je nach Bereich eine andere Bedeutung. Es handelt sich immer, um eine Tätigkeit, die erledigt wird, um gewisse Ziele zu erreichen. Nach Hegel gibt es vier Ziele, die mit Arbeit angestrebt werden:

Der Erhalt von materiellen Gütern
Aufrechterhaltung und Förderung des gesellschaftlichen und sozialen Lebensstandards
Verbesserung des eigenen sozialen Status sowie der von Familie und Freunden
Schaffen von Innovation in der Wissenschaft, Kunst oder Politik

Die oben genannten Ziele stellen die verschiedenen Grundmotivationen für das Ausführen von Arbeit da. Doch es hilft noch nicht dabei zu verstehen, was Arbeit für den Arbeitenden bedeutsam werden lässt. Diese Fragestellung versucht die Job Characteristic Theory (JCT) zu beantworten. Dabei hat sie konkret das Ziel, Richtlinien zu definieren mit der der Arbeitsplatz so gestaltet werden kann, dass der Arbeitende seine Arbeit als bedeutsam empfindet. Hackman und Oldham haben die JCT in dem Buch Work Redesign (1980) beschrieben und sie dienen noch heute als gute Orientierungspunkte.

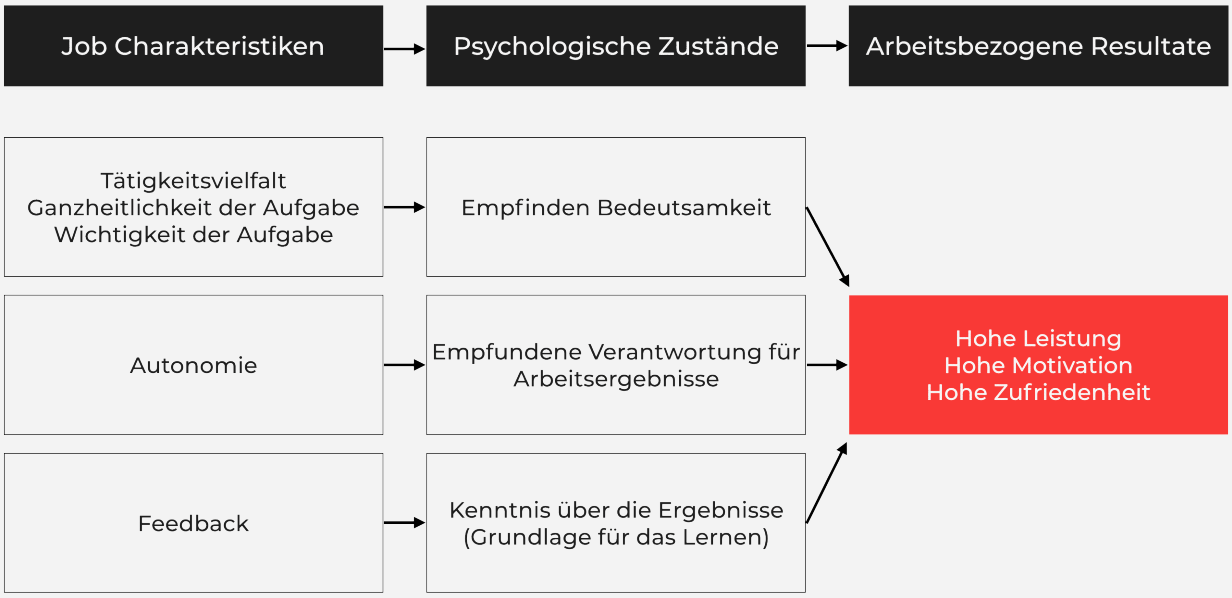


Abb. 7: Job Characteristic Theory

Wenn also alle Job-Charakteristiken erfüllt werden, ist die Leistung des Arbeiters nicht nur hoch, sondern er empfindet zudem eine hohe Motivation sowie Zufriedenheit.

Deep-Work

Der Deep-Work Begriff entstammt aus dem gleichnamigen Buch von Cal Newport und beschreibt Arbeit, die in einem Ablenkungsfreiem Zustand erledigt wird und die kognitiven Fähigkeiten komplett ausnutzt. Laut Newport schaffen diese Anstrengungen Mehrwert, verbessern die eigenen Fähigkeiten und sind schwer zu reproduzieren. (Newport, 2016)

Das Pendant zu Deep-Work ist die Shallow-Work. Im Vergleich zur Deep-Work ist sie leicht zu reproduzieren und beinhaltet meist kognitiv mühelose Aufgaben, die auch mit Ablenkung erledigt werden können.

Der Deep-Work Begriff bezieht sich auf die Arbeit von Knowledge Workern, deren Arbeitsleistung vor allem durch geistige Anstrengung erbracht wird. Beispiele für Knowledge Worker sind Programmierer, Wissenschaftler, Designer, Ingenieure, Anwälte und Akademiker. Eine klare und einheitliche Definition für Knowledge Worker gibt es nicht. (Pyöriä, 2005)

Doch allgemein müssen alle Knowledge Worker die Fähigkeit besitzen flexibel auf neue Probleme zu reagieren, da diese Jobs selten routiniert ablaufen. (Reinhardt, 2011)

Im folgenden werden wir die Deep-Work These evaluieren. Dafür werden wir uns zuerst anschauen was den Experten vom Laien unterscheidet und inwiefern Deep-Work für die Transformation zum Experten notwendig ist. Danach schauen wir uns die Bedeutung von Aufmerksamkeit bzw. Ablenkungsfreiheit für das Erledigen von Aufgaben an.

Kann Deep-Work dabei helfen ein Experte zu werden?

In dem Werk „Expert Performance Its Structure and Acquisition“ beweist K. Anders. Ericsson, dass der größte Unterschied zwischen Experten und Laien in einem Feld nicht angeborene Fähigkeiten sind, sondern das Lernen von komplexen Fähigkeiten. Ericsson schildert, das Experten komplexe Fähigkeiten nur durch fokussiertes und mühevolles Training erlernen. Dabei muss das Training zeitlich limitiert sein. Egal in welchem Bereich, die Experten schaffen selten mehr als vier Stunden am Tag fokussiert zu trainieren. (Ericsson, 1994)

Dieses fokussierte und mühevolle Training ist Deep-Work, da es die kognitiven Ressourcen maximal auslastet und nur in einem Zustand ohne Ablenkung ausführbar ist. Doch welche Bedeutung hat die Aufmerksamkeit für das Erledigen von Aufgaben?

In einem Experiment entdeckt Leroy 2009 das Phänomen von „Attention Residue“ (Aufmerksamkeitsrückstände). Für Menschen ist es schwierig von einer Aufgabe zu einer anderen zu wechseln ohne das etwas Aufmerksamkeit an der alten Aufgabe zurückbleibt.

Sie beschreibt, dass man vorher das Denken von Aufgabe 1 komplett abgeschlossen haben muss. Das funktioniert nur effektiv, wenn Aufgabe 1 komplett abgeschlossen wurde. Doch auch allein das reicht nicht aus. Wenn Zeitdruck herrscht, kann zudem schneller umgeschaltet werden. Dies begründet weiter Deep-Work, weil konstante Unterbrechungen der Aufgabe für Aufmerksamkeitsrückstände sorgen und Aufgaben somit nicht optimal erfüllt werden können. (Leroy, 2009)

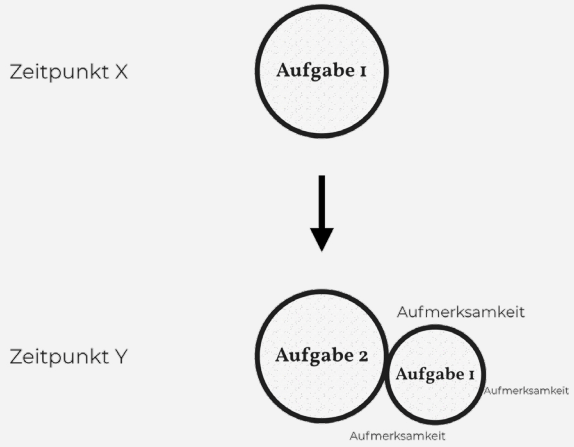


Abb. 8: Aufmerksamkeitsrückstände

Wie man Deep-Work macht

Um Deep-Work erfolgreich leisten zu können, sollten prinzipiell alle Störfaktoren minimiert werden. Zudem erfordert Deep-Work Übung und kann nicht lange Perioden am Stück ausgeführt werden. Die empfohlene Menge an Arbeit für eine Deep-Work Session beträgt zwischen 60 und 90 Minuten und kann praktisch nicht mehr als vier Stunden am Tag betragen. (Newport, 2016)
Nach dieser Zeitdauer sind die kognitiven Ressourcen soweit benutzt, dass Deep-Work nicht mehr ideal möglich ist. Des Weiteren sollte nach jeder ca. 60 bis 90 Minuten langen Session eine Pause zwischen ca. 10 bis 20 Minuten stattfinden.

Mit der Zeiterfassungs- und Produktivitätsapp DeskTime untersuchte die Draugiem Group die Gewohnheiten der produktivsten Mitarbeiter und lernte, dass die produktivsten Personen im Durchschnitt 52 Minuten lang arbeiten, dann 17 Minuten Pause machen, bevor sie weitermachen. (Gifford, 2016)

Die exakten Werte sind natürlich für jeden individuell und aufgabenabhängig und sollten somit eher als grobe Richtlinien betrachtet werden.

Das Flow-Prinzip

Das Deep-Work-Prinzip ist nah mit dem des Flow verwandt, denn Deep-Work bildet durch den ablenkungsfreien Zustand und die kognitive Beanspruchung die Grundlage für Flow. In diesem Zustand befindet die Person sich mit ihrem kompletten Bewusstsein in der aktuellen Aktivität. Da somit alles außerhalb davon verschwindet, wird es auch als Gefühl von Weltvergessenheit beschrieben. (Csikszentmihályi, 1990)

Um Flow zu erleben, muss die volle geistige Leistungsfähigkeit auf eine Aufgabe gerichtet sein. Wenn man das schafft, ist nicht genug mentale Kapazität verfügbar, um hungrig zu sein, Schmerzen zu haben, oder zu Überlegen was andere gerade Denken. Damit lässt sich im Flow nicht nur anspruchsvolle Arbeit mit wenig Anstrengung erledigen, es erhöht auch die Zufriedenheit. Für den Flow Zustand darf die zu erledigende Aufgabe weder unterfordernd noch überfordernd sein. Flow stellt den Zustand da bei dem die Aufgabe optimal beanspruchend ist und somit erledigt werden und gleichzeitig ein Lernprozess stattfinden kann.

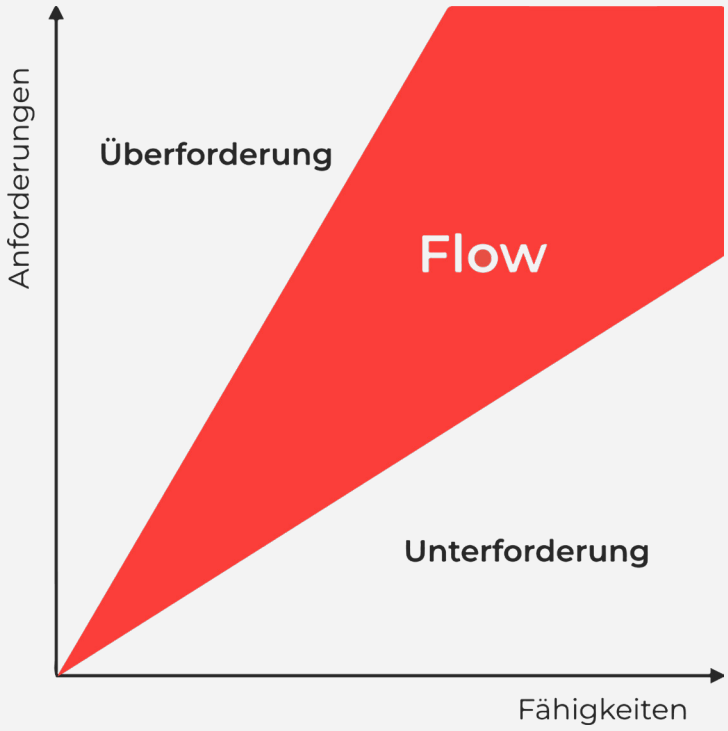


Abb. 9: Das Flow-Prinzip

Nach Csíkszentmihályi und Nakamura gibt es sechs Flow Faktoren:

1. Intensive und fokussierte Konzentration auf den aktuellen Moment

2. Aktion und Wahrnehmung verschmelzen

3. Das reflektierte Bewusstsein schwindet

4. Gefühl von Autonomie und Kontrolle über die Situation

5. Das Zeitgefühl verändert sich

6. Erfahrung ist intrinsisch belohnend.

Diese Faktoren können bei Aktivitäten in jedem Gebiet festgestellt werden egal ob es sich um Sport, Musik oder Poesie handelt. Des Weiteren beschreibt Csikszentmihályi drei Bedingungen, die erfüllt sein müssen, damit der Flow Zustand bei der Arbeit erreicht werden kann:

1. Das Ziel ist klar definiert
2. Sofortiges Feedback
3. Ein Gleichgewicht zwischen Fähigkeiten und Kapazität

Messbarkeit von Flow

Es gibt verschiedene Verfahren und Ansätze, um Flow zu messen. Der häufigste Ansatz nennt sich „experience sampling method“ (ESM) bei dem die Teilnehmer einen Pager mit sich führen und Fragen über den eigenen Zustand beantworten müssen, wenn dieser piept. (Giovanni, 2012)

Fazit

Abschließend kann gesagt werden, dass Flow für das Konzept eine besondere Bedeutung hat, weil es nicht nur die Arbeitsperformance, sondern auch das Wohlbefinden steigert. Nach Csikszentmihályi ist Flow eine von Natur aus positive Erfahrung, weil es “intense feelings of enjoyment” produziert. (Csikszentmihályi, 1988)

Zudem sagt Csikszentmihályi, dass Wohlbefinden aus persönlicher Entwicklung und Wachstum entsteht, die durch Flow gefördert wird. (Csikszentmihályi, 1997)

Gewohnheiten

„Gewohnheiten sind in ganz vielen Fällen ja unheimlich praktisch. Stell dir vor, du müsstest dir jedes Mal, wenn du dir die Zähne putzt oder beim Autofahren beschleunigst, bewusst überlegen, was du dazu genau machen musst.“(vgl. Alps, 2012)

Eine Gewohnheit ist das Gelernte, was von Gehirn und Körper automatisiert wurde. Man schont mentale Ressourcen für andere, wesentliche Funktionen wie Entscheidungsfindung, Planung und ähnliche Prozesse. (vgl. Ewert, 2019)

Es sind Handlungen, Gefühle oder Gedanken, die so oft wiederholt wurden, dass sie zur Routine werden. Das Gehirn liebt Gewohnheiten und ist stets, bestrebt neue Denk- und Verhaltensweisen zu automatisieren und alte zu erhalten. Insbesondere für das Handeln macht es das, indem es Routinen in dem Gehirnareal Basalganglien abspeichert. Sobald der richtige Reiz (Trigger) auftritt, kann der Prozess von den Basalganglien direkt abgerufen werden.
(vgl. NLP-Zentrum Berlin, 2017)

Eine Gewohnheit lässt sich in vier Komponenten einteilen:

1. Der Trigger

Der Auslöser einer Gewohnheit.

**Beispiel: Man spürt das Handy vibrieren und greift automatisch zur Hosentasche.*

2. Die Routine

Die vom Trigger ausgelöste Handlungskette.

**Beispiel: Das Handy wird aus der Hosentasche genommen, der Homebutton wird gedrückt und die Chatapp der Wahl geöffnet.*

3. Der Reward

Die Belohnung: Glückshormone werden nach vollführter Handlung ausgeschüttet.

**Beispiel: Eine Nachricht für dich wird dir angezeigt. Du hast das Gefühl, Teil am sozialen Leben zu haben.*

(vgl. Duhigg, 2012)

4. Der Glaube

Die Belohnung: Der Aberglaube, die Gewohnheit im Griff zu haben.

**Beispiel: “Ich kann jederzeit mit dem Rauchen aufhören.”*

(vgl. Shukla, 2017)

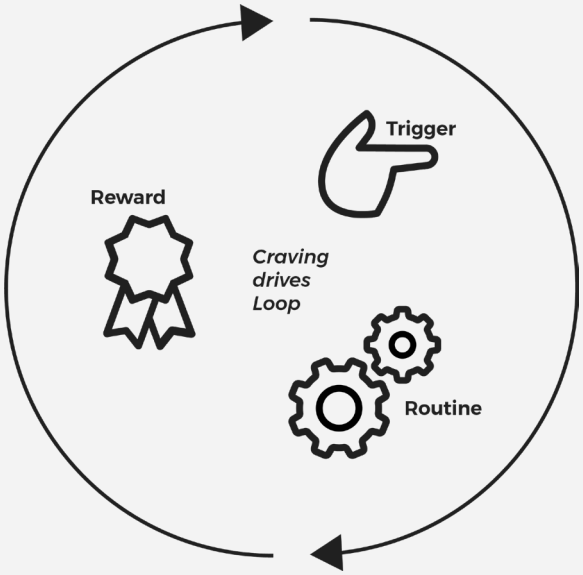


Abb. 10 nach Duhigg: Habit Cycle

Gewohnheiten werden über die Zeit stärker und somit schwerer abzulegen oder zu verändern.
(vgl. Crozier, 2018)

Es ist um einiges leichter, neue Gewohnheiten zu bilden, als alte abzulegen. Besonders einfach funktioniert das, indem man eine neue Tätigkeit, an eine bestehende Gewohnheit dranhängt.
(vgl. NLP-Zentrum Berlin, 2012)

Wer dennoch alte Gewohnheiten ablegen will, muss vor allem den Kontext verändern.
(vgl. Wood, 2012)

Eine von diesen Gewohnheiten, ist die Prokrastination.

Anhand dieser wird erklärt, wie man erfolgreich alte Gewohnheiten ablegt.

Prokrastination *Prokrastination kann auch ein Symptom bei vielen psychologischen und neurologischen Störungen sein. Nicht Verwechseln!*

Prokrastination bezeichnet die absichtliche und freiwillige Verzögerung einer beabsichtigten Tätigkeit, auch wenn die Verzögerung negative Folgen hat. Es ist der Prozess, welcher erfolgt, wenn man eine unwichtige Aufgabe vorzieht, obwohl man eine wichtigere Aufgabe zu tun hätte. Dabei geht es nicht nur um Zeitmanagement, Faulheit oder schwachen Willen. Die Basis für Prokrastination bildet eine schlechte emotionale Regulierung. (vgl. Shukla, 2019)

Emotionale Regulierung beschreibt die Fähigkeit einer Person, Emotionen, emotionale Erfahrungen und emotionale Gedanken effektiv zu verwalten, zu kontrollieren und damit umzugehen.

**Beispiel: Man ist wütend und muss Dampf ablassen.*
Positive Regulation = *Spazieren gehen*
Negative Regulation = *einen Streit anfangen.*

Manche Aufgaben versetzen uns in gute Stimmung und andere in schlechte. Ersteres wird vom Gehirn bevorzugt.

> "People procrastinate or avoid aversive tasks to improve their short-term mood at the cost of long-term goals."
>
> (Shukla, 2017)

> "to overcome procrastination, we need to address emotions and thoughts about the outcome of completing certain tasks"
>
> (Shukla, 2017)

Leicht anspruchsvolle Aufgaben können stressig werden, wenn eine Person zu viel nachdenkt.

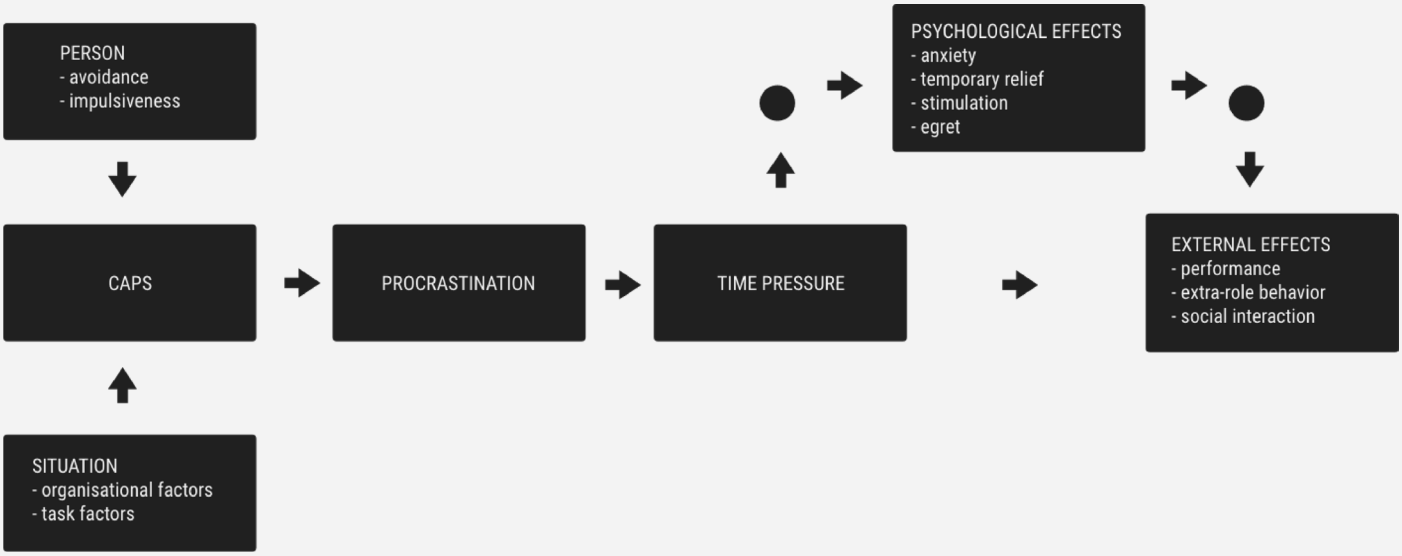


Abb. 11 nach Wendelien Van Eerde: The progress of procrastination

Das Gefühl, dass man sich über das Ergebnis einer Aufgabe im Klaren ist, ist die Hauptursache für das Aufschieben. Motivationen und äußere Umstände sollten trotzdem immer mit einbezogen werden.

Prokrastination ist also eine Gewohnheit, die durch einen emotionalen Trigger ausgelöst wird. Somit ist der wichtigste Schritt, um eine Gewohnheit zu ändern, den Trigger zu identifizieren. Auf Prokrastination bezogen bedeutet das, die Angst zu adressieren und somit die Ursache des Zögerns herauszufinden. Es hilft der vermiedenen Aufgabe mehr Bedeutung zu verleihen und den Einstiegsprozess so einfach wie möglich zu gestalten. Es ist förderlich, eine To-do-Liste zu erstellen und die Aufgabe zeitlich zu beschränken. Gut dafür geeignet ist die Pomodoro-Technik.

Pomodoro-Technik

1. Pick a timer (or an app), Set it to 20 or 25 minutes

2. Spend 20-25 minutes on a task

3. Once the timer rings, stop.

4. Take a break of 5 minutes (cycle 1)

5. Return to another 20-25 minutes on a series of tasks

6. Take a break of 5 minutes (cycle 2)

7. After 4 cycles, take a longer break

Zusammenfassend lässt sich sagen, dass sich Gewohnheiten folgendermaßen ändern lassen:

Abb. 12 nach Shukla: Breaking the habit

1. Der Trigger

muss identifiziert werden.

2. Die darauf folgenden Routine

muss angepasst werden.

3. Der alte oder ein neuer Reward

muss mit der neuen Routine in Verbindung gebracht werden.

4. Der Glaube an die Gewohnheit

muss angepasst werden.

(vgl. Shukla, 2017)

DIE LÖSUNG: FLOWLINK

Value Proposition

FlowLink ist ein Werkzeug, welches sich in den flexiblen Arbeitsalltag von Wissenschaftlern und Knowledge-Workern integriert und durch das Verknüpfen des Kontexts der Arbeitsphasen, mit dem Roten Faden, die Einstiegshürde in den Flow minimiert und somit das Leisten von schwer reproduzierbarer Arbeit verbessert.

Im Gegensatz zu herkömmlichen Selbstorganisationswerkzeugen wird beim Roten-Faden-Prinzip die komprimierte Denkmasse am Ende einer hoch konzentrierten Arbeitsphase genutzt. Ziel ist es, etwas dieser wertvollen Denkmasse abzugreifen und sie über den Draft in ein verständliches Format zu bringen, sodass der Wiedereinstieg einfach gelingt.

So inconvenient wie nötig

Ziel von FlowLink ist es über geleitetes Selbstmanagement den Flowzustand zur Gewohnheit zu machen. Hierbei wird mitunter auf den gezielten Einsatz von Inconvenient Design gesetzt.

Kleine Handlungen, die auf den ersten Blick unnötiger Anstrengung bedürfen, dienen dazu, den Interaktionen durch erhöhte Reibung mehr Bedeutung zuzuweisen und fungieren zudem als Trigger-Elemente. Somit ist das Konzept mit dem Aufbau von gängigen Ritualen und Gewohnheiten vergleichbar.

Charakter der Anwendung

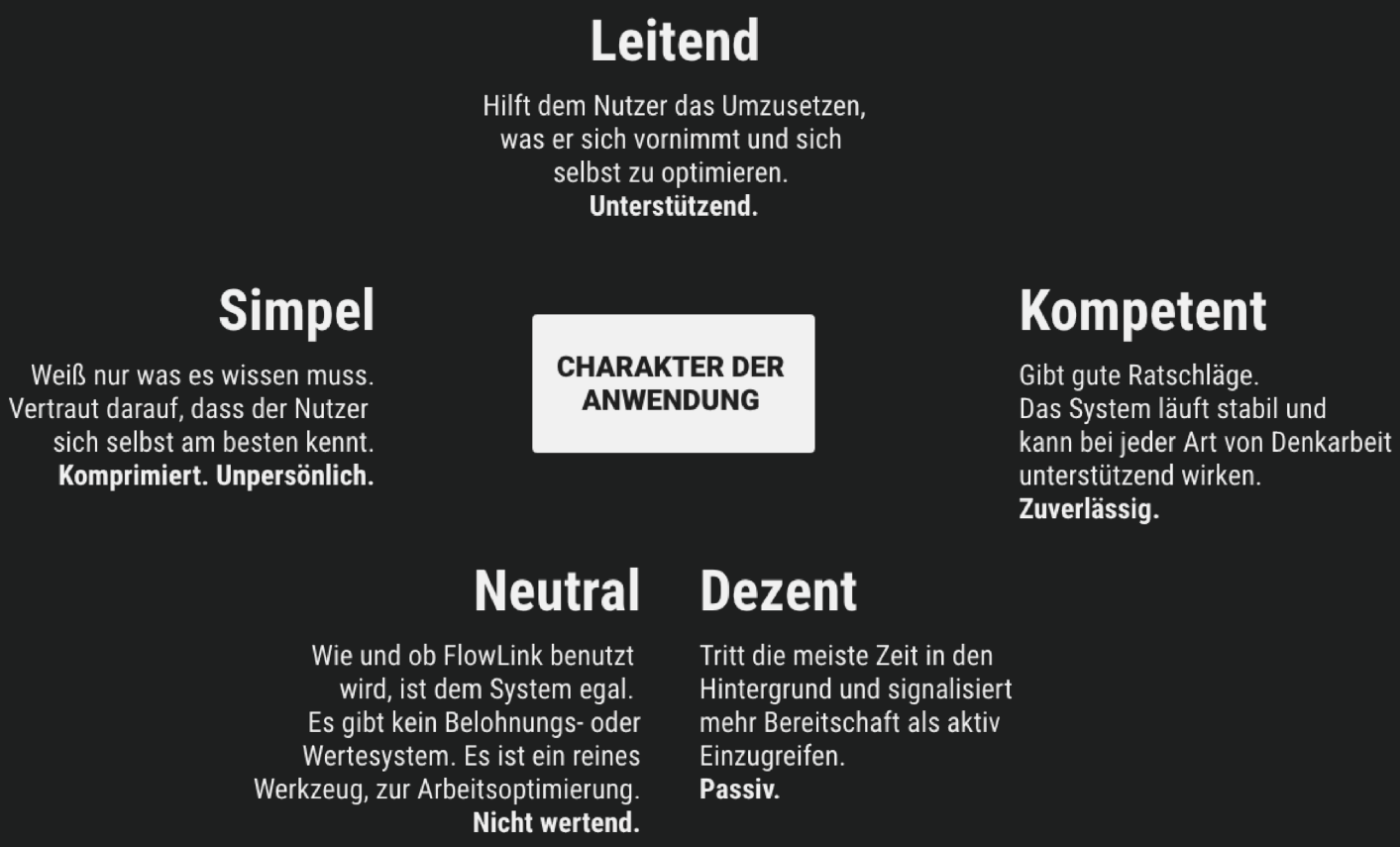


Abb. 13: Charakter der Anwendung

FlowLink ist am ehesten mit einem wohlwollenden Manager vergleichbar. Die Anwendung ist simpel gestrickt und weiß nur, was sie wissen muss, ohne dabei zu persönlich zu werden.

Sie agiert unterstützend und hilft dem Nutzer über geleitete Selbstreflexion den für sich besten Weg herauszufinden. Dies tut sie auf der einen Seite durch inconvenient Interactions an den richtigen Stellen, wiederum ist sie an anderen Stellen so direkt wie möglich.

Sie ist sehr flexibel und kommt so gut wie nie mit „der Peitsche“ um die Ecke, sondern signalisiert eher Bereitschaft, um den Nutzer an seinen eigenen Plan zu erinnern:
„die Arbeit ist bereit für dich“.

Die Anwendung ist jedoch auch sorgfältig und sorgt dafür, dass die Rituale, erst einmal vom Nutzer gestartet, auch durchgeführt werden. Darüber hinaus muss sie kompetent und zuverlässig sein, da sie als Werkzeug im täglichen Gebrauch funktionieren muss. Über „Pro-Tipps“ und die geleiteten Fragen hilft sie dem Nutzer seine Prozesse zu optimieren.

Dabei ist sie nicht urteilend, sondern konstruktiv und immer respektvoll.

Die erste Einrichtung

FlowLink kann direkt über unsere Website oder über einen plattformspezifischen App Store heruntergeladen werden. Nach dem Installationsprozess öffnet sich ein Fenster. Der Nutzer wird begrüßt. Im Anschluss folgt ein interaktives Tutorial, welches den Nutzer durch die Anwendung führt. Ziel hiervon ist es dem Nutzer das Konzept spielerisch mitzuteilen und die Vorteile, die für seine Arbeit möglich sind, zu erläutern. Es ist wichtig, dass das System hierbei anregt auszuprobieren und der Arbeitsmethode eine Chance zu geben.

Hierbei werden auch optional Accountverknüpfungen zum Beispiel mit Spotify zum Streamen von individueller Musik erstellt. Natürlich können die Verknüpfungen im Nachhinein stets angepasst werden.

Um den Einsatz von FlowLink auch in Teams, bei denen zum Beispiel Arbeitsmethoden wie Kanban oder Scrum schon eingesetzt werden, möglich zu machen, gibt es eine abgespeckte Version von FlowLink. So kann der Nutzer nach dem Tutorial wählen zwischen dem Deep-Work-Tool, welches den Timer in Kombination mit dem Roten-Faden-Prinzip beinhaltet und universell einsetzbar ist, und dem Deep-Work-Assistenten, welcher zusätzlich noch das Framing-Ritual in Kombination mit Boot-Up, Shut-Down und den verschiedenen Formen von Recap beinhaltet.

Je nach Wahl des Nutzers kann jetzt der Tag mit einem Boot-Up beginnen oder, falls der Nutzer sich bereits voll im Arbeitstag befindet, sofort eine Aufgabe gestartet werden.

Die Deep-Work-Phase: Der Rote Faden

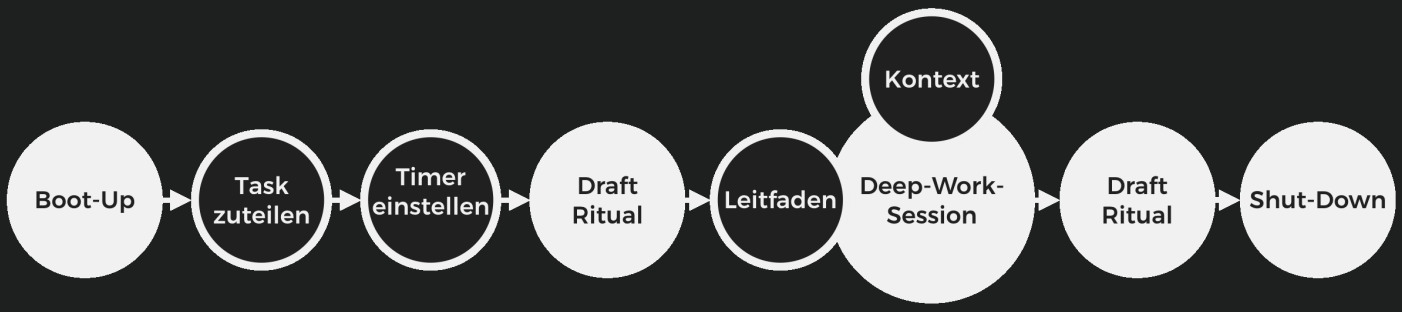


Abb. 14: Der Ablauf der Roten Fadens

Leitprinzip für Arbeit mit FlowLink ist der Rote Faden.

Die Deep-Work-Phasen und die damit verbundenen Überleitungen durch den Roten Faden sind der Kern des Konzeptes. In den Deep-Work-Phasen wird Arbeit geleistet, die kognitiv extrem fordernd ist.

Damit der Einstieg in einen neuen Deep-Work-Task strukturiert erfolgt, muss beim ersten Starten einer neuen Aufgabe der Draft ausgefüllt werden. In diesem müssen Zielsetzung, Herausforderungen und Lösungsansätze der Aufgabe definiert werden. Danach wird er immer nur am Ende der Arbeitsphasen erstellt. Gerade bei Deep-Work-Tasks ist ein strukturierter Ansatz vonnöten, um die Aufgabe erfolgreich abzuschließen.

Doch zunächst muss der bevorstehenden Phase ein Task zugeteilt werden. Der Nutzer kann hierbei aus den Projekten und Tasks wählen, welche er im Framing-Ritual für die Woche festgelegt hat. Es ist zusätzlich jederzeit möglich einen bis zu dem Zeitpunkt undefinierten Task zu starten. Dieser wird automatisch in den erstellten Plan eingebunden.

Um zu starten, ist mehr als ein einfacher Knopfdruck notwendig. Der Timer muss manuell aufgezogen werden, um die Zeitbegrenzung für die Deep-Work-Session festzulegen. Wichtig hierbei ist der Akt des Aufziehens, auch wenn es nur virtuell ist, als bewusster Start und einleitendes Trigger-Element für bedeutsame Arbeit. Damit findet sich hier ein direkter Einsatz von Inconvenient Design. Durch das Aufziehen des Timers wird unnötig scheinender Aufwand in der Interaktion betrieben, um dem Vorgang mehr Bedeutung zuzuweisen und die Wirkung des Triggers zu verstärken.

Ebenso ist es möglich, dass die Zeit für einen im Framing definierten Task gekommen ist. Dann macht sich das System automatisch bemerkbar. Der Nutzer hat nun die Option, über das Icon, die Deep-Work-Phase zu starten. Im Timer-Fenster ist bereits der geplante Task eingetragen. Der Nutzer muss noch die gewünschte Zeit einstellen und der Task beginnt. Entscheidend ist dass das System lediglich Bereitschaft signalisiert indem es den Nutzer an die Aufgabe erinnert, aber sie nicht eigenständig startet.

Wird ein neuer Task begonnen, startet die Deep-Work-Playlist. Der Draft startet das erste Mal, bevor der neue Task losgeht. Wurde er abgeschlossen oder ist bereits ein Leitfaden definiert, wird dieser jetzt angezeigt. Die im Draft gegebenen Antworten werden als Leitfaden bezeichnet. Mit dem Leitfaden wird der Nutzer zunächst seiner Arbeit überlassen. Je nach Präferenz bleibt der Leitfaden während der Session im Vordergrund und lässt sich anpassen, aber kann auch minimiert werden.

Wird ein bestehender Task fortgesetzt, werden die damit verknüpften Trigger ausgelöst, um nahtlos an dessen Kontext anzuknüpfen. Die Playlist wird an der gleichen Stelle abgespielt, an der sie beim letzten Mal pausiert wurde und es öffnen sich die gleichen Werkzeuge, die offen waren. Die Deep-Work-Playlist wird standardmäßig mit dem System installiert und ist stets anpassbar.

Es gibt zudem die Möglichkeit Spotify Premium in das System zu integrieren und so jegliche Playlists, die für geeignet empfunden werden, zu benutzen. Wenn die Deep-Work-Phase abgelaufen ist, hat der Arbeitende die Option über einen Shortcut auf der Tastatur **CMD+D+W** oder **STRG+D+W** die Deep-Work-Phase zu verlängern. Ansonsten wird das Draft-Ritual eingeleitet. Hier bereitet der Nutzer den Leitfaden vor. Zudem hat der Nutzer die Möglichkeit, einen abgeschlossenen Task mit einem neuen Task des gleichen Projektes zu verknüpfen. Falls er dies aber bereits im Framing Ritual beschlossen hat, wird hier der Draft für den verknüpften Task angezeigt. Mit Abschluss des Drafts, ist die Deep-Work-Phase beendet und die Musik stoppt.

Bei Abbruch eines Tasks, fragt das System, ob der alte Leitfaden übernommen werden soll. Antwortet der Nutzer mit „Ja“ landet er wieder im Timer-Fenster und kann in eine neue Deep-Work-Session starten. Bei einem „Nein“ wird das Draft-Ritual für diesen Task eingeleitet.

Das Interface

Das Hauptfenster der Anwendung ist das Fenster des Timers. Hier kann eine neue Aufgabe gestartet und die dazugehörige Zeit eingestellt werden. Die Kernelemente dieses Fensters sind das Aufgabenfeld, die Start-, Stop- und Pause-Buttons, sowie der Timer selbst. Deswegen ist es wichtig, dass diese Elemente im Vordergrund stehen und nebensächliche Funktionalitäten versteckt bleiben.

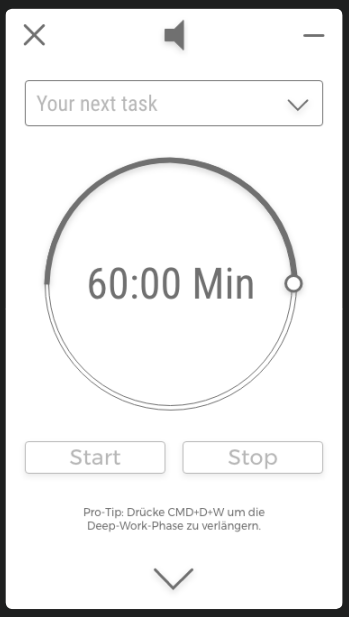


Abb. 15: Wireframe 1 Timer Start

Aus diesem Grund befindet sich am unteren Ende des Fensters auch ein Pfeil mit dem ein extra Musikmenü angezeigt werden kann. Bevor nicht auf den Pfeil gedrückt wurde, ist dieses Fenster auch nicht sichtbar. Wenn das Musikfenster ständig zusehen wäre, könnte der Nutzer die Anwendung auch als Musikplayer verstehen, doch es muss klar bleiben, dass es hierbei um Arbeit geht. Dies begründet auch das starke, visuelle Element des Timers, da dieser mit Produktivität assoziiert wird. Zudem muss der Timer eine große Fläche einnehmen, damit das Aufziehen die nötige inconvenience mitbringt.

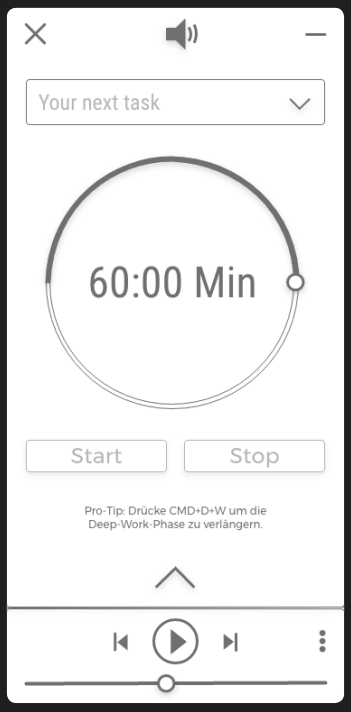


Abb. 16: Wireframe 2 Timer Musik

Wir sind uns darüber bewusst, dass so der Eindruck entstehen kann, die Zeit sei im Vordergrund unseres Systems, obwohl ja der Rote Faden und der Flow das Wichtigste sind. Nichtsdestotrotz ist die Zeit und der Zeitdruck, den sie erzeugt, ein essenzielles Werkzeug für den Flow und hat somit auch eine besondere Rolle verdient. Da sich der Timer nach dem Starten des Tasks einklappt und die Zeit somit verschwindet, kann sich so auf den Flow konzentriert werden, ohne dass die Zeit die Hauptrolle einnimmt. Durch das Einklappen hoffen wir die richtige Balance aus Fokus auf den Zeitdruck und Fokus auf den Flow zu schaffen. Wichtig ist bei dem Timer zudem, dass hier nicht eine beliebige Zeit eingestellt werden kann. Die ideale Arbeitszeit für eine Deep-Work-Session beträgt 60 bis 90 Minuten. Damit hier noch etwas Spielraum gegeben ist, wurde die maximal-einstellbare Zeit auf 120 Minuten beschränkt. Es wurde sich hier deswegen auch bewusst gegen das vorhandene Modell entschieden, das besagt, dass eine Umdrehung 60 Minuten entspricht, damit die Arbeitszeiten von optimaler Dauer sind.

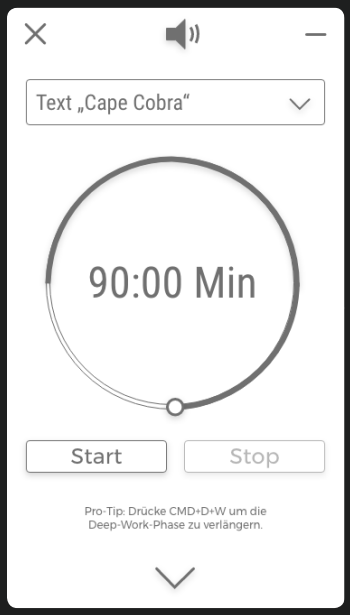


Abb. 17: Wireframe 3 Timer 90 min

Abb. 18: Wireframe 4 Timer Taskfeld

Weiteres Kernelement ist das Aufgabenfeld. Hier wird die nun zu bewältigende Aufgabe eingetragen. Damit hier keine Zeit verloren geht, ist das Feld mit einem Dropdown und Autovervollständigung versehen, damit schnell die richtige Aufgabe gestartet werden kann. Dies ist besonders wichtig, weil das System die Leitfäden unter den Aufgabennamen abspeichert. Entsteht hier also ein Rechtschreibfehler, würde das System es als neue Aufgabe einordnen. Dem kann durch die Autovervollständigung entgegengewirkt werden, weil das Eintippen einer nicht vorhandenen Aufgabe sofort dafür sorgen würde, dass der Nutzer keine Vorschläge mehr im Dropdown angezeigt bekommen würde und somit schnell klar ist, dass das System diese Aufgabe nicht kennt. Wichtig ist auch die Positionierung des Feldes. Es befindet sich über dem Timer und ist der in Reihenfolge somit ganz oben. Diese Reihenfolge wurde gewählt, um den Nutzer zu animieren, zuerst die Aufgabe und dann die Zeit zu definieren. Prinzipiell kann das System aber auch mit einer anderen Reihenfolge umgehen. Um diesen Prozess weiter zu beschleunigen, wird im Aufgabenfeld eine Condensed Schrift verwendet. Das sorgt für einen sehr kompakten Text, der schnell verdaut werden kann und zudem Platz spart und deswegen Raum für längere Aufgaben lässt. Trotzdem ist das Aufgabenfeld nur eine Zeile hoch, damit der Aufgabenname nicht ausartet. Für jegliche weiteren Details ist nämlich das Draft Fenster der richtige Ort.

Nachdem jetzt ein Aufgabenname festgelegt und eine Zeit eingestellt wurde, kann jetzt der Start-Button betätigt werden. Hier wurde sich bewusst gegen ein Icon entschieden, damit hier nicht die Verwechslung zwischen dem Starten des Timers und dem Starten der Musik stattfindet. Denn im Musikfenster, das ausgeklappt werden muss, benutzen die Steuerelemente bekannte Icons für das Starten und Stoppen von Musik.
Als essenzielles Feature in den Usertests hat sich der Mute Button erwiesen. Viele Menschen arbeiten sehr gerne ohne Musik und möchten zwar FlowLink und den Roten Faden verwenden, aber auf den Musik-Trigger verzichten.

Zudem ist Musik nicht in jedem Kontext angebracht. Aufgrund dieser Wichtigkeit befindet sich dieser Knopf direkt zentral oben im Fenster und verändert je nach Zustand seine Form. So kann auf einen Blick erkannt werden, ob das System gerade auf lautlos gestellt ist oder nicht. Dies ist wichtig, damit nicht aus Versehen Musik gespielt wird in Situationen, in denen dies nicht gewünscht ist. Dies lässt sich nur erreichen, wenn das Icon nicht versteckt ist, wie die Musiksteuerung und findet deswegen auch Platz im Hauptbildschirm.
Zudem kann man dort noch einen Schließen und Minimieren Button finden. Diese sind aber ausgegraut, damit sie nicht zur Verwendung anregen. Je nach Betriebssystem ändert sich die Seite der beiden Buttons, um auf Gewohnheiten des jeweiligen Systems zurückzugreifen.

Das Draft-Ritual

Ziel des Draft-Rituals ist es, den Leitfaden vorzubereiten, um die Einstiegshürde für die kommende Deep-Work-Phase zu minimieren und eine erste, nicht zu schwierige Aufgabe für den Einstieg in den Flow-Zustand zu schaffen.

Zudem schafft sich der Nutzer mit dem Leitfaden eine Art Rahmen für die kommende Aufgabe. Falls einmal der Faden verloren geht, kann er durch den Leitfaden schnell wiedergefunden werden.

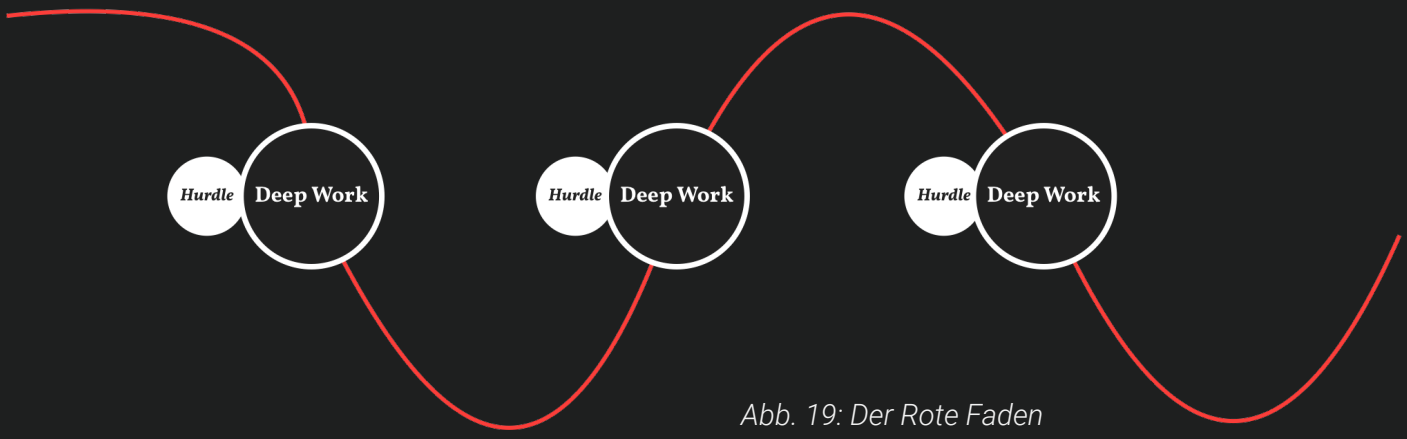


Abb. 19: Der Rote Faden

Der Rote Faden dient dazu, verschiedene Deep-Work-Phasen des meist selben oder verschiedener, aber ähnlicher, Tasks miteinander zu verknüpfen. Der Grundgedanke besteht darin, die komprimierte Denkmasse aus der Deep-Work-Session zu nutzen und in den letzten fünf Minuten des Tasks zu verwenden, um während man noch im Kontext der Aufgabe steckt, einen Leitfaden für das nächste Mal vorzubereiten.

Dieser wird beim Fortsetzen des Tasks wieder übergeben. So wird die Denkarbeit, welche beim Wiedereinstieg in die Aufgabe möglicherweise eine halbe Stunde kosten würde, am Ende des Tasks auf fünf Minuten komprimiert.

Das System leitet den Nutzer an für sich drei Dinge festzuhalten:

1. **Die Motivation.** Das Taskziel. Der Schatz. *Why*
2. **Das Problem.** Der Bösewicht. *What*
3. **Der Lösungsansatz.** Das Schwert. *How*

Dabei wird sich einerseits an der klassischen Heldenreise orientiert. Der Wissenschaftler ist hierbei der Held, welcher seine eigene im Draft-Ritual festgehaltene Motivation als Waffe hat. Jedes Draft-Ritual ist eine kleine Reise für sich und hat eine eigene Überschrift, seinen eigenen Bösewicht, aber den gleichen Helden.

Der Held wird somit vor der Höhle des Löwen stehen gelassen, jedoch mit dem selbst gegebenen Lösungsansatz, dem Schwert, um das nächste Mal den Bösewicht mit voller Stärke angreifen zu können. Der ganze Prozess dient dazu, den Leitfaden, welcher beim nächsten Start des Tasks angezeigt wird, vorzubereiten. Die Belohnung ist das Erreichen des Taskziels.

Abb. 20: Unsere Version der Heldenreise

Logisch betrachtet werden emotionale Aspekte mit dem Kontext verknüpft und somit der Trigger, welcher zu Prokrastination führen könnte direkt adressiert. Die klassischen Fragen: “Why”, “What” und “How” werden verwendet, um den Kontext zu bestimmen.

Wichtig hierbei ist die Formulierung und auch die Reihenfolge der Fragen. Diese sind nicht als Fragen formuliert, sondern direkte Anforderungen an den Nutzer um deutlich zu machen, dass diese Felder auszufüllen sind:

1. Explain your motivation to complete this task.

Es ist wichtig, mit einer einfachen Frage einzusteigen, welche Spaß macht zu beantworten, um den Nutzer in das Draft-Ritual einzuführen und nicht direkt zu überfordern. Auch hier findet das Flow-Prinzip Anwendung, denn eine der Anforderungen für Flow nach Csikszentmihályi ist ein klar definiertes Ziel. Diese Frage ist extrem wichtig, da sie auch direkt eine emotionale Bindung zwischen Task und Nutzer aufbaut und die Relevanz der Aufgabe verdeutlicht. Zudem bildet die Motivation die Grundlage für das Erfüllen jeder Aufgabe und damit auch die Basis für die Beantwortung der folgenden Punkte.

2. List possible pain points.

Etwas schwerer zu formulieren, aber leicht zu identifizieren, sind die Stellen der Aufgabe, welche Schwierigkeiten bereiten werden. Dadurch dass der Nutzer sich frühzeitig damit beschäftigt, wird er nicht überrumpelt und kann, kalkuliert und sich der Schwierigkeit der zu bearbeitenden Aufgabe bewusst, in den Flow-Zustand eintreten. Die „Pain-Points“ dienen des Weiteren als „Cliffhanger“ an unbeantworteten Fragen, die dazu leiten sollen, dass der Nutzer seine Lösungsansätze testen möchte.

3. List possible approaches.

Sie ist die am schwersten zu beantwortende Frage, welche jedoch auf Frage eins und zwei aufbaut und den runden Abschluss für den Leitfaden liefert. Sich einfach nur mit der richtigen Motivation auf die Probleme zu stürzen, könnte eine zu große Einstiegshürde sein. Deswegen wird der Nutzer dazu geleitet sich eine Einstiegsrampe zu bauen und mögliche Lösungsansätze zu notieren. Zusätzlich schützt sich der Nutzer so davor, lange Zeit mit einem schlechten Lösungsansatz zu verschwenden.

Sind diese drei Punkte festgehalten, speichert das System automatisch alle verfügbaren Trigger ab. Es merkt sich, welcher Song gerade abgespielt wird und sekundengenau an welcher Stelle dieser sich befindet. Danach wird die Deep-Work-Phase beendet und das System tritt wieder in den Hintergrund.

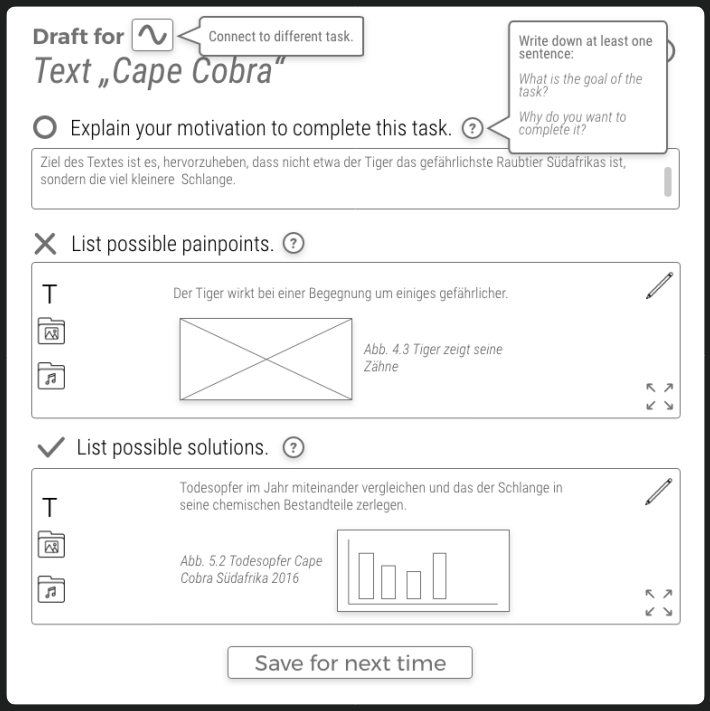


Abb. 21: Wireframe Draft

Der Leitfaden

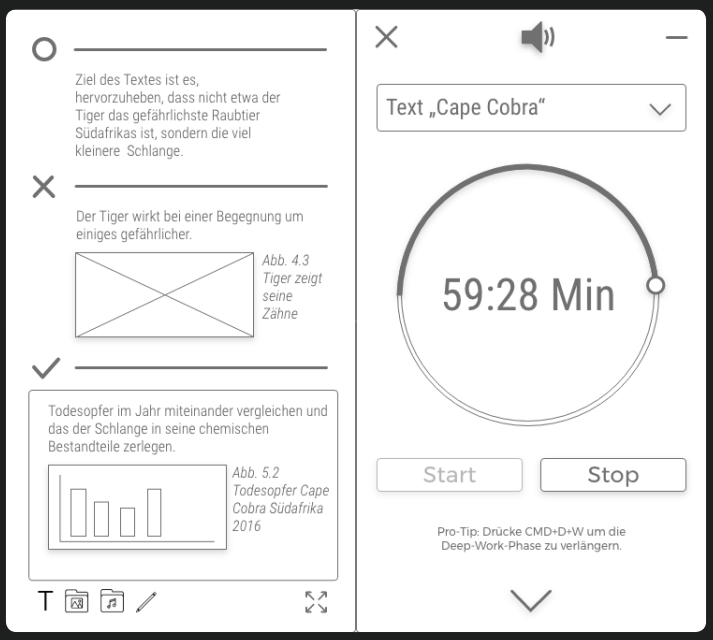


Abb. 22: Wireframe Leitfaden

Beim Starten einer mit Leitfaden versehenen Aufgabe öffnet sich ein Fenster mit den im Draft gegebenen Antworten. Dieses ist permanent verfügbar und dient während der Deep-Work-Phase wortwörtlich als Leitfaden, um den bestehenden Task zu vollenden. Ziel des Leitfadens ist es, den Bogen vom Nutzer, über das Problem, hin zum Taskziel zu spannen. Die Problemstellung und das Taskziel wurden beim letzten Draft-Ritual im Anschluss an diesen Task oder beim ersten Anlegen des Tasks festgelegt. Das Taskziel ist nicht mehr bearbeitbar. Das Solution- und das Problem-Feld können noch bearbeitet werden. Sie sind einerseits die Einstiegsrampe, um die Hürde in die nächste Session zu erleichtern, aber dienen auch als Notizfeld, um sich auf dem Weg weitere Lösungen zu notieren. Hier werden Hyper-Media-Windows benutzt. So kann der Nutzer auf die Art und Weise, die ihm am besten liegt, multimediale Notizen machen und so den für ihn optimalen Lösungsweg festhalten. Es ist also möglich schriftliche Notizen mit hilfreichen Grafiken oder Bildern zu erweitern oder sogar kurze Audionachrichten oder Sounds einzubetten.

Der Rahmen

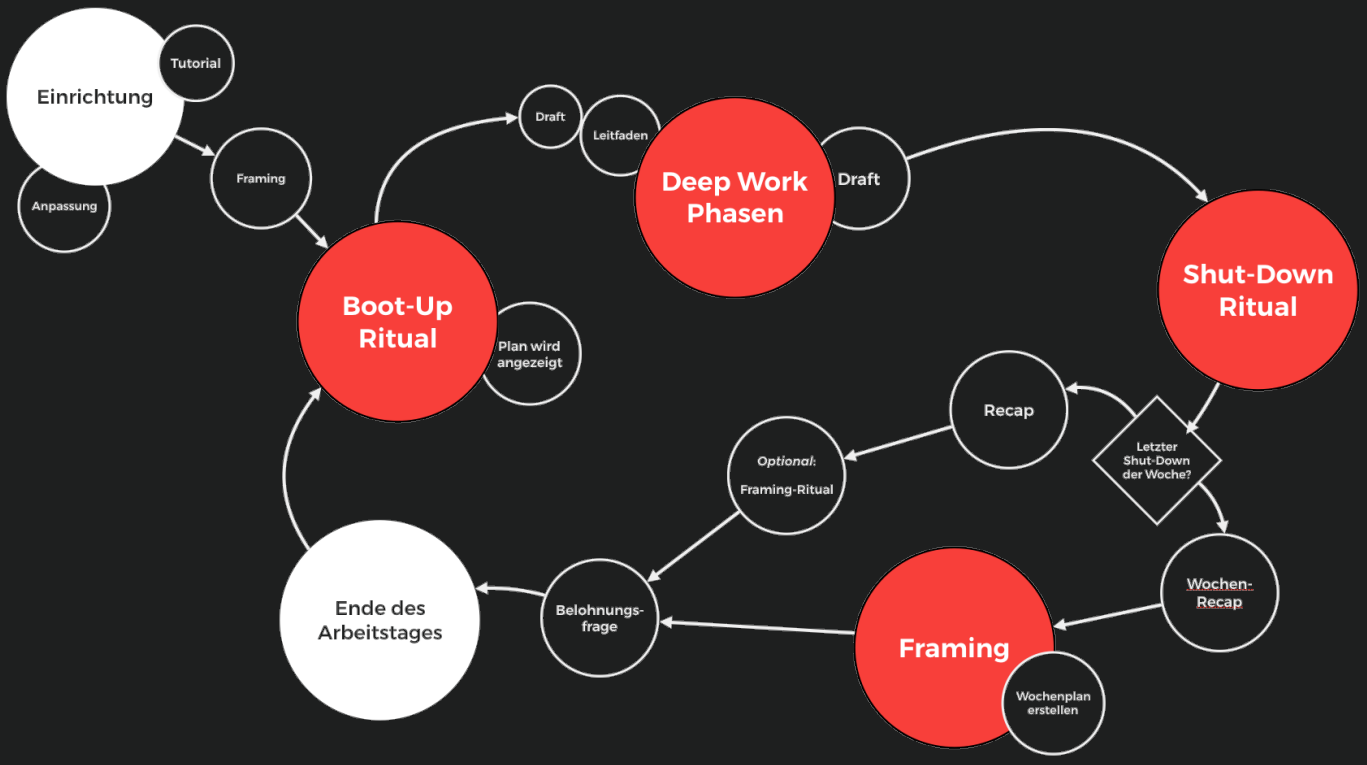


Abb. 23: Ablauf der Rahmen

Aus den Interviews mit der Zielgruppe hat sich ergeben, dass die Tages- und Wochengestaltung sehr variabel und projektbezogen abläuft. Deswegen ist eines der Kernprinzipien von FlowLink die Flexibilität, um den Arbeitenden nicht unnötig einzuschränken.

Aus der Recherche hat sich aber ergeben, dass ein Minimum an Struktur dabei hilft Deep-Work zur Gewohnheit zu machen und Prokrastination vorzubeugen. Deswegen bietet das System einen Rahmen, der Struktur bieten soll, aber stets angepasst werden kann.

FlowLink kommt in Form einer Software. Diese ist eine plattformübergreifende Anwendung für Windows, MacOS und Linux. Die Anwendung läuft hierbei im Hintergrund und macht sich nur zu eingestellten Zeiten bemerkbar oder auf Anfrage des Nutzers. Es wurde sich bewusst gegen die Verwendung auf dem Smartphone entschieden, da dieses während den Arbeitsphasen im besten Fall im Flugmodus ist und zu dessen Verwendung nicht animiert werden soll.

Im Alltag startet das System mit dem Boot-Up und endet mit dem Shut-Down-Ritual. Die Zeiten für diese Rituale werden vom Nutzer selbst festgelegt und können stets angepasst werden.

Das Boot-Up symbolisiert dabei den Start des Arbeitstages und das Shut-Down beendet diesen. Dazwischen liegen die selbst eingeteilten Deep-Work-Phasen, welche über den Roten Faden miteinander verknüpft werden.

Da FlowLink als ein unterstützendes Arbeitswerkzeug dienen soll, ist das System auch offline funktionsfähig.

Das Boot-Up-Ritual

Wann das Boot-Up stattfindet, wird im Framing-Ritual festgelegt und dieses kann zu jeder Zeit bearbeitet werden.

Wenn der eingestellte Zeitpunkt erreicht ist, macht sich das System über die Taskleiste oder Menübar bemerkbar. Diese fangen an in gelb-hellblauen Farben zu pulsieren. Damit soll das mentale Modell eines Sonnenaufgangs angesprochen werden. Das System signalisiert dem Nutzer dezent, dass der Arbeitstag nach eigener Planung jetzt losgehen kann: „Die Arbeit ist vorbereitet“. Der simulierte Sonnenaufgang soll sowohl den Nutzer willkommen heißen, als auch Wertschätzung gegenüber der Arbeit vermitteln.

Das eigentliche Boot-Up beginnt jedoch erst, wenn der Nutzer auf das nun leuchtende Icon von FlowLink klickt.

Ein Fenster öffnet sich, zusammen mit dem charakteristischen Boot-Up-Sound, welcher als Trigger-Element für den Start des Arbeitstages dient. Eine Übersicht für den Arbeitstag, mit Boot-Up- und Shut-Down-Zeitpunkt, die eingetragene Shallow-Work und die geplanten Deep-Work-Tasks für den Tag, verknüpft mit den jeweiligen Roten Fäden, werden angezeigt. Es gibt die Option den Wochenplan noch mal einzusehen indem man das Framing-Fenster öffnet und somit das Framing-Ritual erneut durchführen kann. Unterhalb der Übersicht befindet sich der Vorsatz, welchen sich der Nutzer im Recap des Vortages geschrieben hat. Er soll so erst eine Übersicht über den Tag bekommen und danach noch an seinen eigenen Vorsatz erinnert werden, um diesen auch umsetzen zu können.

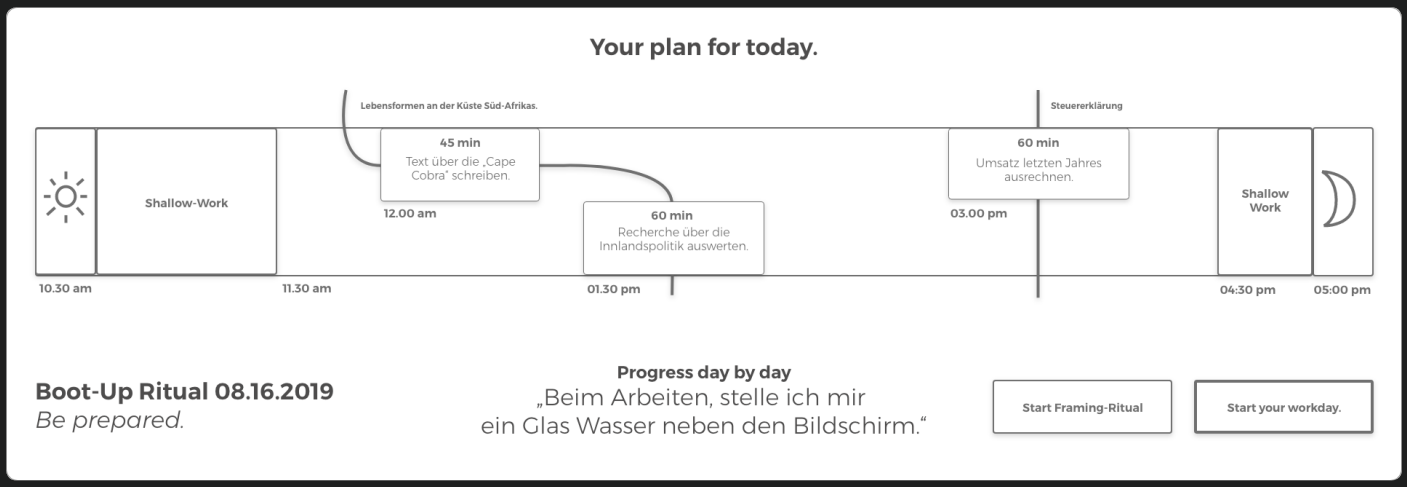


Abb. 24: Wireframe Boot-Up

Mit Bestätigung des Tagesplans tritt die Anwendung in den Hintergrund, ist jedoch jederzeit wieder abrufbar und bleibt über ein kleines Icon am Bildschirmrand präsent.

Ist ein geplanter Zeitpunkt erreicht, macht sich das System dezent bemerkbar, um das Rote-Faden-Prinzip zu starten und somit eine Deep-Work-Phase einzuleiten. Das geschieht über ein kleines Pop-up am oberen Bildschirmrand. Es geht auch hierbei wieder darum, den Nutzer an seinen eigenen Plan zu erinnern und Bereitschaft des Systems zu signalisieren.

Das Shut-Down-Ritual

Ist der im Framing-Ritual festgelegte Zeitpunkt für das Shut-Down erreicht, macht sich das System zunächst unauffällig bemerkbar. Wie beim Boot-Up-Ritual wird auch hier die Sonne simuliert. Da das Shut-Down, das Ende des Arbeitstages symbolisiert, wird hier der Sonnenuntergang in Orange-, Rot- und Blautönen dargestellt. Tippt man das Wort „Shut-Down“ in das Task-Feld ein, wird das Ritual eingeleitet. Der Interaktionsvorgang ist mit einem Befehl im Terminal vergleichbar.

Ist der eingestellte Zeitpunkt bereits überschritten, ist das Wort Shut-Down eine voreingestellte Option in dem Dropdown für die gespeicherten Tasks und kann direkt ausgewählt werden, muss jedoch noch mal eingetippt werden. Das dient als eine Art Sicherung, damit das Shut-Down nicht aus Versehen eingeleitet werden kann. Diese Inconvenience ist notwendig, damit das Shut-Down, vom Gehirn, als eindeutiger Trigger für das Ende des Arbeitstages abgespeichert werden kann. Das spielt sich alles im klassischen Timer-Fenster ab, nur dass der Start-Button sich nun in einen „Shut-Down einleiten“ Button verändert hat.

Ziel des Shut-Downs ist es durch ein Ritual den Arbeitstag zu beenden und somit einen klaren Kontextwechsel einzuführen. Der durch die Arbeit entstandene Stress soll hinter sich gelassen werden. Daher ist der konkrete Akt des Beendens durch den Nutzer essenziell.

Des Weiteren findet das Recap für den Tag statt. Hier wird über den Tag und mögliche Verbesserungen reflektiert. Im Anschluss gibt es die Möglichkeit den Plan für den Rest der Woche, über das Framing-Ritual, wenn man das möchte, anzupassen.

Nach dem Shutdown fährt das Programm in die Menüleiste zurück und das Symbol verliert an Deckkraft (es wechselt von Weiß zu Grau) und tritt somit in den Hintergrund.

Es kann jedoch noch weiterhin genutzt werden. Steht dem Nutzer der Sinn nach einer spontanen Deep-Work-Session nach dem Shut-Down-Ritual, dann ist das möglich, wenn auch nicht gewollt. Klickt er auf das Icon, färbt es sich wieder weiß und es öffnet sich die Kontrollfrage: „Bist du sicher, dass du den bereits abgeschlossenen Arbeitstag um diese Deep-Work-Session verlängern möchtest?“.

Drückt der Nutzer auf ja, öffnet sich das Timer-Fenster und die Session kann ganz normal gestartet werden. Nach Beenden des Tasks signalisiert das System wieder Shut-Down-Bereitschaft und kann klassisch über das Taskfeld beendet werden.

Als weitere Option gibt es die Möglichkeit direkt das kommende Boot-Up einzuleiten, falls der Nutzer früher als geplant das Bedürfnis hat den Arbeitstag zu beginnen. Hierbei drückt er bei der Kontrollfrage statt auf „Ja“ auf „neuen Arbeitstag beginnen“, um das Boot-Up einzuleiten.

Das Recap

Das tägliche Recap ist wichtig für die Optimierung des Arbeitsprozesses und der eigenen Leitfäden durch den Nutzer. Es findet immer als Teil des Shut-Down Rituals statt. Hier wird der Nutzer zu seinem eigenen Manager und reflektiert über sein Geleistetes. Er bekommt die Möglichkeit aus der nahen Deep-Work-Ebene und einzelnen Tasks einen Schritt zurückzutreten und das größere Bild zu betrachten. Ein Prozess, der hier als Zoom-Out bezeichnet wird. Wichtig hierbei ist, dass der Arbeitende sein Geleistetes selbst bewertet. So gibt es drei Fragen, die zur Selbstreflexion über die eigene Arbeitsweise dienen. Ein weiterer Kernpunkt des Recaps ist die Optimierung des Leitfadens. Der Nutzer soll sich selbst die Fragen stellen:

1. Was lief heute gut?
2. Was würdest du verbessern?
3. Was wirst du morgen besser machen?

Elementar ist, dass die Fragen konstruktiv formuliert sind und den Nutzer nicht in Bedrängnis stellen sich rechtfertigen zu wollen. Das System wertet nicht und speichert die Antworten einfach nur ab, ohne sie weiterzuverarbeiten. Sie dienen hauptsächlich für den Arbeitenden selbst und es geht vor allem darum, dass sich der Nutzer mit dem Fragen persönlich auseinandersetzt. Eine Ausnahme ist Frage Drei. Deren Antwort ist beim nächsten Boot-Up-Ritual gut sichtbar als Erinnerung für den Tag zu lesen. So sind die Fragen Eins und Zwei über ein Hyper-Media-Window, welches nahezu die volle Breite des Fensters einnimmt, miteinander verknüpft. Ein Hyper-Media-Window bietet dem Nutzer die Möglichkeit, seine Antwort frei zu formulieren.

Die Idee hinter Hyper-Media ist der natürlichen Struktur des Gehirns freien Lauf zu lassen (Borsook, 2010). Der Nutzer kann darin schreiben, Audio-Files und Bilddokumente abspeichern und Zeichnen. So ist es ihm möglich, simpel in Text zu antworten oder komplexe Multimedia-Diagramme und Kontexte zu erstellen. Dies ist vor allem wegen des abwechslungsreichen Arbeitsalltags der Zielgruppe wichtig und passt sich so jeglichen Bedürfnissen der Dokumentation an.

Um das Recap zu beenden, drückt man auf „Arbeitstag beenden“.

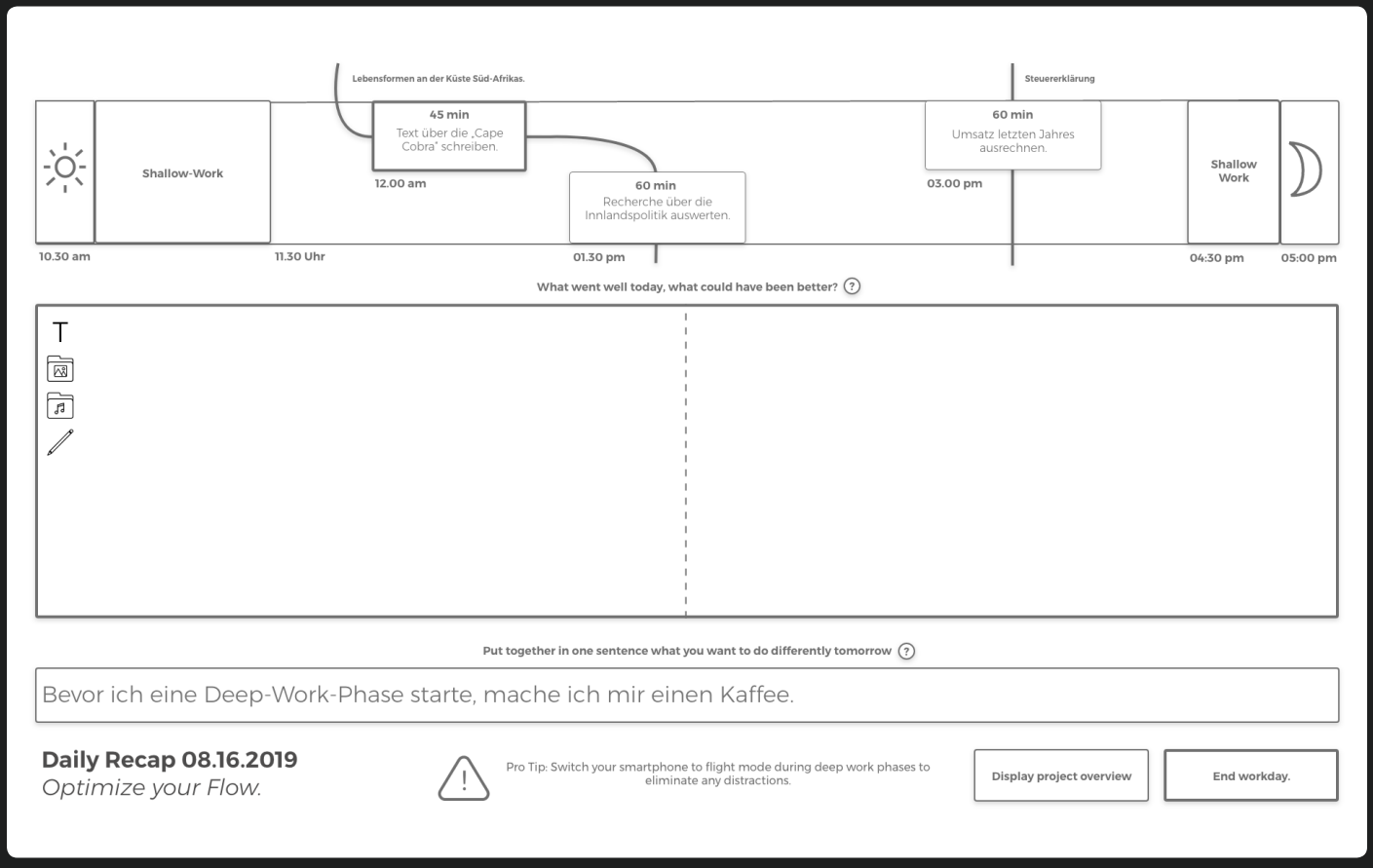


Abb. 25: Wireframe 1 Recap

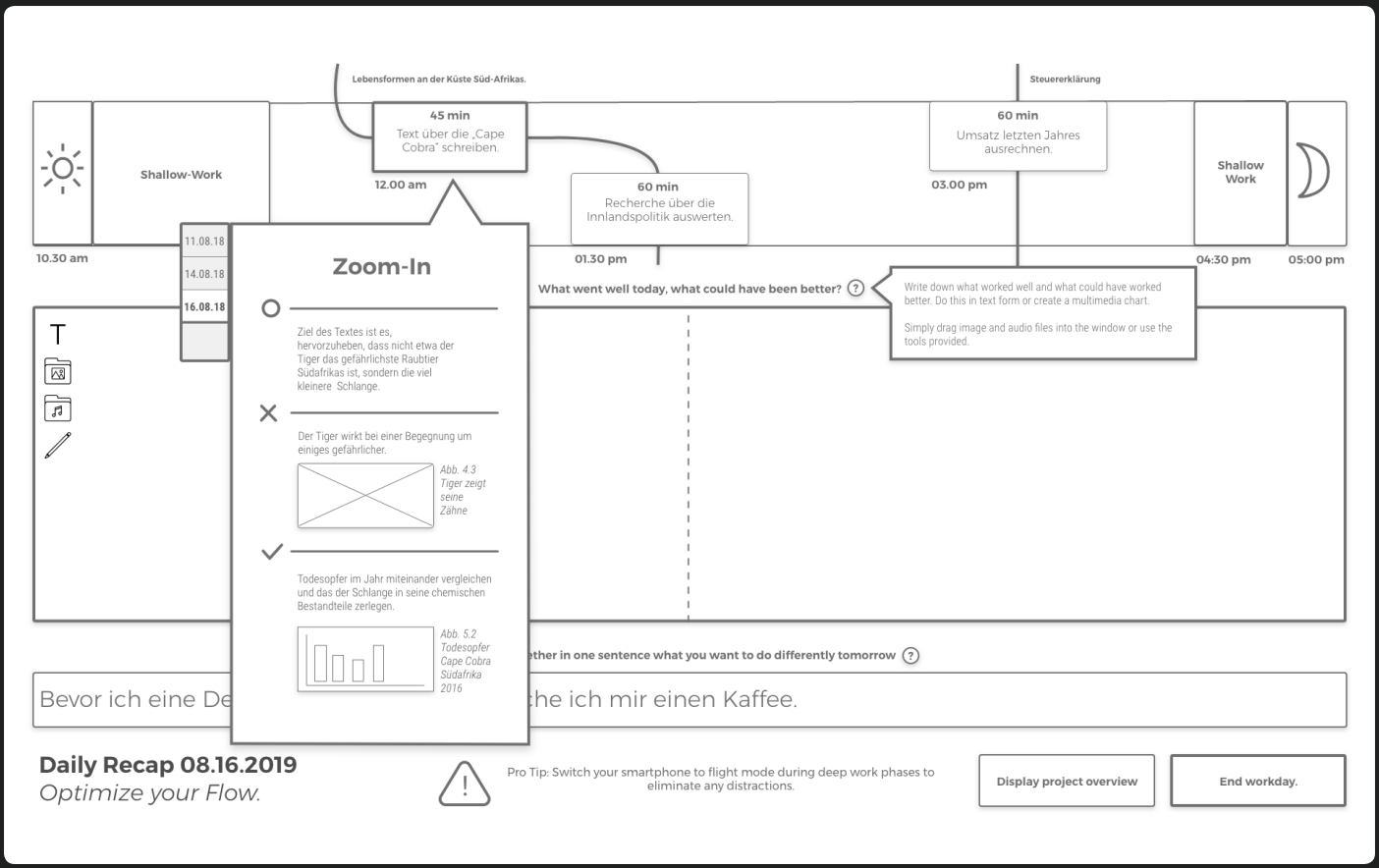


Abb. 26: Wireframe 2 Recap

Es gibt immer die Option sich eine Übersicht über die vorhandenen Projekte anzeigen zu lassen. Der Nutzer sieht hier, wie viele Tasks er in jedem Projekt schon hat und wie viele Deep-Work-Sessions, mit exakter Länge, jeder Task besitzt. Des Weiteren bekommt er, wenn er auf die Tasks klickt, die jeweiligen Leitfäden angezeigt, plus die Verbesserungsvorschläge, welche er zu den jeweiligen Leitfäden mal geäußert hat. Der Nutzer kann so das Projekt noch mal Revue passieren lassen und auch vergleichen. Bei einem ähnlichen Task kann er sich so zurückerinnern, wie er den damaligen Task gelöst hat. Diese Ansicht ist jedoch nur im Recap, also als Teil des Shut-Down-Rituals und im Framing möglich. Die Inconvenience ist notwendig, um dem „Instagram-Syndrom" vorzubeugen. Dabei geht es darum, ein ständiges Überprüfen des Projektstandes unmöglich zu machen, um hier einem Suchtfaktor entgegenzuwirken. Wie auch die Shallow-Work ihren festen Platz im Arbeitstag hat, so ist das Ende des Arbeitstags und die Planung der nächsten Woche für die Projektübersicht gedacht.

Das Wochen-Recap

Am letzten Arbeitstag der Woche gibt es statt dem normalen Recap beim Shut-Down-Ritual, ein Wochen-Recap.
Der Nutzer bekommt eine Übersicht über alle Arbeitstage der Woche und kann die Leitfäden der jeweiligen Tasks begutachten.
Man kann so für sich evaluieren, wie die Woche gelaufen ist. Der ganze Prozess dient auch hier wieder zur Selbstreflexion des Nutzers und zur Optimierung der Leitfäden. Im Vergleich zum Daily-Recap soll hier über die komplette Woche reflektiert werden. Zudem muss ein Leitsatz definiert werden für die kommende Woche, der Verbesserungsvorschläge für die eigene Arbeitsweise enthält. Ziel des Wochen-Recaps ist es einen entfernteren Blick auf die eigenen Aufgaben und Projekte zu erreichen als dies beim Daily-Recap möglich ist. Das System verarbeitet die angegebenen Daten nicht weiter. Im Gegensatz zum Daily-Recap, lässt sich hier die Planung der kommenden Woche nicht vermeiden. Man wird direkt zum Framing der nächsten Woche weiter geleitet.

Dasselbe Prinzip lässt sich anwenden auf Monats- oder sogar Jahres-Recaps. Da diese einem ähnlichen Vorgehen folgen, werden sie hier nicht weiter ausgeführt.

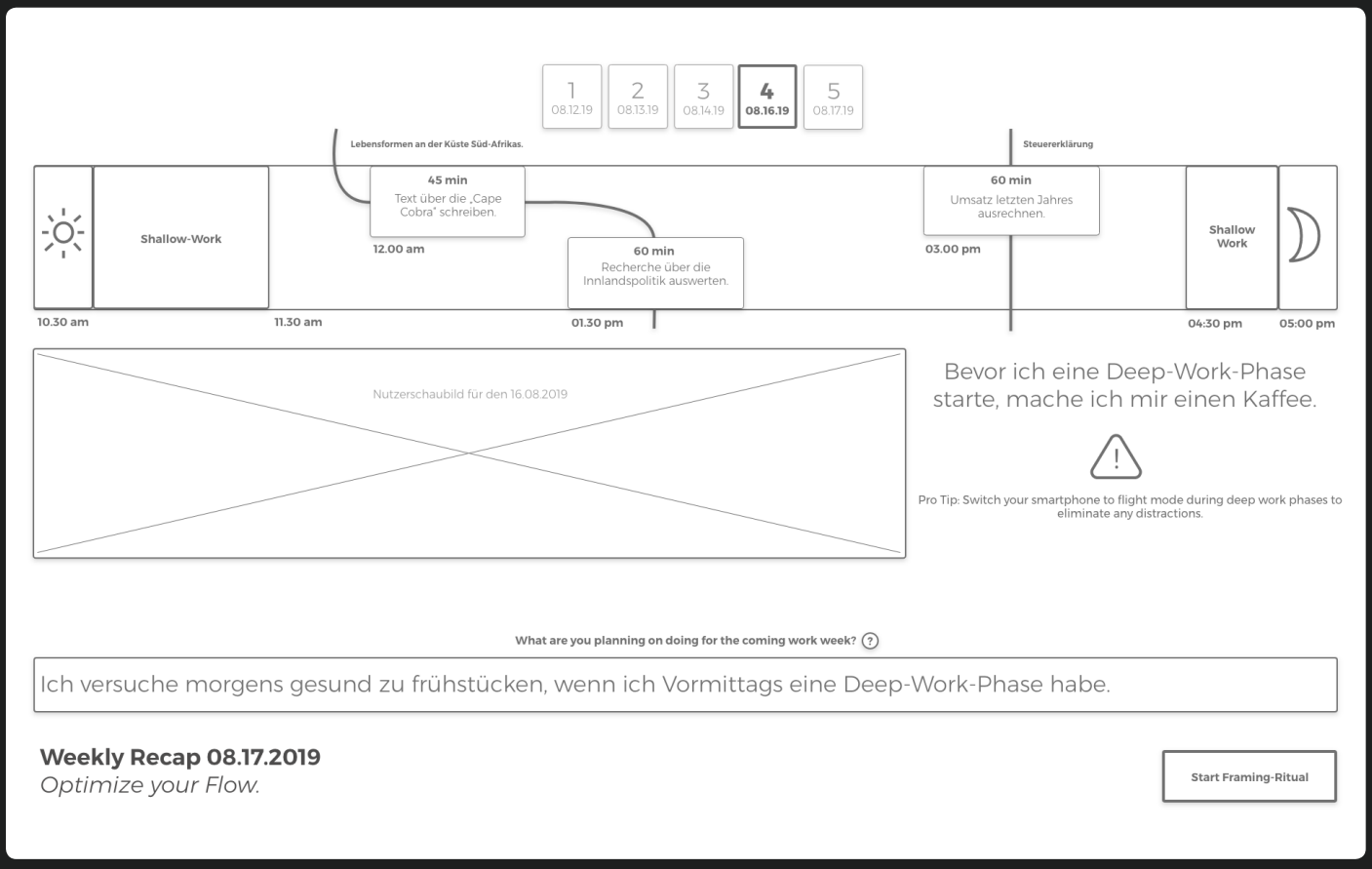


Abb. 27: Wireframe 3 Weekly-Recap

Das Framing-Ritual

Das Framing-Ritual dient dazu, die Deep-Work-Phasen für die kommende Woche einzuplanen. Es findet im Anschluss an das Wochen-Recap statt, ist jedoch in jedem Shut-Down-Ritual anpassbar. Die bestehenden Tasks und Projekte werden aufgelistet und können auf die Tage der Arbeitswoche verteilt werden. Zudem leitet das System dazu an Shallow-Work Phasen zu definieren, die entweder direkt nach dem Boot-Up oder kurz vor dem Shut-Down stattfinden sollten. Ziel davon ist es zunächst Shallow-Work, wie das Abarbeiten von E-Mails abzuhaken, damit sich auf die Deep-Work konzentriert werden kann. Zudem bietet die Shallow-Work einen einfachen Start in den Tag und minimiert deswegen anfängliche Prokrastination.
Weiter wird, um Prokrastination vorzubeugen, dazu aufgefordert den einzelnen Tasks eine geplante Länge und einen Tag mit Startzeit zuzuordnen. Nun wird auch festgelegt, wann das Boot-Up für den Start der nächsten Arbeitswoche stattfinden soll. Diese Zeit überträgt sich automatisch auf alle Arbeitstage der Woche, kann jedoch bei jedem Shut-Down-Ritual für den nächsten und kommende Tag(e) angepasst werden. Das Gleiche gilt für die Planung des Zeitpunktes für das Shut-Down-Ritual selbst.

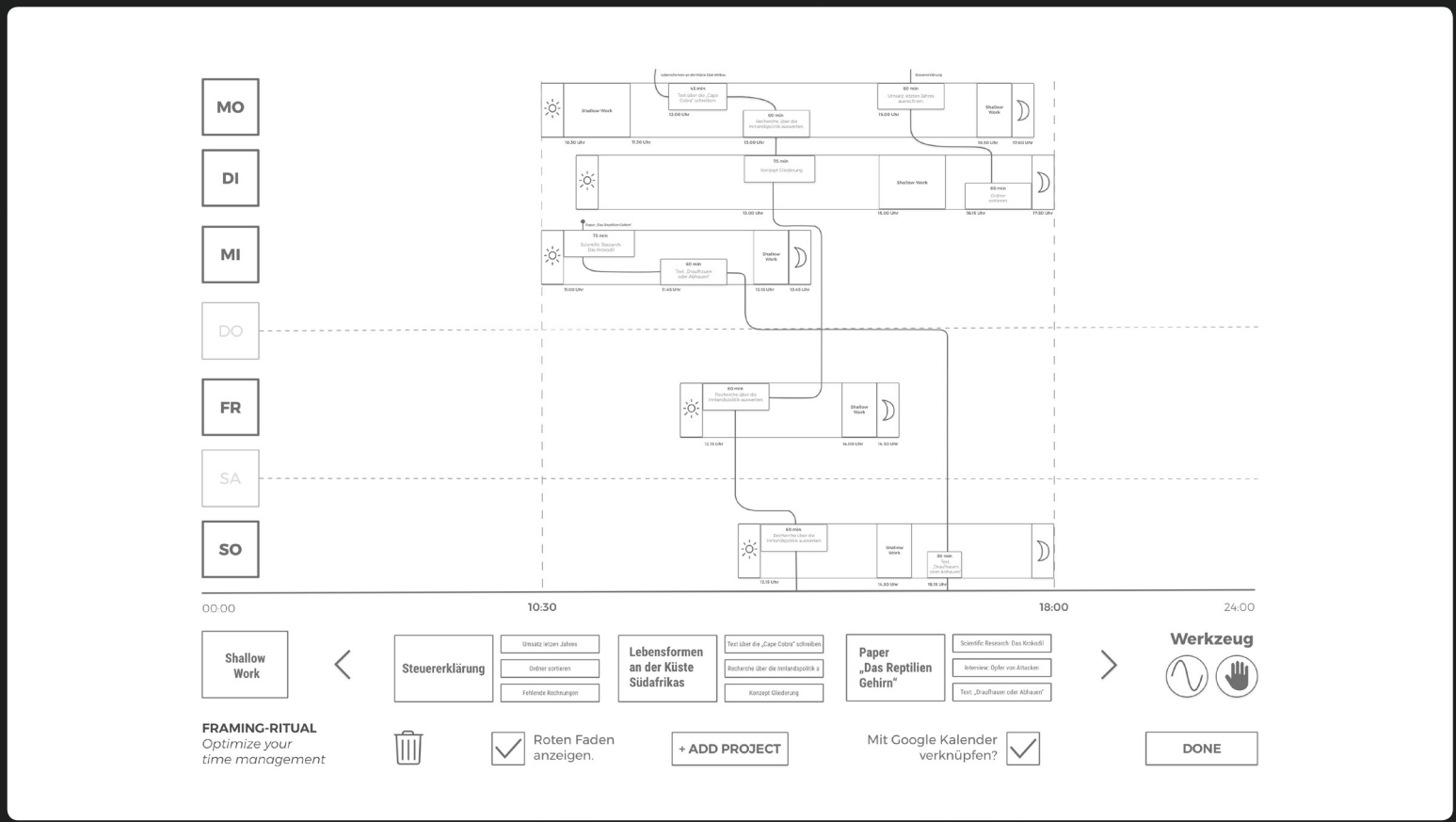


Abb. 28: Wireframe 1 Framing

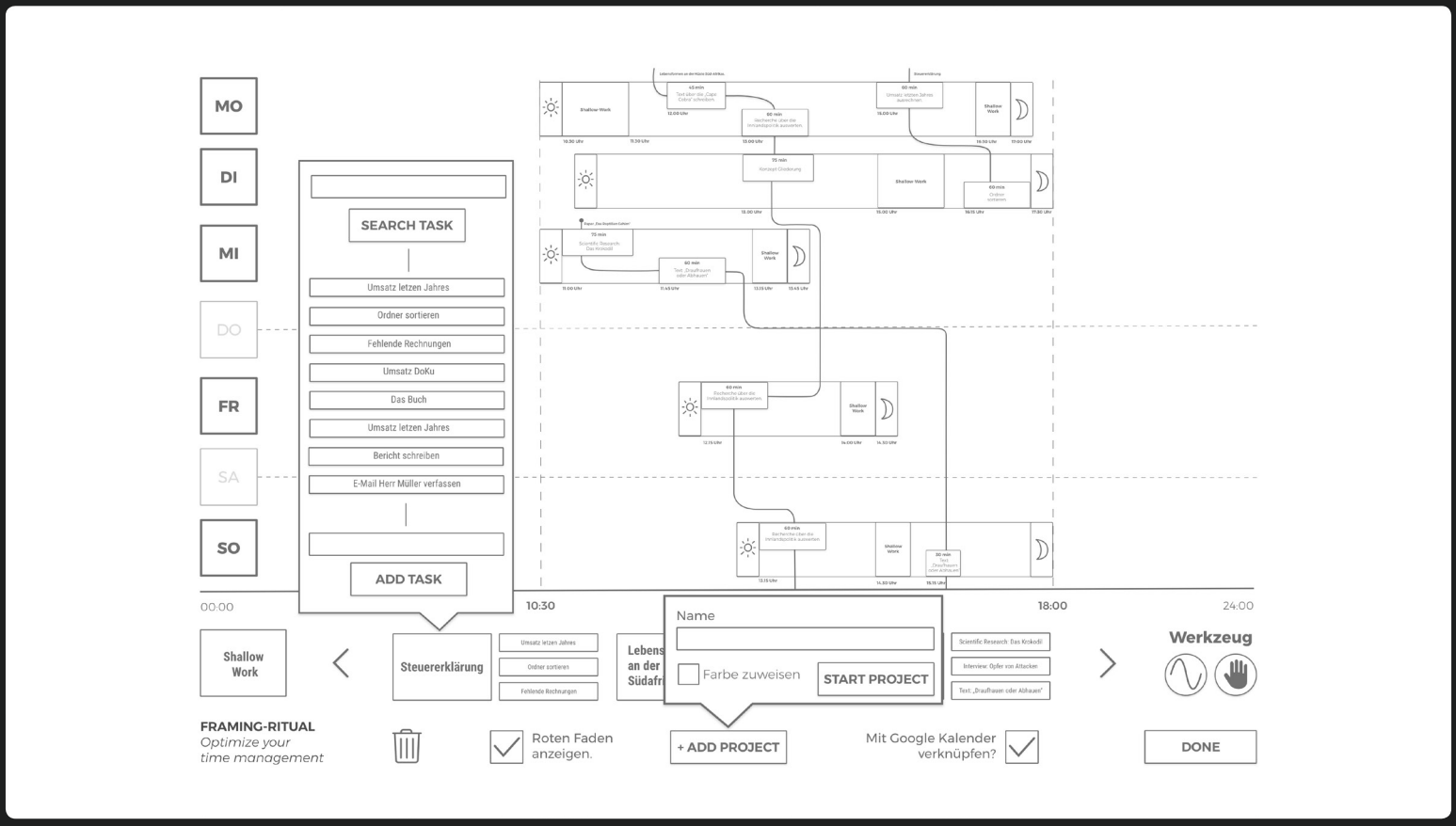


Abb. 29: Wireframe 2 Framing

Anschließend ist es zudem möglich unterschiedliche Aufgaben mit dem Roten Faden zu verknüpfen. Dies macht es möglich die Vorteile des Rote-Faden-Prinzips so auch bei der Überleitung verschiedener Aufgaben zu nutzen.

Wenn zum Beispiel an einem Tag mehrere Texte des gleichen Projekts geschrieben werden müssen, können diese Tasks miteinander verknüpft werden und so wird nach Abschluss des ersten Textes ein Leitfaden für den Folgenden vorbereitet. Für das System sind das zwei verschiedene Aufgaben, aber da diese inhaltlich ineinander überleiten, kann so auch der Einstieg in eine neue Aufgabe erleichtert werden.

Das Verbinden von Aufgaben ist auch immer spontan möglich, aber falls die Aufgaben bereits im Framing Ritual verbunden wurden, hat das System sich dies gemerkt und öffnet automatisch das passende Draft-Fenster. Natürlich kann die Verknüpfung auch im Nachhinein aufgehoben werden, falls die erste Aufgabe zum Beispiel nicht abgeschlossen wurde. Da hier der Plan für die komplette Woche erstellt wird, kann es vorkommen, dass mehrere Stränge des Roten Fadens existieren. Diese werden hierbei farblich unterschieden.

Das Framing ist in vier Schritte eingeteilt:

1. Die Arbeitstage der kommenden Woche zuordnen

2. Bevorstehende Projekte benennen und Tasks zuteilen

3. Boot-Up und Shut-Down festlegen und die Tasks in die passenden Timeslots ziehen

4. Über das Rote-Faden-Tool die passenden Tasks miteinander verknüpfen.

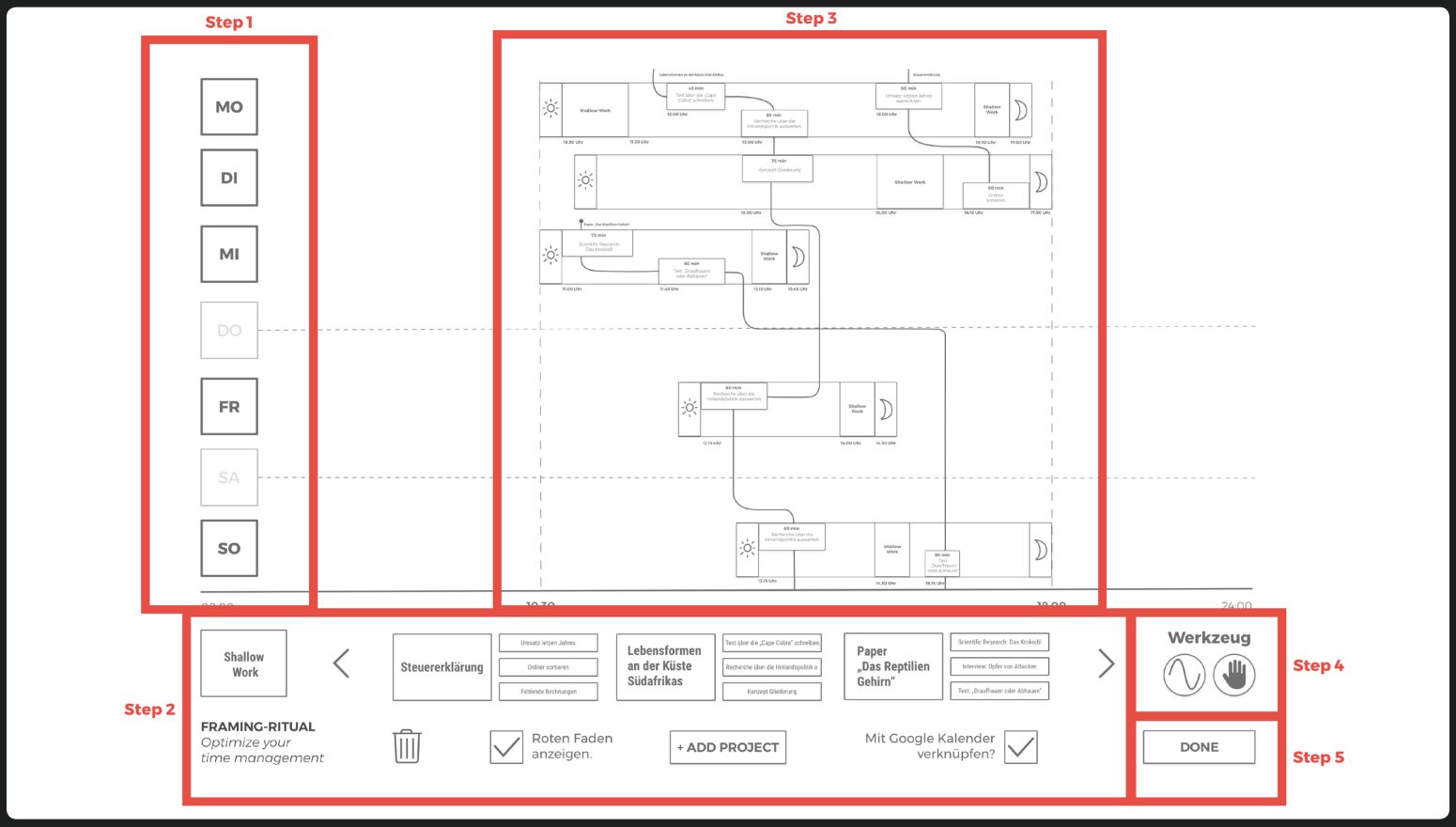


Abb. 30: Wireframe 3 Framing

Zudem gibt es die Option, den Google-Kalender oder ähnliche Kalender-Tools mit offener Schnittstelle einzubinden. Dann ist es nicht möglich, vom Kalender blockierte Timeslots mit Deep-Work-Phasen zu belegen und andersherum werden die Deep-Work-Phasen optional automatisch im Kalender mit eingetragen.

Genauso gibt es die Option, sich die Roten Fäden zwischen den Tasks anzeigen zu lassen oder lieber nicht.

Es wurde sich bewusst dazu entschieden, eine Wochenansicht zu wählen, um die Verknüpfung der Tasks über die verschiedenen Arbeitstage möglich zu machen. Den Arbeitstag als „Frame" darzustellen ist einerseits praktisch, weil sich so präzise ein Zeitraum begrenzen lässt und es zudem auch die Metapher des Rahmens unterstützt.

Die nun eingetragenen Projekte, Tasks, Zeiten und Verknüpfungen dienen als Grundlage für das System. Dadurch wird festgelegt, wann es Bereitschaft signalisieren oder durch Trigger gewisse Phasen einleiten soll. Für den Nutzer fungiert es zudem so wie eine To-do-Liste für die Woche.

Sound-Konzept

Interaktions-Sounds

In einer Welt voller vernetzen Objekte, smarten Devices und Kommunikationstools, ist ein gut durchdachtes Sound-Design wichtiger denn je. Denn jedes dieser Systeme hat Benachrichtigungstöne, Alarm-Sounds und einen ganzen Haufen anderer auditiver Feedback-Signale.

Um der Geräuschkulisse nicht noch unnötig beizutragen, werden so wenige Sounds wie möglich verwendet. In dem Wissen, dass diese gezielten Klänge und Töne alltäglich gehört werden und einen großen Teil der Charakteristik des Systems ausmachen, ist es umso wichtiger, dass diese einem klaren Konzept folgen. Dabei wird das Sound Engineering von Matthew Bennett, dem Lead Sound Engineer von Microsoft als Leitprinzip genommen. Die meisten Benachrichtigungsgeräusche basieren auf einem Alarm-Paradigma. Diese sind meistens laut, unerwartet und lassen dem Reptilien-Teil des Gehirns instinktiv nur zwei Optionen: Angriff oder Flucht. Der Körper ist unterbewusst einem hohen Stress ausgesetzt. Matthews Ansatz geht davon aus, dass der Mensch Sprache auf einem gewissen Level als Musik wahrnimmt. Es geht um die Melodie, wie sich die Tonhöhe verändert, den Rhythmus und das Tempo.

Wenn man jetzt zum Beispiel “Ready to go” als Musik wahrnimmt, lässt sich feststellen, dass diese Sprachübergreifend funktioniert. Es geht darum, die Botschaft der Sprache zu nehmen und sich die emotionale Kontur des Gesagten zunutze zu machen und diese im Sound zu vermitteln. Auch wichtig ist, wie das Geräusch ins Leben gerufen wird und auch wieder mit der Stille verschmilzt. Davon hängt auch ab, ob das Gehirn dieses als Gefahr wahrnimmt und Stresshormone ausgeschüttet werden oder nicht.

Bei unserem System funktionieren die Sounds nicht nur als Benachrichtigungen, Begrüßung oder als Alarm. Die verwendeten Geräusche dienen als Trigger-Elemente, um Gewohnheiten einzuführen und um dem Nutzer dabei zu helfen, sich schneller auf einen neuen Kontext einlassen zu können. Sie sind somit ein direktes Mittel, um die Einstiegshürde in den Tunnel zu eliminieren und den Start in den Arbeitstag zu erleichtern. Welche Aussagen sollten diese Töne haben? Bei der Gestaltung wurde eine Kombination aus der sprachlichen Aussage und mentalen Modellen, um diese zu unterstreichen, gewählt.

Abb. 31: Sprechmelodie

Boot-Up
Mentales Modell: Die Sonne geht auf.
Aussage: “Nice to meet you.“ oder “Ready to go?” oder “Lets start work!”

Bestätigungssound
Mentales Modell: Schulterklopfer.
Aussage: “Well done.”

Timer aufziehen
Mentales Modell: Etwas aufladen/Energie speichern.
Aussage: “Let’s go!”

Draft formulieren
Mentales Modell: Bogen spannen.
Aussage: “Am I right?”

Shut-Down
Mentales Modell: Sonnenuntergang
Aussage: “Work is done.”

Es wurde nicht überprüft, ob die Klangmelodie der gewählten Sätze in allen Sprachen funktioniert. Lediglich Englisch und Deutsch wurden überprüft. Sollte das Produkt zum Beispiel auf den chinesischen Markt ausgeweitet werden, müssten die Sounds erneut auf ihre Validität überprüft werden.

Die Deep-Work-Playlist

Bei der Gestaltung der Musik, welche im Hintergrund der Deep-Work-Sessions läuft, wurde die Zielgruppe befragt. Dabei sind folgende Vorgaben herausgekommen:

Treibend,
Nicht dramatisch,
Gleichbleibende Stimmung,
Keine ablenkenden Geräusche,
Kein Gesang,
Eher im Hintergrund.

Die ersten Gedanken gehen in die elektronische Richtung „Liquid Drum and Bass“ oder „Deep House“. Des Weiteren wurde recherchiert. Schnell wird klar, dass es für Alphawellen, „binaurale Beats“ oder ähnliche Konstrukte keinen wissenschaftlichen Beleg gibt und deswegen werden diese außen vor gelassen.

Es wurde Brain.fm entdeckt. Ein Start-up, welches sich wissenschaftlich mit der Art von Musik beschäftigt, die nötig ist, um in den Flow-Zustand zu gelangen. Deren Musik deckt sich perfekt mit den Vorgaben der Zielgruppe und auch Brain.fm halten nichts von Alphawellen und haben eine gut fundierte Herangehensweise.

In der Zukunft ist es gut vorstellbar mit Brain.fm zu kooperieren und deren Musik in FlowLink einzubinden. Fürs Erste wird sich für gleichmäßigen „Liquid Drum and Bass" ohne Gesang entschieden. Dieser erfüllt die aufgelisteten Kriterien, da er mit einem gleichmäßigen und schnellen Rhythmus der zwischen 130 und 150 bpm (beats per minute) liegt und somit ungefähr die doppelte Geschwindigkeit des Herzschlages eines Menschen hat. Der minimalistische Einsatz von Klängen und das harmonische Zusammenspiel mit dem Schlagzeug Rhythmus erzeugen eine ausgeglichene aber nicht zu entspannte Arbeitsatmosphäre. In Eigentests hat sich diese Musikrichtung effektiver als sogar Brain.fm erwiesen.

Uns ist jedoch bewusst, dass nicht jeder Mensch gleich auf Musik reagiert. Deswegen wird es in kommenden Iterationen die Einbindung von Spotify Premium geben. Dann ist es jedem Nutzer möglich, seine eigene Deep-Work-Playlist zusammenzustellen und diese in FlowLink zu integrieren.

Iterative Optimierung

Während des Testens des Prototypen haben sich viele Verbesserungen ergeben. Die meisten davon wurden bereits in das Konzept übernommen.

Eine der wichtigsten Beobachtungen aus den Usertests war das Problem der ersten Aufgabe. Das Problem der ersten Aufgabe tritt auf, wenn eine Aufgabe gestartet wird, die noch nicht vorhanden war und somit auch keinen Leitfaden besitzt. Zum Zeitpunkt des Usertests musste der Draft nur ausgefüllt werden, wenn die Aufgabe beendet wurde. Es wurde schnell entdeckt, dass die Nutzer sich beim Starten einer neuen Aufgabe häufig planlos in die Aufgabe stürzen und deswegen viel Zeit benötigen, bis sie im Tunnel sind. Als Lösung dafür wird deswegen der Draft auch direkt, nachdem eine neue Aufgabe gestartet wurde, angezeigt. Dies sorgt für ein strukturiertes Herangehen an die Aufgabe. Es kann so zwar keine wertvolle Denkmasse, die nur im Flow zur Verfügung steht, abgegriffen werden, aber durch die Struktur zumindest die Hürde etwas minimiert werden. Doch gleichzeitig wurde hier auch festgestellt, dass das Ausfüllen des Drafts am Anfang der Arbeit mehr Motivation erfordert als am Ende, weswegen eine vereinfachte Variante des Drafts momentan noch diskutiert und getestet wird.

Des Weiteren wurde viele kleine Features, wie die Möglichkeit die Musik stumm zu stellen oder eine Aufgabe aus der Liste zu löschen, gefordert und implementiert.

Zusätzlich wurde festgestellt, dass die Nutzer die potenziellen Vorteile für ihre Arbeit begreifen müssen, bevor sie gewillt sind, das Tool zu benutzen. Der Grund hierfür ist ganz einfach: Das Tool verlangt von dem Benutzer Arbeit zu leisten. Deswegen sollte im finalen Produkt ein besonderer Fokus auf dem interaktiven Tutorial liegen.

Alternative Interfaces

Das Interface von FlowLink findet ausschließlich auf einem flachen, zweidimensionalen Bildschirm statt. Dies bringt einige Vorteile, wie auch Nachteile mit sich. Diese Vor- und Nachteile, sowie alternative Ansätze für das Interface werden im Folgenden diskutiert.

Eines der Kernprinzipien von FlowLink ist die Eigenschaft, im Hintergrund zu bleiben und den Nutzer auf die Arbeit fokussieren zu lassen. Damit FlowLink das schaffen kann, sollte es sich nahtlos in den Arbeitsalltag des Nutzers integrieren. Bringt FlowLink nämlich Anforderungen mit, die den Nutzer zum Beispiel in seiner Standortwahl oder Arbeitszeit beschränken, besteht die Gefahr, dass die Arbeit negativ beeinflusst wird. Als Software ist FlowLink im PC verankert, was wiederum auch den Arbeitsplatz dort oder zumindest in der Nähe verankert. Dies ist ein Problem, dass durch alternative Interfaces potenziell gelöst werden könnte.

Man stelle sich einen haptischen Gegenstand vor, der leicht transportiert werden kann und die Funktionalität von FlowLink in sich trägt. Damit wäre es möglich, standortunabhängig zu arbeiten. Das ist besonders für die Zielgruppe „Wissenschaftler“ relevant, da diese häufig unterwegs sind und in variablen Bereichen arbeiten. Die Arbeit im Labor oder ohne Computer an einem Whiteboard ist durchaus denkbar. Zudem könnte dieser haptische Gegenstand noch einen weiteren Vorteil mitbringen. Die Pomodoro Technik hat sich diesen zunutze gemacht. Das ist eine Arbeitsmethode, die auf den Einsatz eines herkömmlichen Küchentimers setzt, der manuell betätigt werden muss. Durch das physische Aufziehen des Timers wird der Trigger, der dem Gehirn mitteilt, dass die Arbeit jetzt angefangen hat, verstärkt. Dieses physische Aufziehen könnte auch möglich sein mit dem haptischen FlowLink. Des Weiteren könnte ein physischer Timer als erweitertes haptisches Interface, um mit dem System zu interagieren, dienen. Über auditive und visuelle Signale könnten neue Trigger gesetzt werden und zum Beispiel sanft das Boot-Up eingeleitet werden. Auch die Interaktion zwischen Nutzer und dem System könnte erweitert werden. Außer der Interaktion des physischen Aufziehens könnte ein Nach-Vorne-Kippen des Timers das Shut-Down einleiten oder Ähnliches. In jedem Fall würden diese Tangible Interactions die Trigger-Elemente verstärken und die Benutzung von FlowLink einprägsamer machen.

Doch das screen-basierte Interface bringt Vorteile mit sich, die ein haptisches FlowLink, dass komplett unabhängig von der Software ist, unattraktiv macht. Eine wichtige Komponente von FlowLink und dem Roten Faden ist der auszufüllende Draft und der daraus resultierende Leitfaden. Der Leitfaden steht im Mittelpunkt des Konzepts, weil mit ihm das Problem – die Hürde, um in den Flow zu gelangen – gelöst wird. Ein FlowLink ohne diese Komponente wäre somit ein unsinniges Produkt ohne einzigartigen Mehrwert. Damit müsste das haptische FlowLink also die Möglichkeit besitzen, den Draft auszufüllen. Hierbei ist keine Lösung, sei es mit Sprache, Touch oder Gestik denkbar, die eine bessere Nutzererfahrung bietet als das Ausfüllen am Bildschirm.

Dies führt also zu Mischlösungen, die ein haptisches FlowLink kombinieren mit der vorhandenen Software. Hierbei wäre es denkbar die Funktionalität des haptischen FlowLinks auf ein Minimum zu reduzieren und die screen-basierten Interaktionen der Software zu überlassen. Somit könnte ein physisches Aufziehen das verstärkende Trigger-Element sein und das Ausfüllen des Drafts am Bildschirm mit Tastatur stattfinden. Diese Kombinationen aus beiden Welten kommt aber leider nicht ohne Opfer. Die standortunabhängige Nutzung von FlowLink wäre somit wieder eingeschränkt. Das haptische FlowLink reduziert auf die physische Interaktion des Aufziehens ist zwar standortunabhängig, aber der Kern des Konzeptes, der Rote Faden, nicht.

Diese Erkenntnisse haben dazu geführt, dass das Team sich für das Erste auf eine Software fokussiert. Trotzdem sind die Vorteile von haptischen Gegenständen klar und stark. Die Exploration weiterer, alternativer Interfaces steht somit trotzdem weit oben auf der Agenda

VISUELLES KONZEPT

Bei der Gestaltung von FlowLink wurde stark Bezug auf den Charakter der Anwendung genommen.

So sieht das aus.

Um FlowLink im alltäglichen Gebrauch gut nutzbar zu machen, wurde sich für einen sehr minimalistischen Stil entschieden.

Die Farbwahl

Es werden lediglich drei Farben verwendet:

Weiß

#F1F1F1

Rot

#FB3E39

Schwarz

#202020

So dient schwarz die meiste Zeit als Hintergrundfarbe und weiß ist die Schrift- und Textfeldfarbe. Es wurde sich bewusst für dezente Schwarz- und Weißtöne entschieden. Das sieht einerseits hochwertiger aus, ist andererseits auch nicht so anstrengend für die Augen. Aus dem gleichen Prinzip wurde sich für Schwarz als Hintergrundfarbe entschieden. Dieser „Trick“ wurde sich von vielen Programmen, welche für das Programmieren entwickelt wurden, abgeschaut, da ein dunkler Hintergrund weniger anstrengend für die Augen ist und somit nicht unnötig ablenkt.

Bei der Wahl der dritten Farbe wird auf das mentale Modell des Roten Fadens gesetzt. In der Anwendung ist sie die Interaktionsfarbe und setzt Akzente. In dieser Dokumentation eher Letzteres.

Der Rotton muss somit als Signalfarbe dienen, aber auch harmonisch mit dem Schwarz- und Weißtönen agieren. So wird sich für ein gedecktes, aber dennoch leuchtendes Rot entschieden.

Die Gefahr besteht, dass man mit Rot eine Warnfarbe assoziiert. Das wird jedoch umgangen indem es innerhalb der Anwendung nie für Warnungen oder Ähnliches benutzt wird, sondern immer nur in Verbindung mit einer Interaktion, wie zum Beispiel einem Button oder Regler steht.

Im Logo von FlowLink wird Rot benutzt, um das „Link“ einzufärben, da der Rote Faden genau dafür gedacht ist die Flow-Phasen zu verlinken. Das „Flow“ variiert je nach Hintergrundfarbe zwischen Rot und Schwarz.

Abb. 32: Das Logo

Schriften

Die verwendeten Schriften sind Montserrat und Roboto. Beide sind in den Google-Fonts enthalten und fallen somit unter die Open Font License (OFL). Zielsetzung der OFL ist es, das gemeinsame weltweite Weiterentwickeln von Schriften zu fördern. Die OFL erlaubt es, die lizenzierten Schriften frei zu verwenden, zu studieren, zu modifizieren und weiterzugeben, solange sie nicht selbst verkauft werden.

FlowLink
FlowLink
FlowLink
FlowLink
FlowLink
FlowLink
FlowLink
FlowLink
FlowLink

abcdefghiklmnopqrstuvwxyz
ABCDEFGHIKLMNOPQRSTUVWXYZ
0123456789 (!@#$%&.,?:;)

abcdefghiklmnopqrstuvwxyz
ABCDEFGHIKLMNOPQRSTUVWXYZ
0123456789 (!@#$%&.,?:;)

abcdefghiklmnopqrstuvwxyz
ABCDEFGHIKLMNOPQRSTUVWXYZ
0123456789 (!@#$%&.,?:;)

abcdefghiklmnopqrstuvwxyz
ABCDEFGHIKLMNOPQRSTUVWXYZ
0123456789 (!@#$%&.,?:;)

Montserrat wird für die Überschriften verwendet. Sie eignet sich gut, da sie eine serifenlose geometrische Schrift ist. Sie ist klar definiert, etwas breiter als die Durchschnittsschrift und hat einen eindeutigen Charakter. Dieser stammt auch von der Herkunft der Schrift. Sie wurde von der argentinischen Designerin Julieta Ulanovsky entworfen und ist inspiriert von der urbanen Typografie des 20. Jahrhunderts in Montserrat, einem Stadtteil von Buenos Aires. Dort wurden ähnliche Schriften primär für Poster und Schilder verwendet.

Die Leitsätze, die dieses Projekt inspiriert und geprägt haben, sind Arbeit, Hingabe, Pflege, Farbe, Kontrast, Licht und Leben, Tag und Nacht. Sie sollen das Stadtleben von Montserrat widerspiegeln und bieten somit genau die gewünschte Atmosphäre aus simpel, aber leidenschaftlichem Kontrast, welcher für die Überschriften gewünscht ist. So wird Montserrat sowohl für die Überschriften als auch das Logo von FlowLink verwendet. Die Schrift ist außerdem in einer hohen Varianz an verschiedenen Dicken verfügbar, was die Schrift flexibel einsetzbar macht.

FlowLink

FlowLink

FlowLink

FlowLink

FlowLink

FlowLink

abcdefghiklmnopqrstuvwxyz
ABCDEFGHIKLMNOPQRSTUVWXYZ
0123456789 (!@#$%&.,?:;)

abcdefghiklmnopqrstuvwxyz
ABCDEFGHIKLMNOPQRSTUVWXYZ
0123456789 (!@#$%&.,?:;)

abcdefghiklmnopqrstuvwxyz
ABCDEFGHIKLMNOPQRSTUVWXYZ
0123456789 (!@#$%&.,?:;)

abcdefghiklmnopqrstuvwxyz
ABCDEFGHIKLMNOPQRSTUVWXYZ
0123456789 (!@#$%&.,?:;)

abcdefghiklmnopqrstuvwxyz
ABCDEFGHIKLMNOPQRSTUVWXYZ
0123456789 (!@#$%&.,?:;)

Roboto wird für den Fließtext verwendet. Roboto ist eine speziell für das Android-Betriebssystem entwickelte statische Schrift, welche insbesondere den Anforderungen von User-Interface-Designs und hochauflösender Displays gerecht werden soll. Die Schrift ist derzeit in sechs Strichstärken mit zugehörigen Oblique-Schnitten ausgebaut. Außerdem gibt es Condensed-Schnitte und eine Slab-Serif-Version. In den meisten Fällen wird Roboto Condensed verwendet, vor allem in Eingabefeldern und wenn wenig Platz zur Verfügung steht.

Im Fließtext hat Roboto zwei verschiedene Seiten. Es hat ein massives Skelett und die Formen sind weitgehend geometrisch. Gleichzeitig bietet die Schrift freundliche und offene Kurven. Während manche anderen Grotesken die Buchstabenformen verzerren, um einen starren Rhythmus zu erzwingen, macht Roboto keine Kompromisse, sodass die Buchstaben in ihre natürliche Breite gebracht werden können. Dies führt zu einem natürlichen Leserhythmus.

Icons und Sonstiges

Das Interface von FlowLink soll minimalistisch und zuverlässig wirken. So sind die meisten Formen sowie die Schriften geometrisch. Da sich die Bedienung trotzdem modern und flüssig anfühlen soll, gibt es bei FlowLink wenig rechte Winkel. Diese sind immer mindestens mit einer Rundung von 4pt versehen. Auch bei den Icons werden runde etwas dickere Strichformen verwendet.

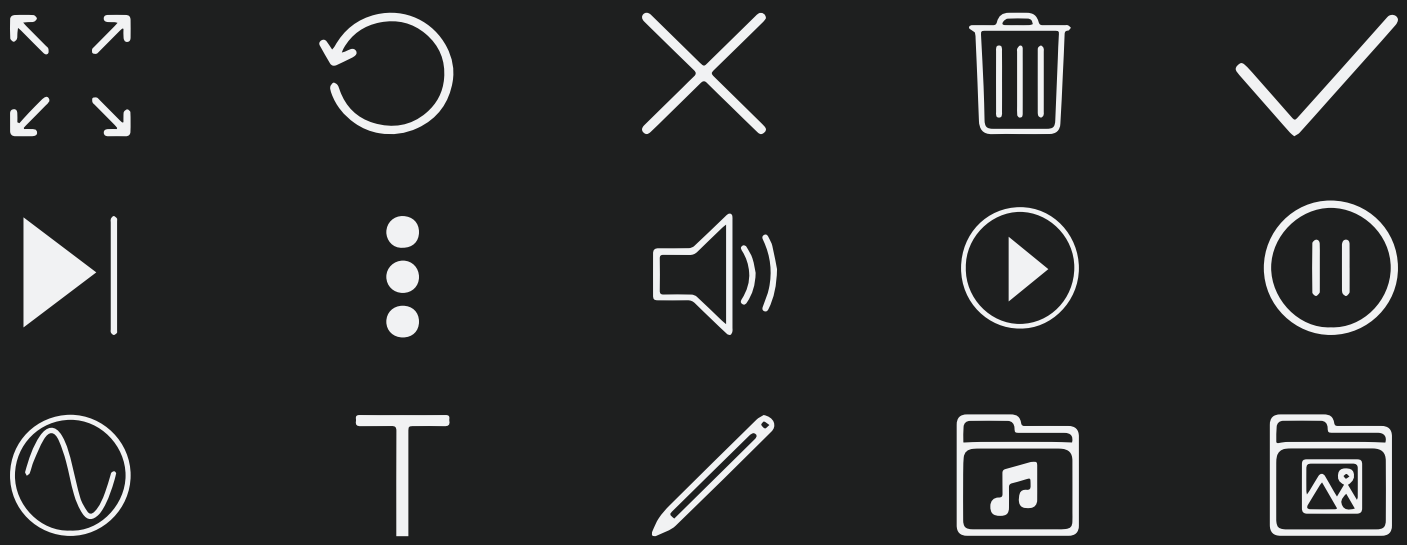


Abb. 33: Icons Icons Icons

Es geht bei der Wahl der Icons nicht darum, aus der Norm zu stechen, sondern so effizient, vorhandene mentale Modelle zu nutzen, wie möglich, um einen unkomplizierten und schnellen Einstieg in das System zu ermöglichen.

Dem Minimalismus ist geschuldet, dass so wenig verschiedene Icons wie möglich verwendet wurden. So ist der Haken Zeichen der Bestätigung. Das X steht für Schließen oder im Kontext des Task-Dropdowns für das Entfernen aus dieser Liste. Der Mülleimer als alleinstehendes Symbol stellt den Mülleimer für Projekte dar und löscht diese aus der Anwendung. Der Lautsprecher dient als Mute-Button und signalisiert über die Schallwellen, ob die Musik derzeit aktiv ist oder nicht. Das Modell des CD-Players wird in der Musiksteuerung benutzt. Über den Faden lassen sich Verknüpfungen zwischen den Tasks setzen und mit der Hand lassen sich Tasks greifen und Verschieben.
Um keine Verwirrung mit der Musiksteuerung zu schaffen, wurde sich gegen das Symbol von Reglern bei den Einstellungen entschieden. Diese wurden durch ein senkrechtes „…“ ersetzt, dass einerseits an das klassische Burger-Menu anlehnen soll und dadurch signalisiert, dass hier erweiterte Optionen liegen.

Für die Bedienung der Hyper-Media-Windows steht der Stift für das händische Zeichnen. Für das Hineinladen externer Dateien, wird sich für das klassische Ordnersymbol, jeweils mit einem Bild oder einer Note darin, für Image- und Audiofiles. Ein „T“ in der für FlowLink charakteristischen Schrift Montserrat erstellt ein Textfeld. Innerhalb der Felder wurde auf eine so unkomplizierte und intuitive Bedienung wie möglich gesetzt. Weniger ist hier mehr.

Das gestalterische Konzept wird nicht nur in dieser Dokumentation angewandt, sondern auch im ersten Prototyp.

SO FUNKTIONIERTS.

Der Prototyp von FlowLink

Prinzipien des Prototypen

Bei der Programmierung gibt es selten Entscheidungen, die eindeutig richtig oder falsch sind. Deswegen haben wir uns dafür entschieden Prinzipien zu definieren, die ein Fundament für das Treffen von Entscheidungen legen. Diese Prinzipien sind:

1. Bietet die Grundlage für eine marktreife Applikation
2. Stabilität über Features.
3. Features über Performance.
4. Ein möglichst modularer Aufbau.

Abb. 34: Der Prototyp Start

Das erste Prinzip basiert darauf, dass zwar im Rahmen dieser Abgabe kein fertiges Produkt entsteht, wir aber das Ziel verfolgen mit der Applikation an den Markt zu gehen.

Das erhöht zwar temporär den Aufwand, sorgt aber auf lange Sicht für eine Zeitersparnis.
Das zweite Prinzip leitet sich einerseits aus dem ersten, aber auch aus dem Kontext, ab. Für einen erfolgreichen Nutzertest müssen unsere Nutzer den Prototypen regelmäßig und am besten über mehrere Tage hinweg benutzen. Hier reicht es also nicht aus, wenn der Prototyp die meiste Zeit funktioniert. Zudem wird der Prototyp besonders in den fragilen Flow-Phasen eingesetzt. Dort wären Abstürze der Applikation extrem kontraproduktiv hinsichtlich unserer Zielsetzung. Daher haben wir uns entschieden die Stabilität höher als die Implementierung von Features zu priorisieren.

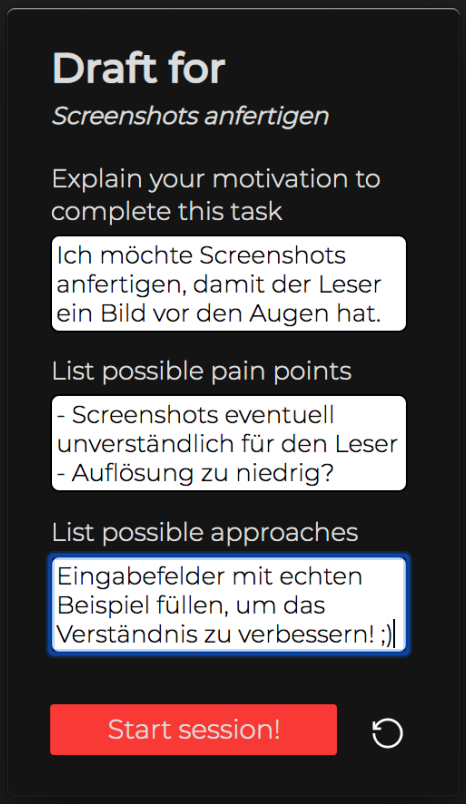


Abb. 35: Der Prototyp Draft

Nichtsdestotrotz ist die Featurevollständigkeit für uns immer noch wichtiger als die Performance. Eine Anwendung, die den Computer verlangsamt ist natürlich suboptimal, da FlowLink die Arbeit optimieren soll. Jedoch macht es wenig Sinn zu viel Zeit auf kleine Performanceoptimierungen zu verschwenden, die im Endeffekt die Nutzererfahrung wenig beeinflussen.

Zuletzt möchten wir die Anwendung noch möglichst modular aufbauen. Während der Entwicklung des Prototypen kann sich das Konzept stark verändern. Das könnte dazu führen, dass Features hinzugenommen oder entfernt werden müssen. Ein modularer Aufbau hilft dabei diesen Prozess zu erleichtern.

Technology Stack

Als Technology Stack werden alle Systeme und Services bezeichnet, die man verwendet, um eine Applikation zu bauen und auszuliefern.

	Funktion	Technologie	
Externe Services (**online**)	Website Hosting	Github Pages	
	Download Server	Amazon S3	
	Analytics / Tracking	Firebase Realtime Database	*Optional*
Interne Services (**offline**)	Building	Electron Builder	
	Testing	Spectron (with Ava)	
	Applikation	Electron App	

Abb. 36: Technology Stack

Das vorherige Schaubild gibt einen Überblick über die verwendeten Technologien. Im Anschluss werden die Systeme und Services im Detail behandelt und begründet, warum sich wie entschieden wurde.

Unsere Anwendung ist mit Electron gebaut. Um einen reibungslosen und stabilen Ablauf der Applikation zu gewährleisten, wird ein Testing Framework verwendet. Spectron ist ein Framework, dass erlaubt Javascript Test Runner wie in diesem Fall Ava zu verwenden. Um dann schließlich eine gepackte Anwendung, wie eine „.exe“ auf Windows oder eine „.dmg“ auf Mac zu erhalten, wird der Electron-Builder verwendet. Damit steht zunächst zwar eine fertige Anwendung zur Verfügung, aber es werden noch Services benötigt, damit diese auch zum Nutzer gelangt. Die Applikation wird auf Amazon S3 gelagert, wo sie dann über ein Web-Frontend heruntergeladen werden kann. Das Web-Frontend wird über Github Pages gehostet. Für den Prototypen haben wir, falls der Nutzer zustimmt, noch die Möglichkeit seine Aktionen zu tracken. Diese Aktionen werden in der Firebase Realtime Database gespeichert.

Zudem ist für die Zukunft geplant, noch ein Continuous Deployment Service in den Technology Stack zu integrieren. Continuous Deployment (CD) Services sind in der Lage den aktuellen Stand der Software aus einem Versionskontrollsystem wie Git zu entnehmen und dann automatisch den Build-Prozess in Gang zu setzen und bei Fertigstellung die Version direkt für den Nutzer verfügbar zu machen. Dies wird uns einiges an Zeit sparen, weil das Bauen und Hochladen der Anwendung automatisiert abläuft. Zudem können CD-Services mit dem Testing Framework zusammenarbeiten und einen automatischen Build nur dann starten, wenn alle Tests erfolgreich ablaufen. Möglich hierfür wäre zum Beispiel GitLab, was gleichzeitig als Versionskontrolle und CD-Service fungiert. Zur Unterstützung ist hier der Update Server Hazel denkbar. Damit wäre es möglich neue Versionen der Anwendung nicht nur auf der Website zu publizieren, sondern auch bestehenden Nutzern zukommen zu lassen.

Damit sieht der potenzielle Technologie Stack für die Applikation nach dem Prototypen so aus:

	Funktion	Technologie	
Externe Services (**online**)	Website Hosting	Github Pages	
	Download Server	Amazon S3	
	Update Server	Hazel	*Neu*
	Continous Deployment	GitLab (CD-Pipeline)	*Neu*
	Analytics / Tracking	Firebase Realtime Database	*Optional*
Interne Services (**offline**)	Building	Electron Builder	
	Testing	Spectron (with Ava)	
	Applikation	Electron App	

Abb. 37: Technology Stack Update

Electron

Beim Prototypen wurde sich für die Plattform Electron entschieden. Dabei ist zu beachten, dass sich die Wahl dieser Plattform nicht nur auf den Prototypen selbst beschränkt, sondern auch als gute Option für das Endprodukt gesehen wird. Denn viele der Gründe, die Electron als Plattform für den Prototypen attraktiv machen, gelten auch für das Endprodukt. Im Folgenden werden die drei ausschlaggebenden Argumente für Electron näher beleuchtet.

Plattformübergreifend

Eine der Grundanforderungen für den Prototypen und das Endprodukt ist die Fähigkeit für unsere Anwendung auf Windows, MacOS und Linux betriebsfähig zu sein. Hierbei gibt es prinzipiell zwei Möglichkeiten. Entweder wird ein plattformübergreifendes Framework verwendet oder mehrere native Anwendungen programmiert.

Nativ vs. Plattformübergreifend

Native Anwendungen sind Anwendungen, die speziell für ein Betriebssystem entwickelt werden. MacOS bringt zum Beispiel über die X-Code-IDE die Möglichkeit mit, native MacOS Anwendungen mit Swift zu programmieren. Dabei ist es sehr einfach, auf MacOS proprietäre Features wie die Touchbar oder das Dock zurückzugreifen, weil die Anwendung nur für MacOS kompiliert werden muss. Ähnliches gibt es natürlich auch bei Windows über die „Universal Windows Apps“ (UWP). Durch die Optimierung auf eine Plattform ist die Performance von nativen Apps häufig besser als die von Plattformübergreifenden. Zudem integrieren sich nativ programmierte User-Interfaces (UI) häufig besser in das restliche Betriebssystem, weil sie den jeweiligen Standards des Betriebssystems folgen.

Diese Vorteile sind natürlich nicht ohne Preis zu genießen. Eine Desktopanwendung nativ zu programmieren bedeutet drei Desktop-Anwendungen zu programmieren. Deswegen lohnt sich der Aufwand meist nur, wenn die Anwendung stark abhängig ist von nativen Features, Performance ein sehr relevanter Faktor ist oder wenn sich die Userbase größtenteils nur auf einer Plattform befindet. Da diese Punkte bei uns nicht zu treffen, wurde sich gegen die Programmierung mehrerer nativer Anwendungen entschieden. Generell war die Wahl des Betriebssystems unter den befragten Wissenschaftlern divers, weswegen eine einzelne, native Anwendung auch nicht infrage kam. Zudem führt unsere Anwendung weder CPU- noch GPU-intensive Berechnungen durch. Deswegen ist die Performance auch kein extrem ausschlaggebender Faktor.

Plattformübergreifend, aber wie?

Es wurde sich also entschieden ein Plattformübergreifendes Framework zu verwenden. Die folgenden beiden Punkte behandeln, warum sich letztendlich für Electron entschieden wurde. Andere, mögliche Frameworks wären QT, Proton oder Flutter for Desktop.

Etablierte UI Umgebung

Es gibt unzählige Tools für das Bauen von UIs. Hierbei hat Electron einen großen Vorteil gegenüber der Konkurrenz. Da Electron für das Rendern des UIs auf Chromium – die Opensource-Version des Chrome Browsers – setzt, kann hier für das UI HTML und CSS eingesetzt werden. Für das Bauen von UIs via HTML und CSS gibt es unzählige Frameworks, die es erlauben leicht responsive Anwendungen zu bauen.

Im Vergleich bietet QT zum Beispiel ein gut etabliertes Framework für ein plattformübergreifendes UI. Doch QT verlangt im Gegensatz zu Electron Lizenzgebühren und benutzt als Sprache C++, was mit unserem Kenntnisstand das Bauen eines schnellen Prototypen unmöglich macht.

Ein weiterer Konkurrent zu Electron ist Proton Native, der sich bewusst als Electron-Konkurrent platziert. Vorteil gegenüber Electron sei vor allem die Einsparung von Ressourcen, da Proton Native keinen vollwertigen Browser mitliefert. Proton Native ist ein sehr interessantes und vielversprechendes Projekt, jedoch keineswegs so etabliert, umfangreich oder community-stark wie Electron. In Zukunft könnte sich das ändern und Proton Native eine vielversprechende Möglichkeit darstellen, eine ressourcenschonende Anwendung für den Desktop mit Javascript zu entwickeln.

Ein weiteres, vielversprechendes Framework, das aber nicht bereit für die Produktion ist, ist „Flutter for Desktop“. Flutter ist ein UI-Toolkit von Google, welches zuerst für plattformübergreifende Anwendungen in der mobilen Welt (iOS und Android) entworfen wurde, aber inzwischen auch anstrebt im Web und auf dem Desktop einsatzbereit zu sein. Als Programmiersprache wird hier die relativ unbekannte und von Google selbst entworfene Sprache Dart verwendet. Flutter selbst schreibt, dass die Desktop API in den frühen Phasen der Entwicklung sei. Sollte sich später dafür entschieden werden eine mobile Anwendung von FlowLink zu veröffentlichen, könnte Flutter die richtige Wahl sein, weil so die Desktop- und mobile Anwendung eine gemeinsame Basis teilen würden.

Große Community

Der wichtigste Punkt, der für Electron spricht, ist die Größe der Community. Das Entwickeln von Anwendungen wird extrem beschleunigt, wenn es viele Mitstreiter gibt, die Ähnliches probiert haben. Es erlaubt dem Programmierer schnell Lösungen für Probleme zu finden, weil mit der Größe der Community auch die Chance steigt, dass jemand anderes das gleiche Problem bereits gelöst hat. Zudem ist ein weitverbreitetes Framework meist stabil, da es schon von unterschiedlichsten Leuten auf unterschiedlichsten Computern mit der unterschiedlichsten Hardware getestet wurde und so eine große Anzahl an Bugs bereits eliminiert sind. Vor allem bei Open-Source-Anwendungen wie Electron ist dieser Effekt verstärkt, weil theoretisch jeder fähige Programmierer einen entdeckten Bug selbst beheben könnte und so nicht darauf angewiesen ist, zu warten bis die Entwickler den Bug behoben haben.

Electron ist mit 75.000 Stars auf Github unter den Top 20 der beliebtesten Repositories auf der Plattform und nur knapp unter dem Linux Kernel. Zudem ist Visual Studio Code – einer der beliebtesten Code Editoren, der mit Elektron erstellt wurde – auch unter den Top 20. Deswegen ist klar, dass es sich hierbei um eine aktive Community handelt, was bei der Programmierung des Prototypen bestätigt wurde.

Nachteile von Electron

Trotz der Vorteile ist natürlich keine Plattform perfekt und es gibt einige Nachteile an Electron, über die man sich als Entwickler bewusst sein sollte. Um die Nachteile zu verstehen, muss zunächst einmal die grundsätzliche Architektur des Frameworks erklärt werden. Electron hat zwei Grundbestandteile: einen Node.js Server für den Hauptprozess und einen Chromium Browser für die Renderprozesse. Zudem gibt es die Interprocess-Communication API (IPC), die den beiden Prozessen über ein Messaging-System erlaubt miteinander zu kommunizieren.

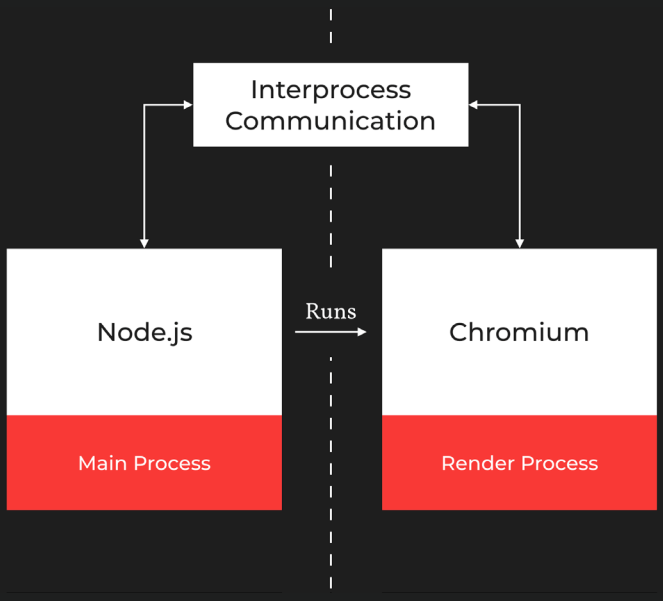


Abb. 38: Electron Architektur

Da der Chromium Browser wie Chrome auch ein vollwertiger Browser ist, sorgt dieser dafür das Electron Anwendungen per se eine Mindestgröße von rund 70 MB haben, auch wenn sie keinerlei Funktionen oder UI besitzen. Dies stellt für umfangreiche Anwendung keinen großen Nachteil da, weil die 70 MB im Vergleich zur restlichen Anwendung nur einen kleinen Teil ausmachen. Doch für FlowLink stellen 70 MB schnell mehr als 50 % der Anwendungsgröße da. Wenn die Musik komplett online gestreamt würde, dann sogar bis zu 80 % . Dieser Nachteil wird unserer Meinung nach durch die Vorteile aufgehoben. Jedoch ist uns bewusst, dass auf jeden weiteren Megabyte geachtet werden muss, um die Dateigröße in Relation zur Funktionalität nicht unzumutbar zu machen.

Im Durchschnitt besitzt Deutschland nach dem „Speedtest Global Index“ von Juni 2019 eine Internet-Geschwindigkeit von 73,05 Mbit/s. Unsere Anwendung wird schätzungsweise (ausgehend von dem Prototypen) eine Downloadgröße von rund 120 MB haben. Damit beträgt die durchschnittliche Downloadzeit in Deutschland rund 13 Sekunden. Laut einem weiteren Bericht des Breitbandatlas besitzen über 92,4 % der Haushalte in Deutschland eine Verbindung mit mindestens 16 Mbit/s Downloadrate.

Gehen wir nun von 16 Mbit/s aus, die praktisch flächendeckend verfügbar sind, handelt es sich um eine Downloadzeit von rund 60 Sekunden. Wir haben für uns festgelegt, dass über 90 % der deutschen Haushalte unsere Anwendung in stets unter 90 Sekunden herunterladen können sollten. Dies ist aber nicht wissenschaftlich belegt und bedarf weiterer Studien.
Des Weiteren ist Electron dafür bekannt, wie auch Chrome, viel RAM zu verbrauchen. Der Prototyp verbraucht im aktuellen Zustand rund 250 MB RAM. Dabei sollte aber gesagt sein, dass hier großes Optimierungspotenzial besteht.

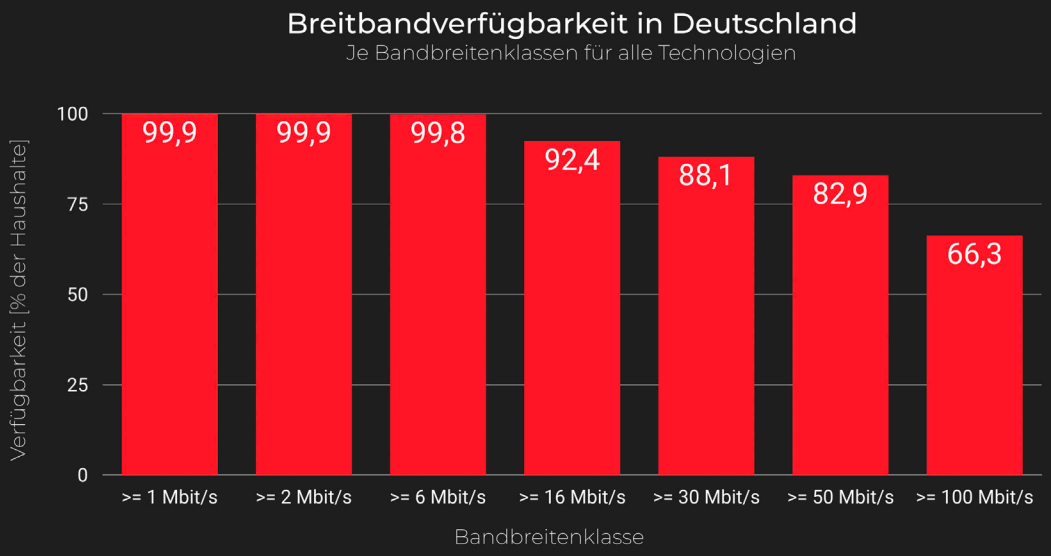


Abb. 39: Breitband in Deutschland

Gehen wir erst mal von 250 MB aus, da auch mit Optimierungen bei weiteren Features diese Menge schnell wieder erreicht sein könnte. Laut der „Steam Hardware Survey“ von Juni 2019 besitzen 38,36% der Steam Nutzer 8 GB an RAM, 33,89% 16 GB an RAM und 17,96% unter 8 GB an RAM, wobei darunter die meisten 4 GB (mit 8,61%) oder 3 GB (mit 3,04%) an RAM besitzen. Die Steam Hardware Survey bietet sich sehr gut an aufgrund ihrer großen Nutzerbasis und Aktualität, aber es sollte beachtet werden, dass die Nutzer von Steam wahrscheinlich dazu tendieren, einen besseren Computer als der Durchschnitt zu besitzen. Deswegen gehen wir hier tendenziell zur Sicherheit vom unteren Ende aus. Es ist also sehr wahrscheinlich das unsere Nutzer mindestens 4 GB an RAM und sehr wahrscheinlich sogar mindestens 8 GB an RAM besitzen. Wenn wir davon ausgehen, dass das Betriebssystem rund 2 GB an RAM in Anspruch nimmt, würde unsere Anwendung bei 8 GB an vorhandenem RAM rund 1/24 des restlichen RAMs in Anspruch nehmen.

Das ist zwar nicht ideal, aber es kann davon ausgegangen werden, dass die meisten Nutzer es nicht bemerken. Bei 4 GB wiederum würde unsere Anwendung rund 1/8 des restlichen RAMs in Anspruch nehmen und ist in diesem Fall für eine Anwendung, die unauffällig agieren soll zu viel RAM-Verbrauch bei Nutzern mit <= 4 GB an RAM. Daraus resultieren zwei Erkenntnisse. Erstens möchten wir in der fertigen Anwendung darauf abzielen, dass unser System, solange es nicht aktiv ist und sich nur in der Menüleiste befindet, unter 100 MB an RAM in Anspruch nimmt. Zweitens ist unsere App für Nutzer mit älteren PCs (<= 4 GB an RAM) deswegen weniger geeignet.

Diese Nachteile machen klar, dass das Entwicklungsteam stets optimieren muss und die Augen offen halten sollte für neue Frameworks bzw. für die Entwicklung von existierenden Frameworks wie Flutter oder Proton, die versprechen ressourcenschonender zu agieren.

Test Driven Development

Aufgrund der vorher festgelegten Grundsätze für den Prototypen, wurde sich für den Einsatz von Test Driven Development (TDD) entschieden. TDD hilft dabei eine stabile Codebase aufzubauen, die sich auch nach dem Prototypen weiter verwenden lässt. Zudem sorgt es für kontinuierliche Stabilität, da kein Build erstellt werden kann, ohne dass alle Tests erfüllt sind. Des Weiteren hilft es dabei unseren Grundsatz der Modularität zu erfüllen, weil sich beim TDD immer auf kleine Teile des Programms fokussiert wird, bis deren Funktionalität steht (sogenannte Unit Tests).

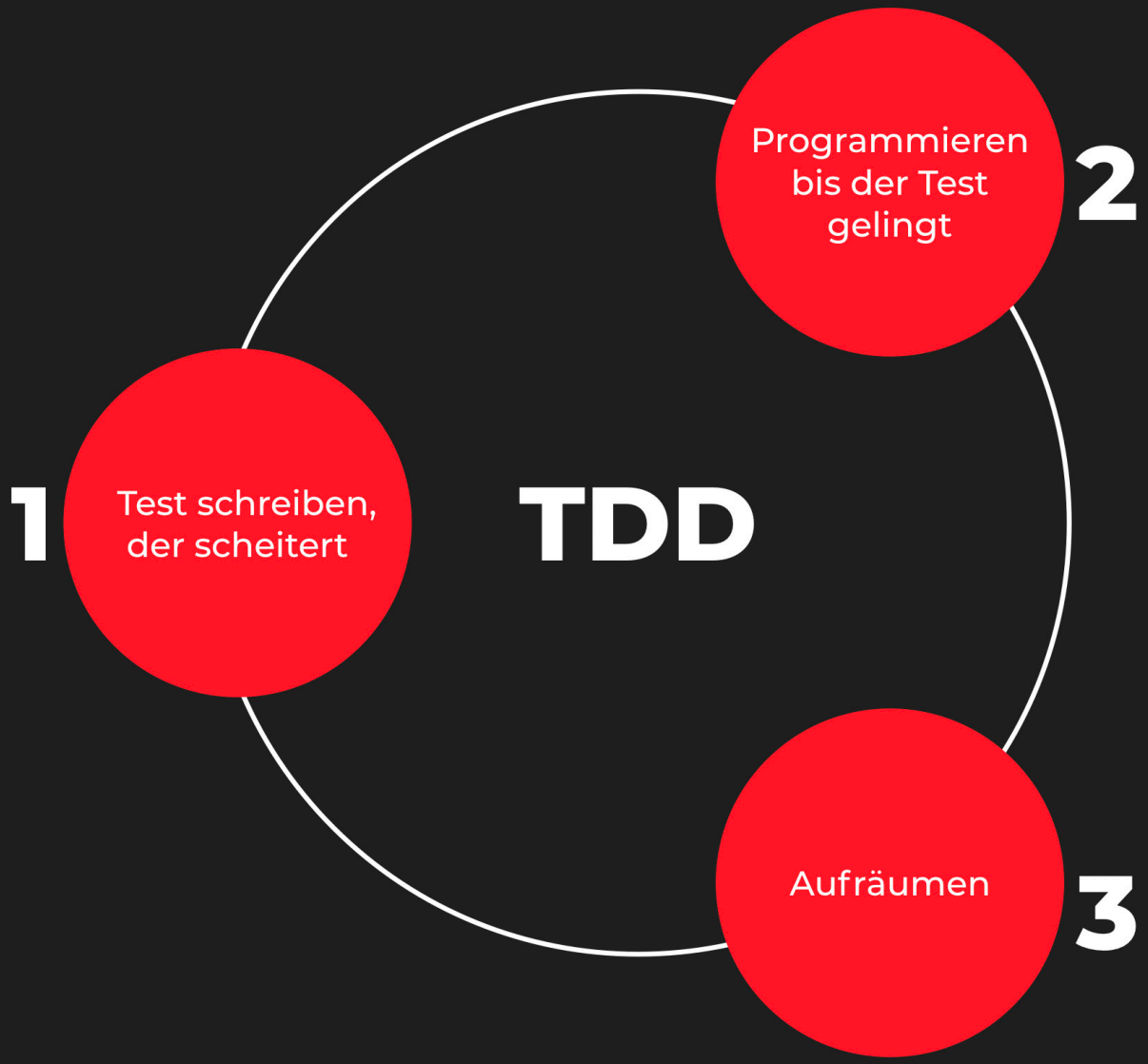


Abb. 40: Test Driven Development

Es geht darum Tests zu schreiben, die definieren, wie die Ausgabe einer Funktion aussehen soll, bevor diese implementiert wird. Es können mehrere Use-Cases mit jeweils einer beispielhaften Eingabe und Ausgabe definiert werden. Dabei interessiert uns, während wir den Test definieren, noch nicht wie die Funktion es schafft das Richtige auszugeben. Zum besseren Verständnis hier ein Beispiel:

```
1   //Example Test written in Mocha.js
2   describe("Hex zu RGB Farbkonvertierung", function() {
3       var rot   = converter.hexToRgb("ff0000"); //Beispiel 1
4       var gruen = converter.hexToRgb("00ff00"); //Beispiel 2
5       var blau  = converter.hexToRgb("0000ff"); //Beispiel 3
6       //Erwartete Ausgabe wird definiert
7       expect(rot).to.deep.equal([255, 0, 0]);
8       expect(gruen).to.deep.equal([0, 255, 0]);
9       expect(blau).to.deep.equal([0, 0, 255]);
10  });
11
```

Abb. 41: Testing Codebeispiel Mocha

Wir möchten zum Beispiel Tests definieren für eine Funktion, die Hex- in RGB-Farben umwandelt. Zuerst rufen wir unsere Funktion auf mit drei verschiedenen Beispielen (Rot, Grün, Blau). Danach geben wir an, was die Funktion bei unseren Beispielen ausgeben soll, damit der Test erfolgreich abläuft. Dies ist ein Unit-Test. Zudem existieren auch noch Integration-Tests bei denen getestet wird, ob mehrere Funktionen so wie geplant miteinander interagieren.
Der geneigte Leser könnte sich nun fragen, ob rapid Prototyping und TDD zusammen funktionieren können. Dem standen wir am Anfang auch skeptisch gegenüber, doch nach der Arbeit an dem Prototypen sind wir der Meinung, dass es funktioniert. Wenn es sich, um einen sehr kleinen Prototypen handelt, der in wenigen Stunden gebaut werden muss wahrscheinlich nicht, aber wenn es sich um ein Projekt über Wochen handelt, halten wir es für sinnvoll. Natürlich kostet das Schreiben von Tests Zeit. Dafür wird aber wiederum Zeit gespart, die sonst mit dem debuggen verbracht werden würde. Zudem hatte Stabilität für uns eine so hohe Priorität, sodass wir bereit waren auch extra Zeit aufzuwenden.

Schließlich hatten wir die Erkenntnis, dass TDD mehr als nur das Schreiben von Tests ist. Es ist eine Philosophie des Programmierens. Es muss zuerst an den Nutzer und wie er mit dem Interface interagiert gedacht werden, als an das Programm selbst. Denn der Nutzer und seine Interaktionen definieren die Tests, und das Programm selbst ist nur da, um die Tests zu erfüllen. Wir sehen hier Parallelen zwischen dem nutzerzentrierten Design und dem TDD.

Wie aber hat TDD in der Welt von Electron und Javascript genau funktioniert?

Spectron und Ava

Für das Testen von Electron haben wir Spectron in Kombination mit Ava verwendet. Spectron ermöglicht das Schreiben von Unit- und Integration-Tests für eine Electron Applikation. Dabei ist Spectron nur das Framework für eine Testing Library, wie in unserem Fall Ava. Ava ist zwar nicht so flexibel und erweiterbar wie zum Beispiel Mocha (eine andere Testing Library), aber gewinnt mit seiner Simplizität. Das Aufsetzen von Ava funktioniert schnell und ohne große Konfiguration, was es für die schnelle Entwicklung des Prototypen sehr attraktiv gemacht hat. Zudem ermöglicht Ava von Haus aus das Testen von asynchronen Funktionen, die vor allem bei der Interprocess-Communication häufig auftreten.

Wir haben im vorherigen Text Beispielcode für das Testen in Javascript gesehen. Nehmen wir das gleiche Beispiel, aber geschrieben in Ava.

```
1  //Example Test written with Ava
2  test('Testing Hex to RGB Conversion', colorTest =>{
3      colorTest.is(converter.hexToRGB("ff0000"), [255, 0, 0]);
4      colorTest.is(converter.hexToRGB("00ff00"), [0, 255, 0]);
5      colorTest.is(converter.hexToRGB("0000ff"), [0, 0, 255]);
6  })
```

Abb. 42: Testing Codebeispiel Ava

Es fällt sofort die schlanke Syntax auf, die das Schreiben von Tests sehr einfach macht. Prinzipiell laufen Tests in Ava gleichzeitig ab, wenn es nicht speziell durch das Kennwort „serial“ gekennzeichnet wurde. Zudem erlaubt Ava die Verwendung von Hooks, zu Deutsch Haken. Damit kann man bestimmte Tests mit Events verbinden . So werden diese nur ausgelöst, wenn das Event getriggert wird. So gibt es zum Beispiel „before“ und „after“ Events, die vor allen anderen bzw. nach allen anderen Tests ausgeführt werden.

Das oben beschriebene Beispiel läuft synchron, aber würde sich einfach durch das hinzufügen von den „async“ und „await“ Schlüsselwörtern in einen asynchronen Test verwandeln.

Electron Builder

Für das Bauen von Electron Anwendung gibt es verschiedene Tools, die jeweils mehr oder weniger Features mitbringen. Jedoch haben alle prinzipiell erstmal die Aufgabe aus dem Quellcode eine gepackte Applikation, welche ausgeführt werden kann, zu machen.

Dabei ist dieser Prozess leider nicht so plattformübergreifend wie das Programmieren, denn zum Bauen einer Anwendung kann es sein, dass das Bauen selbst auf der Zielplattform stattfinden muss. So ist es unmöglich, eine MacOS-kompatible Anwendung auf Windows zu bauen. Umgekehrt ist es zudem nicht möglich eine Windows-Store Applikation auf MacOS zu bauen, wobei das Bauen einer „.exe“ wiederum über den Electron-Build-Service möglich ist. Vor allem das Bauen von MacOS Applikationen ist unmöglich ohne einen Mac, da die Code-Signierung für Mac-Applikationen von Apple bewusst nur auf MacOS Geräten zugelassen wird.

Zum Bauen von FlowLink wurde sich für den Electron Builder entschieden. Es ist das meist verbreitete Tool und bringt im Vergleich zu anderen Buildern über den Electron-Build-Service die Möglichkeit mit sich, zumindest auf MacOS in der Lage zu sein, für alle Plattformen zu bauen. Diese Funktionalitäten bringt Electron-Forge leider momentan noch nicht mit. Dafür bietet Electron-Forge interessante CI/CD Möglichkeiten mit sich. Diese können jedoch ohne Probleme auch mit anderen Tools abgebildet werden.

Code Signierung

Ein Problem mit verpackten Anwendung ist der Bedarf für Code Signierung. Betriebssysteme wie Windows wollen sicherstellen, dass eine Applikation aus sicherer Quelle stammt und dass dem Entwickler vertraut werden kann. Dies geschieht über Zertifikate, die Microsoft selbst oder zertifizierte Unternehmen herausgeben. Für Windows werden sogenannte „code signing certificates“ benötigt, die pro Jahr je nach Anbieter zwischen 75 $ und 500 $+ liegen.

Falls die Applikation nicht mit einem validen Zertifikat signiert ist, dass je nach Betriebssystem unterschiedlich ist, kommen meist Warnmeldung vom Betriebssystem, diese Applikation nicht zu öffnen.

Dies ist für jeden, der nicht gerade einen Entwicklerhintergrund besitzt, ein klares Zeichen die Applikation nicht zu öffnen. Aufgrund der Kosten wurde sich im Prototypen dagegen entschieden eines dieser Zertifikate zu erwerben. Dies ist natürlich aber unabdingbar für eine produktionsreife Applikation.

Musik Streaming

Um die Musikauswahl zu individualisieren, ist geplant die Verbindung zwischen Spotify und FlowLink zu ermöglichen. Dieser Text beschäftigt sich mit den Tücken, die diese Verbindung auf sich hat und warum sich dagegen entschieden wurde es im Prototypen zu implementieren.

Prinzipiell bietet Spotify zwei Services für Entwickler an. Es gibt die Web API und das Webplayback SDK, dass sich momentan noch in der Beta befindet. Was diese Services können und wie sie interagieren, ist nicht sofort ersichtlich.

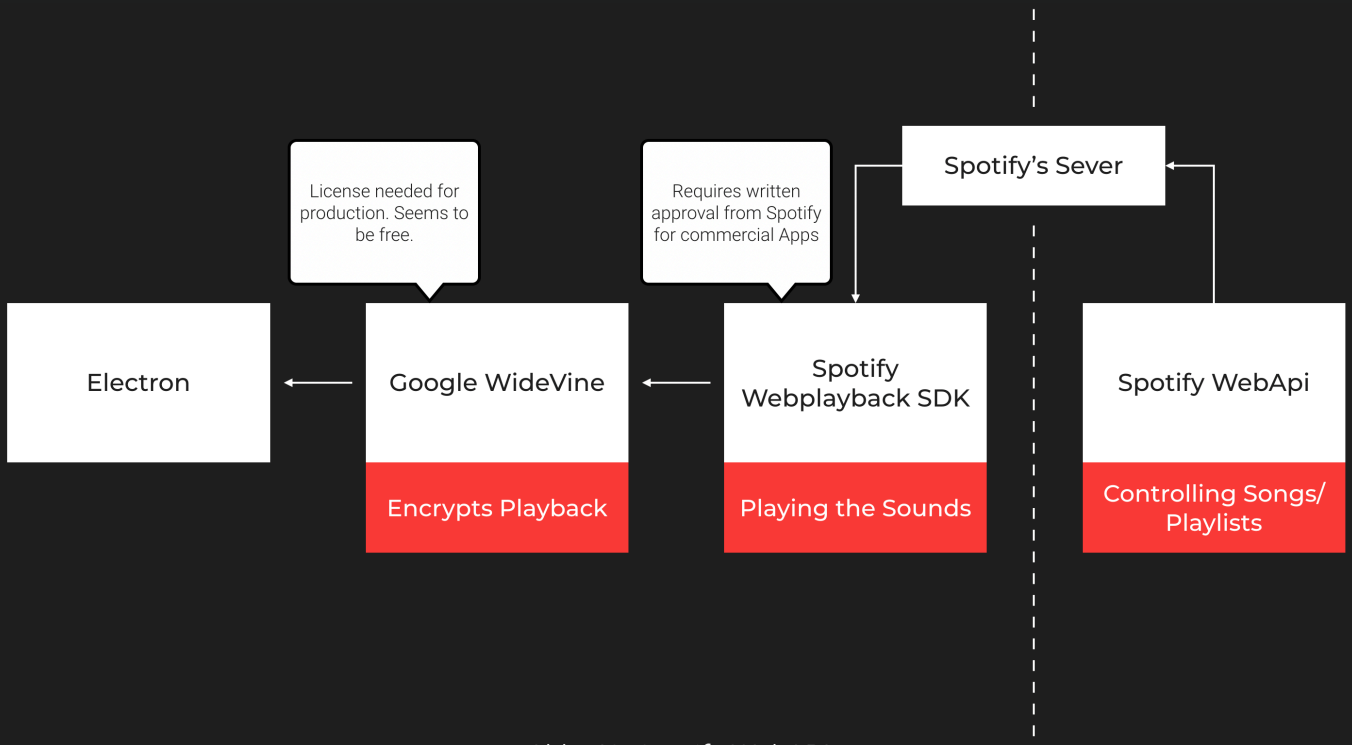


Abb. 43: Spotify WebAPI

Die WebPlayback-SDK erlaubt es dem Entwickler einen Spotify Player, der Musik abspielt in seine Anwendung zu integrieren. Wer aber nun hier nach Funktionen sucht, wie nächsten Song abspielen, ist am falschen Platz. Das wiederum wird nämlich gesteuert über die Spotify WebAPI, die die zentrale Kontrolle über die aktuelle Wiedergabe hat. Das ist so zu verstehen, weil die WebPlayback-SDK von Spotify behandelt wird wie ein weiteres Gerät. So könnte die Steuerung von diesem Gerät, wie es bei Spotify üblich ist, auch auf allen anderen Geräten erfolgen. Somit muss die Steuerung zentral erfolgen und dies wird abgebildet über die WebAPI.

Doch es wird noch komplizierter. Die WebPlayback-SDK verwendet im Hintergrund ein Service, der sich Google WideVine nennt. Dies ist ein Protokoll, welches unter anderem auch von Netflix verwendet wird, um die Wiedergabe von geschützten Material durch Verschlüsselung zu ermöglichen. So erhält die Spotify WebPlayback-SDK die Musikdaten stets verschlüsselt. Dieses System ist standardmäßig in Chrome vorhanden, aber in Chromium nicht, da hier eine Lizenz notwendig ist. Deswegen ist die Auslieferung einer produktionsreifen Electron-Applikation mit Spotifyanbindung nur möglich mit schriftlicher Einverständigung von Spotify und einer Google WideVine Lizenz. Zudem muss ein spezieller Electron Community Build verwendet werden, damit WideVine überhaupt vorhanden ist. Dies hat dafür gesorgt, dass sich im Prototypen auf offline Musik beschränkt wurde. Trotz des Aufwandes ist die Anbindung aber geplant für die finale Applikation.

Analytics und Tracking

Für ein besseres Usertesting haben wir uns dazu entschieden Benutzeraktionen zu tracken. Diese sollen zur Optimierung des Systems dienen und überprüfen, wie die reale Verwendung der Applikation aussieht. Natürlich werden die Nutzer auf das Tracking hingewiesen und keine persönlichen Daten, wie die gesetzten Aufgaben oder Leitfäden gespeichert. Die Trackingdaten enthalten nur grobe Aktionskategorien wie „Timer wurde gestartet“ oder „Applikation wurde beendet“, um die Nutzungsdauer festzustellen.

Abb. 44: Tracking

Für den Prototypen, der nur an ausgewählte Tester geschickt wurde, wird stets in den jeweiligen E-Mails oder Gesprächen, sowie auch auf der Website klar kommuniziert, dass ein Tracking stattfindet. Für die finale Applikation ist geplant, dass das Tracking prinzipiell in den Einstellungen deaktiviert ist. Der Nutzer müsste es hierbei bewusst aktivieren, falls er die Entwickler unterstützen möchte.

Für das Speichern der Trackingdaten wurde auf Firebase Realtime Database gesetzt. Das Praktische an dieser NoSQL Datenbank ist, dass das Speichern von Daten im JSON-Format über Javascript sehr einfach funktioniert. Diese Möglichkeit des schnellen und unkomplizierten Aufsetzen dieser Datenbank hat sie für den Prototypen sehr attraktiv gemacht. Zudem ist sie für unsere Datenmengen noch kostenlos verfügbar. Außerdem besitzt für die Produktions-App auch nützliche Eigenschaften wie „Offline Persistence“, die dafür sorgen, dass die Daten ohne Internetverbindung offline gespeichert werden und dann später synchronisiert werden, wenn wieder Internet vorhanden ist.

Trotzdem müsste für die finale Applikation erst mal eine Evaluierungsphase erfolgen, in der verschiedene Datenbanklösungen verglichen werden. Hierbei wären vor allem Features, die dabei helfen die gespeicherten Daten auch sofort auszuwerten sehr interessant.

Deployment und Hosting

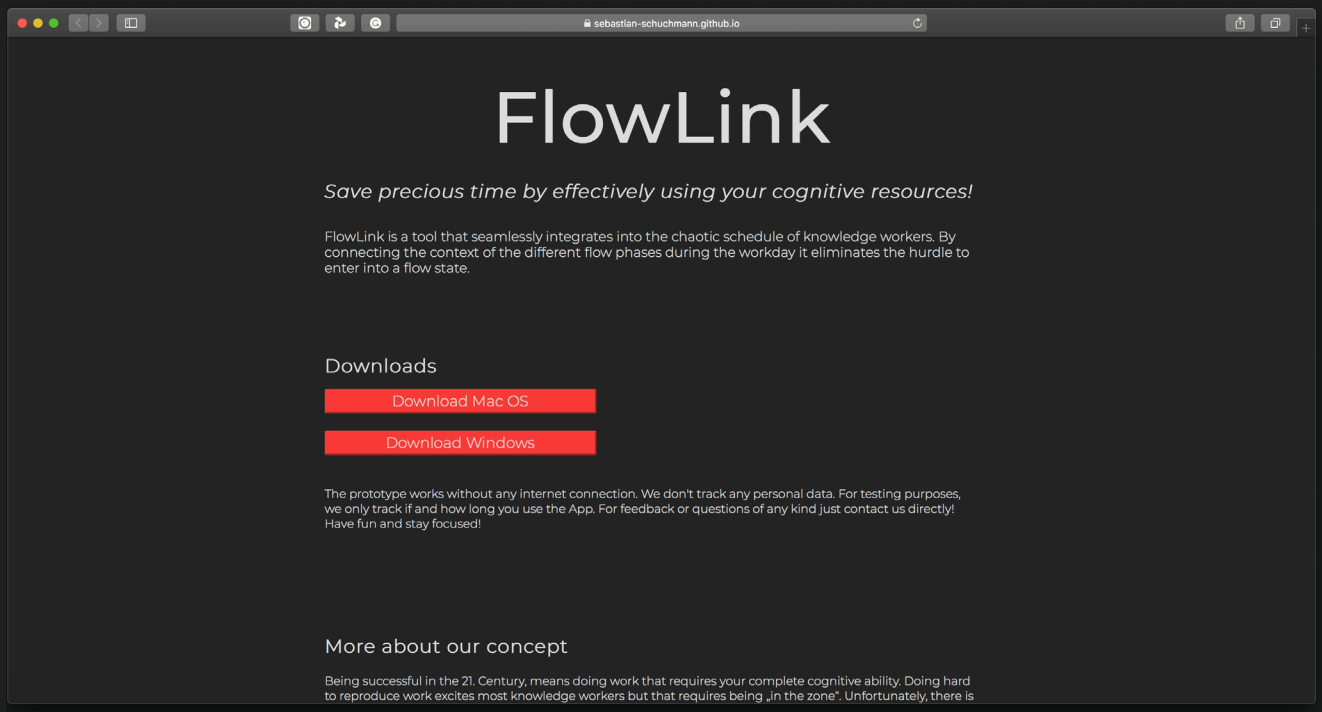


Abb. 45: FlowLink Website

Damit die Applikation auch für die Nutzer zugänglich ist, wird eine Kombination aus Amazons S3 und Github Pages verwendet. S3 ist Teil von Amazons AWS Angebot und ist das Standard Tool, wenn es darum geht Dateien für den Download anzubieten. Der Download wird dann noch über ein Web-Frontend zugänglich gemacht. Hier bietet Github die Möglichkeit, ein dort vorhandenen Repository zu hosten. Es ist die uns schnellste, bekannte Möglichkeit, eine Website zu hosten und deswegen für den Download des Prototypen ideal.

Aufbau der Electron Anwendung

Der Kern der Applikation läuft zwar über Electron, aber das erklärt noch nicht wie unser System funktioniert. Im Aufbau der Anwendung wurde objektorientiert gearbeitet. Das heißt, die Funktionen der Anwendung wurde in verschiedene Klassen und Objekte gekapselt. Jede Klasse hat ihren eigenen Verantwortungsbereich und bekommt nur die Funktionalitäten zugeteilt, die für diesen Verantwortungsbereich notwendig sind. Diese Struktur sorgt einerseits für Übersicht und erleichtert zudem die Zusammenarbeit an der Anwendung.

Zuerst wurden Flowcharts erstellt, um den groben Ablauf der Anwendung zu definieren. Daraus wurden dann Verantwortungsbereiche abgeleitet und schließlich ein Klassendiagramm erarbeitet. Für komplexere Abläufe wurden schließlich noch Sequenzdiagramme erstellt, die gut darin sind die Kommunikation zwischen Komponenten zu definieren. Zu allen folgenden Diagrammen sollte gesagt sein, dass diese konstant überarbeitet wurden und hier nur die finalen Versionen zu sehen sind. Wir sehen Diagramme als flexible Werkzeuge, die bei der Umsetzung durchgehend Änderung unterliegen, da sich ständig neue Anforderungen und Erkenntnisse ergeben.

Programmablauf

Es wurde ein Beispielhafter Programmablauf skizziert. Dieser hat nicht das Ziel jedes Detail der Applikation zu enthalten, sondern die wichtigsten Schritte der Applikation zu definieren. Er bildet die Grundlage, um die Klassen zu definieren.

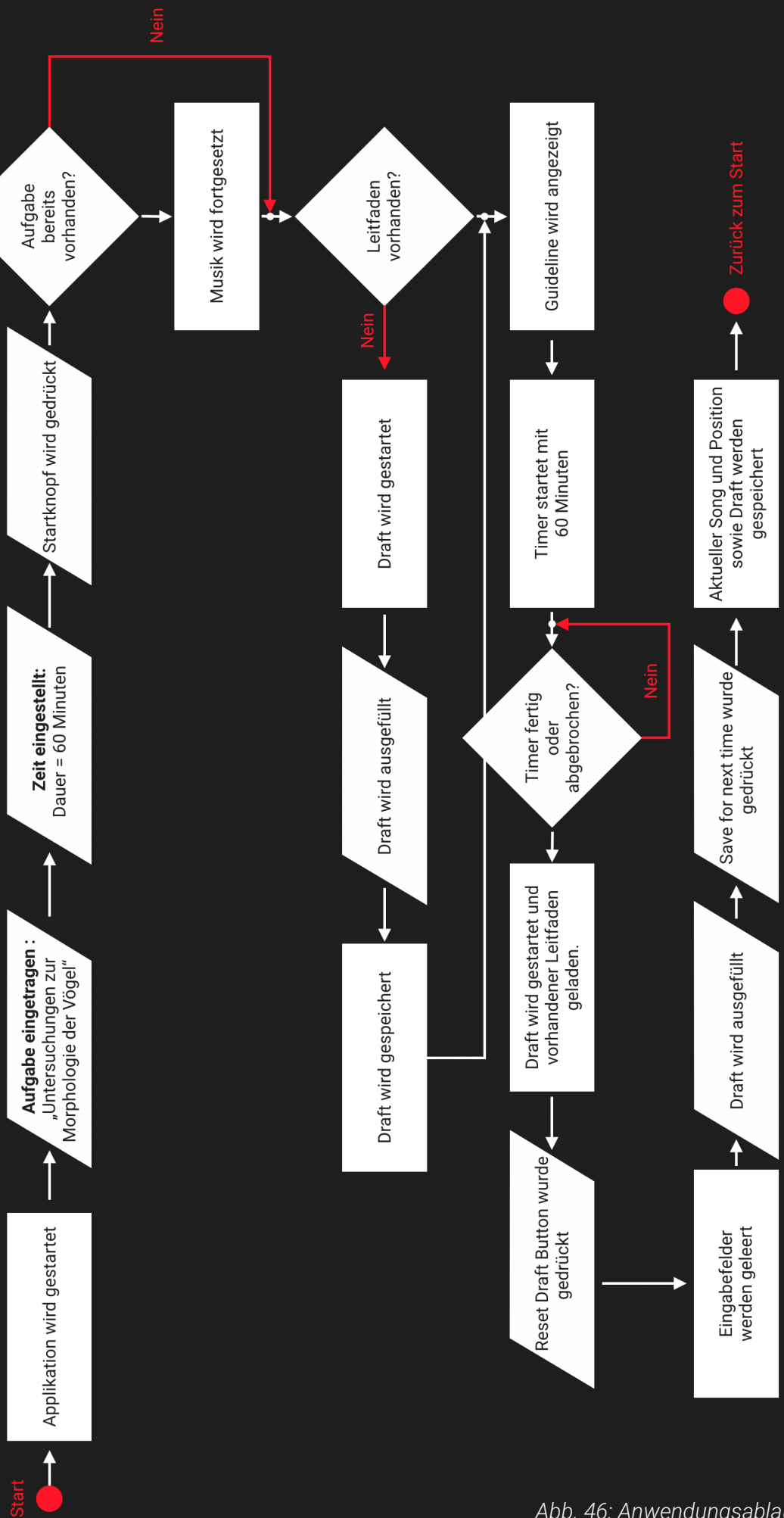
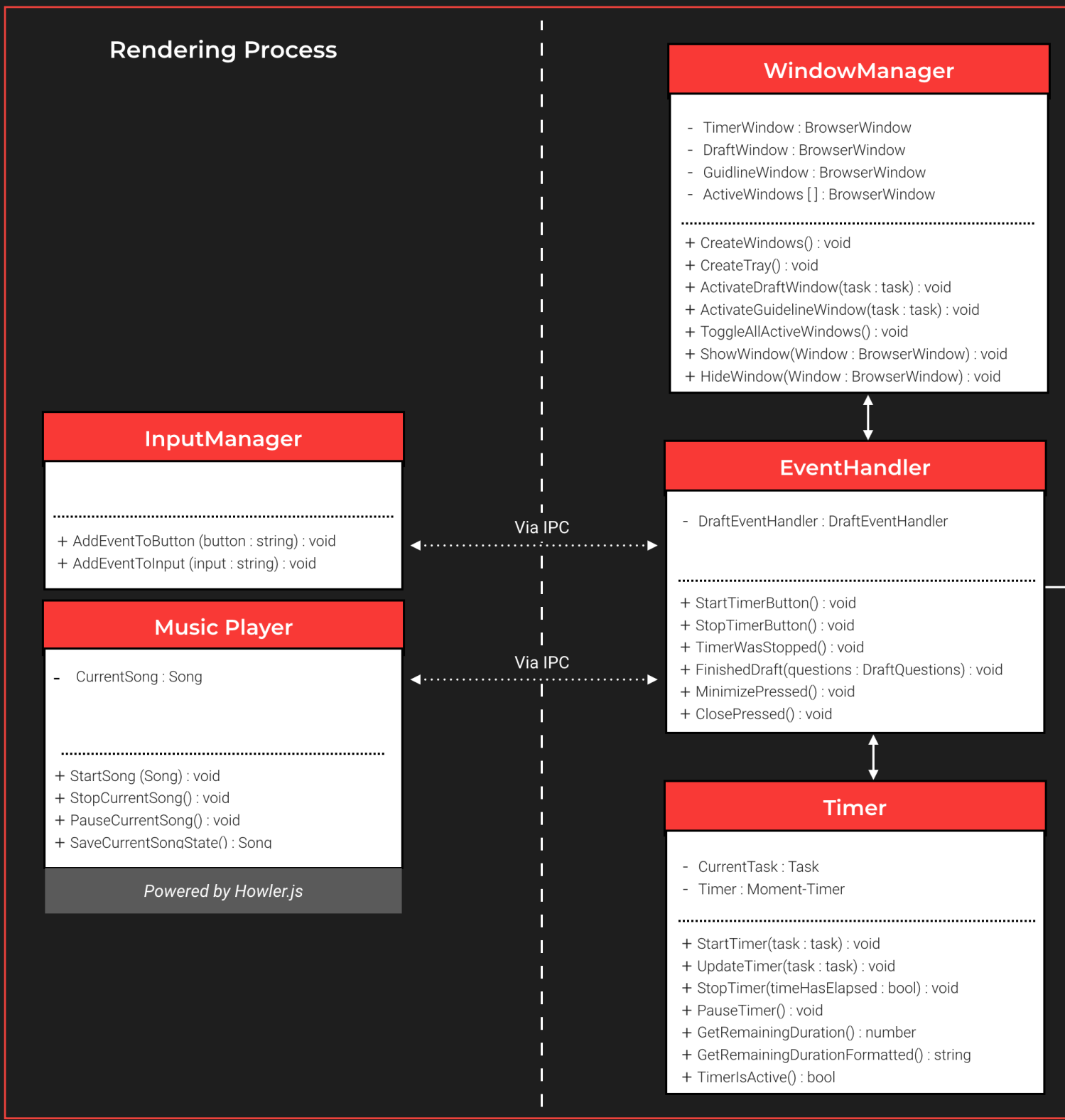


Abb. 46: Anwendungsablauf FlowChart

Verantwortungsbereiche und Aufteilung

Die wichtigsten Klassen des Systems sind der WindowManager, EventHandler, InputManager und der Timer. Sie bestimmen den großen Teil des Programmablaufs und enthalten die meiste Logik. Das ist der Grund, warum sich auf diese Klassen auch bei den Erklärungen fokussiert wird.



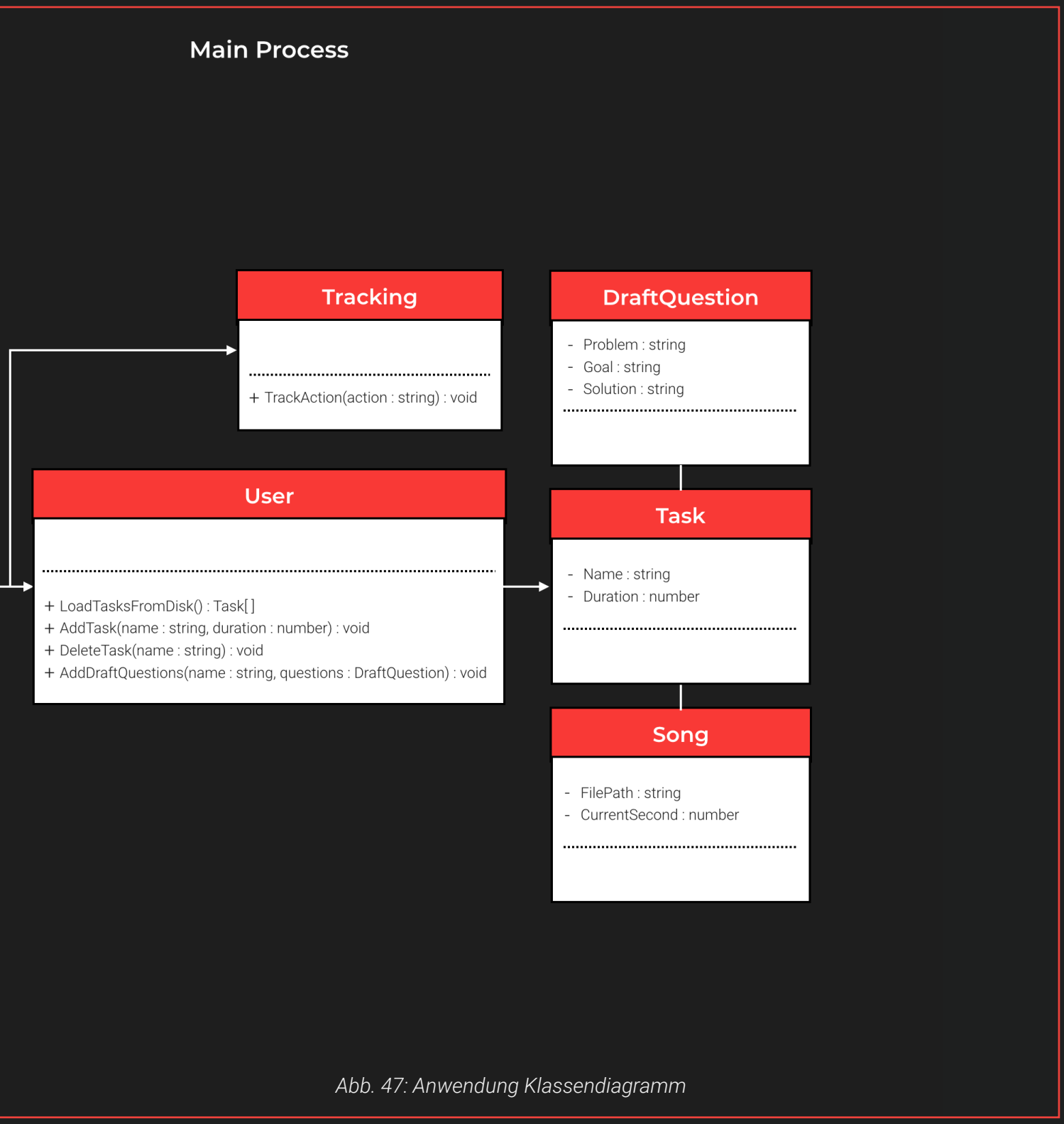


Abb. 47: Anwendung Klassendiagramm

WindowManager

Der Einstiegspunkt der Applikation ist die „main.js“ und dort wird der WindowManger erstellt. Er ist für die Browserfenster verantwortlich, in denen das Interface gerendert wird.

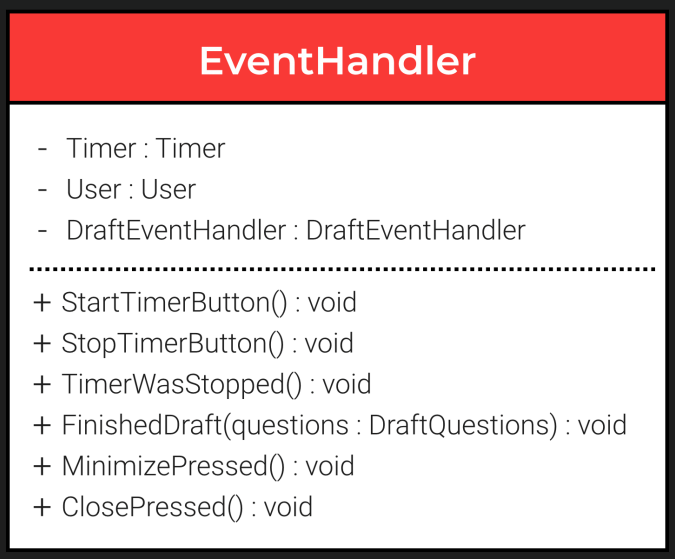
Er besitzt das Hauptfenster des Timers, des Drafts und der Guideline, mit jeweils eigenständigen Fenstern. Somit enthalten seine Funktionalitäten auch alles was mit diesen Fenstern zu tun hat, wie das Schließen, Minimieren und Ausblenden der einzelnen Fenster. Der WindowManager befindet sich im Hauptprozess des Node.js Servers. Für die Kommunikation zwischen dem Hauptprozess und dem Renderprozess ist der EventHandler verantwortlich.

EventHandler

Für die Kommunikation zwischen dem Hauptprozess des Node.js Servers in den Renderprozessen der Browserfenster existiert in Electron ein Interprocess-Communication System. Dies ist ein asynchrones Messaging-System, mit dem Nachrichten zwischen Prozessen über verschiedene Kanäle gesendet werden können. Die verschiedenen Kanäle werden einfach über Strings identifiziert.

Damit hier keine Fehler durch falsche Benennung entstehen, wurde sich dafür entschieden zusätzlich ein JSON, dass sich „ipcEvents“ nennt, zu pflegen, in dem alle Eventkanäle festgehalten werden. Hierdurch kann die Autovervollständigung des Code-Editors sicherstellen, dass die Eventbenennung richtig erfolgt.

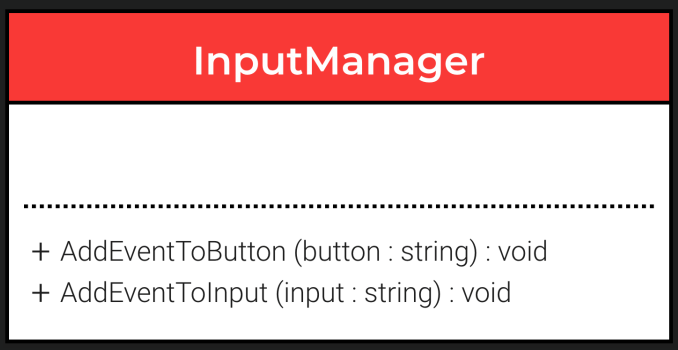
Das Pendant des EventHandlers auf Seite des Renderprozesses ist der InputManager.



InputManager

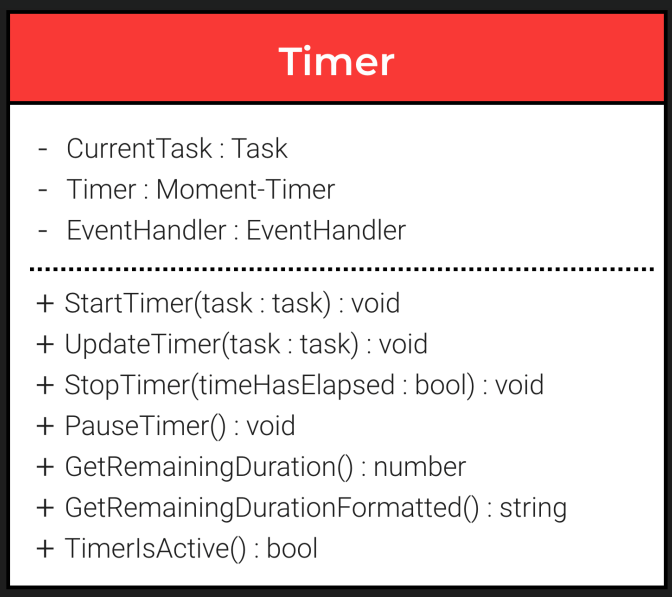
Der Inputmanager ist der Kommunikationspartner für den EventHandler. Er befindet sich auf Seite des Renderprozesses und ist somit in der Lage den DOM zu lesen und zu verändern.

Die Aufgabe des InputManagers ist es auf die DOM-Events zu hören wie zum Beispiel Buttonklicks und sie über IPC an den EventHandler weiterzuleiten, damit die Programmlogik darauf reagieren kann.



Timer

Der Timer befindet sich auf der Seite des Hauptprozesses, damit er Browserfenster unabhängig funktioniert. Der Timer ist in der Lage beliebige Zeiten zu stoppen und sie formatiert auszugeben.



Ein paar Worte zu Modularität

Die im Prototyp definierten Klassen wurden nach dem Prinzip der Modularität entworfen. So waren, zumindest konzeptionell, zum Beispiel die Funktionalitäten des Timers oder des MusicPlayers komplett unabhängig von der Anwendung. Doch in der eigentlichen Implementierung wird klar, dass diese Klassen Methoden besitzen, die nicht allgemein, sondern speziell für das System entworfen wurden.

Deswegen ist an dieser Stelle klar ein Refactoring notwendig. Dieses Problem könnte gelöst werden, indem die Funktionalitäten des MusicPlayers zum Beispiel in zwei Klassen aufgesplittet werden. Die eine Klasse könnte die allgemeinen Funktionalitäten eines Musikspielers enthalten und die andere den allgemeinen Musikspieler so erweitern, dass er speziellen Anforderungen des Systems erfüllt. Damit wäre das System wieder modular, weil der allgemeine Musikspieler schnell durch ein beliebiges Streamingsystem wie Spotify ausgetauscht werden könnte, solange die Grundfunktionalitäten wie Abspielen und Stop implementiert sind.

Interne Abläufe (Sequenzdiagramm)

Der folgende Ablauf wurde erstellt, um die Kommunikation zwischen den wichtigsten Klassen besser zu definieren. In diesem beispielhaften Ablauf wurde bereits eine Aufgabe eingetragen und eine Zeit eingestellt. Der Ablauf beginnt mit dem Drücken auf den Startknopf. In diesem Ablauf sind natürlich noch weiter Akteure vorhanden wie der Musikplayer, die zur Vereinfachung weggelassen wurden.

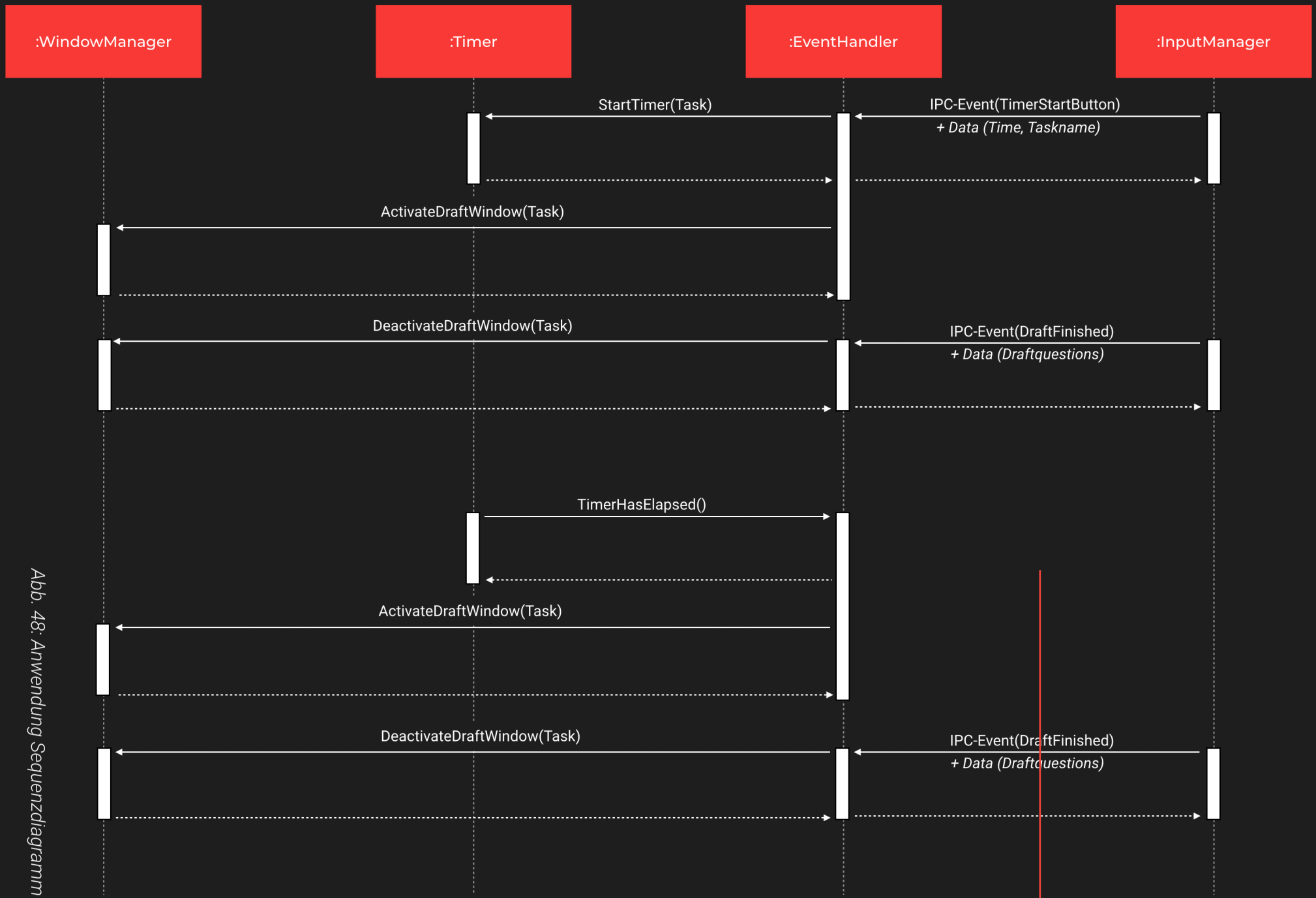


Abb. 48: Anwendung Sequenzdiagramm

Technik Ausblick

Interessante Architektur Inspiration für die Zukunft bietet zum Beispiel das Xray-Editor Projekt von Github, dass leider inzwischen nicht mehr weitergeführt wird. Für diesen Editor wurde sich auch für Electron entschieden, um Webtechnologien für das UI verwenden zu können.

Die Entwickler hatten aber zudem anspruchsvolle Performancestandards. Deswegen wurde sich dafür entschieden die Hauptlogik in Rust abzubilden und nur das Frontend Javascript basiert aufzubauen. Die Entwickler begründen diese Entscheidung damit, dass es leichter ist, performantes Rust zu schreiben als performantes Javascript. Der Rust Quelltext wird dann als Maschinencode durch WebAssembly verwendbar.

Dieser Fokus auf Performance ist für FlowLink wahrscheinlich nicht notwendig, denn die Reduktion der Dateigröße und das Einsparen von RAM sind größerer Probleme, die aber auch mit dieser Architektur per se nicht gelöst sind. Für zukünftige Optimierung ist die Verwendung von WebAssembly dennoch sehr spannend.

Des Weiteren können wir uns Vorstellen das Testing mit funktionalen Tests zu erweitern. Funktionale Tests sind in der Lage User Input zu simulieren und bieten deswegen eine gute Möglichkeit zumindest Teile des User Testings zu optimieren.

Natürlich können so keine Design-Entscheidungen getroffen werden, aber um zum Beispiel Eingabefelder auf diversen Input zu überprüfen und potenzielle Bugs zu enttarnen, reicht es. Hierfür könnte Cypress.io ein potenzielles Tools sein.

Das Zukunftsbild

Wir hatten uns das Ziel gesetzt die Hürde in den Flow zu eliminieren und dafür FlowLink gebaut. Doch, wenn wir an die Zukunft denken, fragen wir uns warum nach der Hürde aufhören? In der Optimierung des Flows selbst sehen wir auch viel Potential. Wenn wir nochmal einen Schritt zurückgehen und ein Blick auf die Recherche werfen, sehen wir dort den perfekten Leitfaden, um dies zu tun.

Csikszentmihályi beschreibt drei Bedingungen, die erfüllt sein müssen, damit der Flow Zustand bei der Arbeit erreicht werden kann:

1. Das Ziel ist klar definiert
2. Sofortiges Feedback
3. Ein Gleichgewicht zwischen Fähigkeiten und Kapazität

Die klare Definition des Ziels haben wir mit dem Leitfaden abgehakt. Doch ein sofortiges Feedback oder das richtige Gleichgewicht wird komplett von der Aufgabe selbst bestimmt. Damit wird schnell klar, wo eingegriffen werden muss, bei der Aufgabe selbst. FlowLink 2.0 könnte zum Beispiel die Schwierigkeit der Aufgabe dynamisch so anpassen, dass der Nutzer immer optimal herausgefordert ist.

Zudem sollte FlowLink 2.0 natürlich immer und überall einsatzbereit sein. Hierfür können wir uns alles von Augmented Reality bis hin zu Gehirnimplantaten vorstellen. Letzteres könnte zudem dabei helfen zu Messen wie stark der Nutzer sich im Flow befindet.
Die Zukunft bleibt spannend und wir sind uns sicher FlowLink wird ein Teil davon sein.

ABBILDUNGSVERZEICHNIS

Abb. 1: Timeline 10

Abb. 3: Value vs Friction 13

Abb. 4: Interview-Leitfaden 14

Abb. 5: User Journey 17

Abb. 6: Persona Charlez 19

Abb. 7: Job Characteristic Theory 22

Abb. 8: Aufmerksamkeitsrückstände 23

Abb. 9: Das Flow-Prinzip 24

Abb. 10 nach Duhigg: Habit Cycle 26

Abb. 11 nach Wendelien Van Eerde: The progress of procrastination 27

Abb. 12 nach Shukla: Breaking the habit 28

Abb. 13: Charakter der Anwendung 31

Abb. 14: Der Ablauf der Roten Fadens 33

Abb. 16: Wireframe 2 Timer Musik 34

Abb. 15: Wireframe 1 Timer Start 34

Abb. 18: Wireframe 4 Timer Taskfeld 35

Abb. 17: Wireframe 3 Timer 90 min 35

Abb. 19: Der Rote Faden 36

Abb. 20: Unsere Version der Heldenreise 37

Abb. 22: Wireframe Leitfaden 39

Abb. 21: Wireframe Draft 39

Abb. 23: Ablauf der Rahmen 40

Abb. 24: Wireframe Boot-Up 41

Abb. 25: Wireframe 1 Recap 43

Abb. 26: Wireframe 2 Recap 44

Abb. 27: Wireframe 3 Weekly-Recap 45

Abb. 28: Wireframe 1 Framing 46

Abb. 29: Wireframe 2 Framing 47

Abb. 30: Wireframe 3 Framing 48

Abb. 31: Sprechmelodie 50

Abb. 32: Das Logo 55

Abb. 33: Icons Icons Icons 58

Abb. 34: Der Prototyp Start 60

Abb. 35: Der Prototyp Draft 60

Abb. 36: Technology Stack 61

Abb. 37: Technology Stack Update 62

Abb. 38: Electron Architektur 68

Abb. 39: Breitband in Deutschland 69

Abb. 40: Test Driven Development 70

Abb. 41: Testing Codebeispiel Mocha 71

Abb. 42: Testing Codebeispiel Ava 72

Abb. 43: Spotify WebAPI 75

Abb. 44: Tracking 76

Abb. 45: FlowLink Website 77

Abb. 46: Anwendungsablauf FlowChart 79

Abb. 47: Anwendung Klassendiagramm 81

Abb. 48: Anwendung Sequenzdiagramm 85

LITERATURVERZEICHNIS

Pausen

Linhart, Lisa-Marie (**2012**). Pause machen – aber richtig! Anleitung zur perfekten Entspannung.
Abgerufen am 04.07.2019 unter:
https://www.karriere.at/blog/pausen-buero.html

Inconvenient Design

Rekimoto, J., & **Tsujita**, H. (**2014**). Inconvenient interactions: an alternative interaction design approach to enrich our daily activities. Proceedings of the 2014 International Working Conference on Advanced Visual Interfaces - AVI '14, 225–228. https://doi.org/10.1145/2598153.2598196

Weaver, Jesse (**2019**). The Value of Inconvenient Design.
Abgerufen am 17.06.2019 unter:
https://medium.com/s/user-friendly/the-power-of-inconvenience-f0ae1773dd77

Ramirez, Juan J. (**2017**). Designed Inconveniences and UX Patterns that can't be taught.
Abgerufen am 17.06.2019 unter:
https://uxdesign.cc/designed-inconveniences-ux-patterns-that-cant-be-taught-998eae9c4546

Zielgruppe

Bittlinger, Robin D. (**2019**). Sowohl zu viel, als auch zu wenig Einfühlungsvermögen und erschwert den Umgang mit sich selbst und anderen Menschen. Die Licht- und Schattenseiten der Empathie.

Arbeit

Hackman, J. R. (**1980**). Work redesign and motivation. Professional Psychology, 11(3), 445-455.
http://dx.doi.org/10.1037/0735-7028.11.3.445

Deep-Work

Newport, C. (**2016**). Deep work: Rules for focused success in a distracted world. Hachette UK.

Pyöriä, P. (**2005**). "The Concept of Knowledge Work Revisited". Journal of Knowledge Management. 9(3): 116–127.
doi:10.1108/13673270510602818.

Reinhardt, W.; **Schmidt**, B.; **Sloep**, P.; **Drachsler**, H. (**2011**). "Knowledge Worker Roles and Actions – Results of Two Empirical Studies". Knowledge and Process Management. 18 (3): 150–174.
doi:10.1002/kpm.378. hdl:1820/3523

Ericsson, K. A., & **Charness**, N. (**1994**). Expert performance: Its structure and acquisition. American psychologist, 49(8), 725.
Leroy, S. (**2009**). Why is it so hard to do my work? The challenge of attention residue when switching between work tasks. Organizational Behavior and Human Decision Processes, 109(2), 168-181.

Newport, C. (**2016**). Deep work: Rules for focused success in a distracted world. Hachette UK.

Gifford, J. (**2016**). The Rule of 52 and 17: It's Random, But it Ups Your Productivity.
Abgerufen von:
https://www.themuse.com/advice/the-rule-of-52-and-17-its-random-but-it-ups-your-productivity

Das Flow-Prinzip

Csikszentmihályi, M. (**1997**). Finding flow. The psychology of engagement with everyday life. New York: Basic Books.

Csikszentmihályi, M. (**1990**). Flow: The Psychology of Optimal Experience. Harper & Row. ISBN 978-0-06-016253-5.

Csikszentmihályi, M. (**1988**), "The flow experience and its significance for human psychology", in Csikszentmihályi, M. (ed.), Optimal experience: psychological studies of flow in consciousness, Cambridge, UK: Cambridge University Press, pp. 15–35, ISBN 978-0-521-43809-4

Moneta, Giovanni (**2012**). "On the Measurement and Conceptualization of Flow". In Engeser, Stefan (ed.). Advances in Flow Research. New York: Springer. ISBN 978-1461423584.)

Gewohnheiten

Alps, Nicole (**2012**). Aus der Gehirnforschung: Wie du eine Gewohnheit änderst.
Abgerufen am 09.07.2019 von:
https://zeitzuleben.de/gewohnheit-aendern/

Ewert, Katrin (**2019**). Warum unser Gehirn Routinen liebt.
Abgerufen am 09.07.2019 von:
https://www.planet-wissen.de/gesellschaft/psychologie/gewohnheiten/gewohnheiten-hirnforschung-100.html

NLP-Zentrum Berlin (2017). Erfolgreich mit Gewohnheiten.
Abgerufen am 09.07.2019 von:
https://nlp-zentrum-berlin.de/infothek/nlp-psychologie-blog/item/erfolgreich-mit-gewohnheiten

Shukla, Aditya **(2017).** Stop Procrastinating With This Research-Based Psychological Method.
Abgerufen am 08.07.2019 von:
https://cognitiontoday.com/2017/08/overcoming-procrastination-with-a-habit-changing-psychological-method/

Crozier, Niall **(2018).** Northern Review: The Power of Habit.
Abgerufen am 12.07.2019 von:
https://www.northernslant.com/northern-review-power-habit/

Duhigg, Charles (**2012**). Die Macht der Gewohnheit. Warum wir tun, was wir tun. Berlin: Bloomsbury Verlag. ISBN: 978-3-8270-0957-9

Wood, Wendy **(2018).** Macht der Gewohnheit.
Abgerufen am 09.07.2019 von:
https://www.spektrum.de/alias/hirnforschung/macht-der-gewohnheit/1134318

Prokrastination

Shukla, Aditya **(2019). Y**ou Procrastinate Because Of Emotions, Not Laziness. Regulate Them To Stop Procrastinating!
Abgerufen am 15.07.2019 von:
https://cognitiontoday.com/2019/05/you-procrastinate-because-of-emotions-not-laziness-regulate-them-to-stop-procrastinating/

Das Recap

Borsook, Terry K., **Higginbotham-Wheat**, Nancy (**1992**). A Psychology of Hypermedia: A Conceptual Framework for R&D.

Delegation of Rights and Non-Violation Statement

FlowLink is a project, developed in the framework of the study course Interactive Media Design of Hochschule Darmstadt - Faculty of Media, during the winter Term 2018/2019. It is a work of Sebastian Schuchmann and Robin Bittlinger. It was mentored by Prof. Claudius Coenen, Prof. Andrea Krajewski and Prof. Dr. Martin Leissler.

I herewith delegate the non-exclusive and timewise non-restricted rights to publish and present the results of the project FlowLink to the Professors of Hochschule Darmstadt and to the coaches directly connected to the academic supervision of this project, named above. In the same time, I declare that with the project no intellectual properties rights of third parties have been harmed.

Location, Date

Signature