Storyplan

KW 19

Erstellt von:	**Überprüft von:**
Miguel Friesen, Noah Schwenk	Miguel Friesen, Noah Schwenk, Stephan Braun

Version	Effektiv ab	Beschreibung / Änderungen
1.0	15.04.2026	Allgemeine Story Erstellung
2.0	27.04.2026	Anpassungen der Stories nach Abschluss von Wochenscheibe 1
3.0	03.05.2026	Anpassungen der Stories nach Abschluss von Wochenscheibe 2
4.0	10.05.2026	Anpassungen der Stories nach Abschnluss von Wochenscheibe 3

Allgemeines

Dieses Dokument beinhaltet die Planung der Stories, die in den fünf Wochen der Implementierung realisiert werden sollen.

Jeder Userstory werden zu erledigende Aufgaben und für die Implementierung verantwortliche Personen festgehalten.

Gearbeitet wird nach dem Prinzip des Pair Programmings. Dabei entwickeln zwei Personen gemeinsam und wechseln sich dabei ab zwischen dem Schreiben von Code und dem Überprüfen des Codes. Dies führt zu einer besseren Codequalität und erlaubt einen Wissenstransfer zwischen den Beteiligten.

Die detaillierte Zerlegung der User Stories in technische Implementierungsaufträge (Sub-Tasks) sowie deren granulare Spezifikation erfolgen direkt im **GitLab Issue-Tracker unserer Repositories (Backend und Frontend)**.

Unsere teams sind wie folgt aufgeteilt:

Team A: Tom, Maurice

Team B: Stephan, Noah, Miguel

Story-Planung

Wochenscheibe 1: Fundament & Datenintegration

User Story 1: Kognitives Arbeitsprofil & Mandanten-Zeitslots

- **User Story:** Als Nutzender möchte ich mein kognitives Arbeitsprofil (Kapazitätsgrenze, Pausen, exklusive Zeitslots) dauerhaft in meinem Profil speichern, damit das System meine persönliche Belastungsgrenze bei der späteren Planung berücksichtigt.
- **Status:** Nicht bestanden / In Wochenscheibe 2 verschoben
- **Akzeptanzkriterien:**
    - Die maximale Dauer zusammenhängender Arbeitsphasen ist auf Werte zwischen 2 und 4 Stunden pro Tag beschränkt und wird vom Backend zwingend validiert.

- Organisationsspezifische Zeitslots können angelegt werden und überschneiden sich nicht (Validierungsprüfung schlägt bei Konflikt fehl).
- Die erfassten Limits werden fehlerfrei und strukturiert in der Firestore-Datenbank persistiert.
- Freelancer haben exklusiv im Dashboard eine angepasste Ansicht mit zwingender Unternehmensauswahl.

- **Zugehörige operative Zuweisung:**
  - **Team A** (Frontend-Integration): UI-Anbindung und Validierung der Formularmasken für das kognitive Profil.
  - **Team B** (Backend-Logik): Implementierung der REST-Endpunkte und Persistierung des 2-4-Stunden-Limits sowie der exklusiven Organisations-Zeitslots.
- **Erforderliche Testarten & Testübersicht:**
  - *Automatisierte Unit-Tests (Backend):* Zwingende Validierung, dass die eingegebene Arbeitsphase für tiefe Wissensarbeit die harte Grenze von 2 bis 4 Stunden pro Tag nicht überschreitet. Prüfung auf Überschneidungsfreiheit von Organisations-Zeitslots.
  - *Manuelle Systemtests:* Klickpfad-Prüfung im UI zur Erfassung der Profil-Parameter und Verifikation der korrekten Persistenz in der Firestore-Datenbank.
- **Definition of Done (DoD):** Die Akzeptanzkriterien sind vollständig erfüllt. Komponententests (Backend/Frontend) existieren gemäß Testkonzept, sind in die Testsuite integriert und passieren die GitLab CI/CD-Pipeline fehlerfrei. Die manuellen Systemtests wurden dokumentiert und bestanden. Der Code ist in das Release-Inkrement integriert.

User Story 2: Parametrisierte Aufgabenerfassung & Blocker

- **User Story:** Als Nutzender möchte ich neue Aufgaben (inklusive Energiebedarf und Abhängigkeiten) sowie statische Blocker erfassen, damit diese für die spätere Planung zentral zur Verfügung stehen.
- **Status:** Nicht bestanden / In Wochenscheibe 2 verschoben
- **Akzeptanzkriterien:**

- Pflichtparameter (Priorität, Energiebedarf, Abhängigkeiten, minimale/maximale Chunk-Größe) werden bei der Eingabe strukturiert ausgelesen und serverseitig validiert.
- Statische Blocker werden zwingend auf Überschneidungsfreiheit geprüft.
- C#-Endpunkte (POST/PUT) verarbeiten die DTOs fehlerfrei und binden die Prototyp-Eingabemaske erfolgreich an die Cloud-Datenbank an.

- **Zugehörige operative Zuweisung:**
  - **Team A** (Frontend-Integration): Implementierung der Erfassungsmasken für Tasks und statische Blocker.
  - **Team B** (Backend-Logik): CRUD-Operationen, Parameter-Validierung (Energiebedarf, Flow-Schutz) und strikte Überschneidungsprüfung.

- **Erforderliche Testarten & Testübersicht:**
  - *Integrationstests (Backend/Datenbank):* Validierung der CRUD-Operationen (POST/PUT) für Aufgaben und Ausfallzeiten (Blocker) gegen den Firestore-Emulator.
  - *Automatisierte Unit-Tests:* Strikte Prüfung der Pflichtparameter (Energiebedarf, Flow-Schutz/minimale Chunk-Größe) auf korrekte Verarbeitung.
  - *Manuelle Systemtests:* Formularvalidierung im UI und Verifikation, dass bei der Eingabe keinerlei Team-Zuweisungen (Schutz des individuellen Selbstmanagements) vorgenommen werden können.
- **Definition of Done (DoD):** Die Akzeptanzkriterien sind vollständig erfüllt. Komponententests (Backend/Frontend) existieren gemäß Testkonzept, sind in die Testsuite integriert und passieren die GitLab CI/CD-Pipeline fehlerfrei. Die manuellen Systemtests wurden dokumentiert und bestanden. Der Code ist in das Release-Inkrement integriert.

- **Bilanz & Retrospektive (Ende Wochenscheibe 1):** Die formale Definition of Done (DoD) wurde verfehlt. Der Code wurde nicht in das Release-Inkrement integriert (fehlender Merge der Backend-Branches) und die UI-Anbindung der Formularmasken durch Team A wurde nicht vollzogen. Somit konnte kein lauffähiges Release für das Weekly Build und keine funktionsfähige Demoversion auf der Webseite bereitgestellt werden. Die unfertigen technischen Implementierungsaufträge (Sub-Tasks) werden als technische Schuld in die Wochenscheibe 2 übertragen.

Wochenscheibe 2: Die Planungs-Engine

- **Einschub: Technischer Refactoring- & Integrations-Sprint**
  - **Ziel:** Behebung der technischen Schulden aus Wochenscheibe 1 zur Herstellung eines lauffähigen Release-Bündels (Weekly Build) vor dem Start der Planungs-Engine.
  - **Zugehörige operative Zuweisung (Neupriorisierung):**
    - **Team A (Frontend & Release):** Zwingender Abschluss der ausstehenden Frontend-Integration Abfangen und Visualisieren von Conflict-Fehlern als transparente Warnmeldungen.
    - **Team B (Backend & Testinfrastruktur):** Sofortige Auflösung der Merge-Konflikte im Main-Branch.
  - **Status:** Erfolgreich abgeschlossen (Vorher: In Bearbeitung)

User Story 3: Algorithmische Zuweisung & Abhängigkeitsauflösung

- **User Story:** Als Nutzender möchte ich, dass meine Aufgaben auf Basis ihrer Abhängigkeiten und Prioritäten logisch korrekt sortiert werden, damit mein generierter Arbeitsplan sinnvolle Reihenfolgen einhält und nicht an Zirkelbezügen scheitert.

- **Status:** Bestanden / Abgeschlossen. Die Implementierung wurde erfolgreich durchgeführt und die aufgestauten technischen Schulden wurden abgebaut (Vorher: Verzögert (Ressourcen-Engpass))

- **Akzeptanzkriterien:**

  - Der Planungsalgorithmus ist modular (z. B. über ein Strategy Pattern) in die Architektur integriert.

  - Das System löst bei der Planerstellung logische Aufgaben-Abhängigkeiten (Vorgänger- und Nachfolger-Relationen) fehlerfrei auf und blockiert die Zuweisung mit einer Warnung, falls sich Aufgaben fälschlicherweise gegenseitig als Voraussetzung blockieren (Zirkelbezüge).

  - Nur Aufgaben ohne offene Vorbedingungen durchlaufen den Algorithmus und werden nach einem berechneten Urgency-Score in freie Zeitslots verteilt.

  - Der Code muss so optimiert sein, dass die topologische Sortierung auch bei 50 verknüpften Tasks ohne merkliche Verzögerung (< 1s) erfolgt.

  - Bei der konkreten Implementierung des Algorithmus sind kleinere fachliche Fragen aufgekommen. Diese blockieren das aktuelle Release-Inkrement nicht, werden aber zwecks Feinjustierung im weiteren Verlauf geklärt.

- **Zugehörige operative Zuweisung:**
    - **Team B** (Backend-Engine): Implementierung der Graphentheorie (DAG) zur topologischen Sortierung und Abhängigkeitsauflösung.
    - **Team A** (Ressourcen-Shift → Backend): Da das UI hierfür obsolet ist, wechselt Team A in die Backend-Entwicklung und unterstützt Team B bei der Berechnung der Priorisierungs-Warteschlangen (Urgency-Scores).
- **Erforderliche Testarten & Testübersicht:**
    - *Automatisierte Unit-Tests (Backend):* Verifikation der topologischen Sortierung (Directed Acyclic Graph) und zwingende mathematische Prüfung auf Zirkelbezüge.
    - *Performance-Tests (Integration):* Sicherstellung, dass die Abhängigkeitsauflösung von 50 Tasks ohne merkliche Verzögerung (< 1s) prozessiert wird.
- **Definition of Done (DoD):** Die Akzeptanzkriterien sind vollständig erfüllt. Komponententests (Backend/Frontend) existieren gemäß Testkonzept, sind in die Testsuite integriert und passieren die GitLab CI/CD-Pipeline fehlerfrei. Die manuellen Systemtests wurden dokumentiert und bestanden. Der Code ist in das Release-Inkrement integriert.

User Story 4: Flow-Schutz & Kognitives Kapazitätslimit

- **User Story:** Als Nutzender möchte ich, dass der Planungsalgorithmus meine mentale Belastungsgrenze respektiert, um vor kognitiver Erschöpfung geschützt zu werden.
- **Status:** Bestanden / Abgeschlossen. (Vorher: Verzögert (Ressourcen-Engpass))
- **Akzeptanzkriterien:**
    - Die Engine bricht die Tagesplanung hart ab und weicht auf den Folgetag aus, sobald das tägliche kognitive Limit (2-4 Stunden) erreicht ist.
    - Auf kognitiv schwere Aufgaben folgen bei der Einplanung zwingend leichtere Aufgaben (Prävention von mentalem Task-Switching).
    - Freie Zeitslots, die kleiner als die minimale Chunk-Größe einer Aufgabe sind, werden zur Vermeidung von Fragmentierung konsequent ignoriert.
    - **Früher Belastungstest:** Ein automatisierter Unit-Test im Backend beweist, dass die Kernschleife mit 50 Tasks bereits jetzt deutlich unter der 20-

Sekunden-Marke bleibt.

- **Zugehörige operative Zuweisung:**
    - **Team B** (Backend-Engine): Implementierung der Kernschleife und des harten kognitiven Guards (Abbruch bei 2-4 Stunden).
    - **Team A** (Frontend & Ressourcen-Shift): Visuelle Darstellung des generierten Plans im UI; nach zeitnahem Abschluss unmittelbarer Wechsel ins Backend zur Unterstützung bei der Anti-Fragmentierungs-Logik (Chunking).
- **Erforderliche Testarten & Testübersicht:**
    - *Automatisierte Unit-Tests (Backend):* Rigoroser Test der Kernschleife: Die Engine muss die Tagesplanung hart abbrechen und auf den Folgetag ausweichen, wenn das individuelle Limit von maximal 2 bis 4 Stunden erreicht wird. Test auf Anti-Fragmentierung: Freie Zeitslots, die kleiner als die minimale Chunk-Größe sind, müssen vom Algorithmus strikt ignoriert werden.
    - *Manuelle Systemtests:* Validierung der erzeugten Arbeitspläne im Frontend auf kompromisslose Einhaltung des kognitiven Limits und des Energiebedarfs.
- **Definition of Done (DoD):** Die Akzeptanzkriterien sind vollständig erfüllt. Komponententests (Backend/Frontend) existieren gemäß Testkonzept, sind in die Testsuite integriert und passieren die GitLab CI/CD-Pipeline fehlerfrei. Die manuellen Systemtests wurden dokumentiert und bestanden. Der Code ist in das Release-Inkrement integriert.
- **Bilanz & Retrospektive (Ende Wochenscheibe 2):** Wochenscheibe 2 wurde erfolgreich abgeschlossen. Es hat soweit alles gepasst: Wir konnten die Rückstände aus der letzten Woche vollständig aufholen und haben auch die Stories von dieser Woche erfolgreich umgesetzt. Das Projekt steht damit wieder voll im Zeitplan und die formale Definition of Done (DoD) wurde erreicht. Lediglich bei der Implementierung des Algorithmus sind noch ein paar kleinere Fragen aufgekommen, die wir im weiteren Verlauf klären.

Wochenscheibe 3: Dynamik, Rescheduling & Sicherheit

User Story 5: Echtzeit-Rescheduling & Automatischer Task-Abschluss

- **User Story:** Als Nutzender möchte ich, dass sich mein Arbeitsplan bei Abweichungen asynchron anpasst, damit er stets die aktuelle Realität widerspiegelt.

- **Status:** Bestanden / Abgeschlossen
- **Akzeptanzkriterien:**
    - Das Hinzufügen neuer Blocker triggert serverseitig ein sofortiges Rescheduling aller kollidierenden Aufgaben.
    - Das manuelle Abhaken von Aufgaben im UI löst eine partielle Neuberechnung für den verbleibenden Restplan aus.
    - Aufgaben, deren eingeplante Zeit vollständig abgelaufen ist, werden automatisch auf den Status "Abgeschlossen" gesetzt.
- **Zugehörige operative Zuweisung:**
    - **Team A** (Ressourcen-Shift → Backend-Assistenz): Vollständige Fokussierung auf die Backend-Zuarbeit für das Echtzeit-Rescheduling.
    - **Team B** (Backend-Logik): Event-Trigger für sofortiges Rescheduling bei Blocker-Erstellung und Logik für automatischen Task-Abschluss.
- **Erforderliche Testarten & Testübersicht:**
    - *Integrationstests:* Die Erstellung eines Blockers muss das Rescheduling-Event für alle zeitlich kollidierenden Aufgaben auf Backend-Ebene sofort triggern.
    - *Automatisierte Unit-Tests:* Prüfung der Logik, die abgelaufene Aufgaben automatisch als "Abgeschlossen" markiert.
- **Definition of Done (DoD):** Die Akzeptanzkriterien sind vollständig erfüllt. Komponententests (Backend/Frontend) existieren gemäß Testkonzept, sind in die Testsuite integriert und passieren die GitLab CI/CD-Pipeline fehlerfrei. Die manuellen Systemtests wurden dokumentiert und bestanden. Der Code ist in das Release-Inkrement integriert.

User Story 6: System-Authentifizierung & Last Admin Protection

- **User Story:** Als Plattform-Betreiber möchte ich sichere Mandantenstrukturen gewährleisten, damit Nutzende isoliert und sicher innerhalb ihrer Organisationen agieren können.
- **Status:** Bestanden / Abgeschlossen
- **Akzeptanzkriterien:**
    - Alle geschützten Backend-Routen validieren zwingend JWT-Tokens via Firebase Auth ([Authorize] Middleware).

- Das System blockiert das Löschen oder Herabstufen des letzten aktiven Administrators einer Organisation serverseitig.
- Das System prüft bei der Erstanmeldung von Mitarbeitern, ob deren E-Mail-Domain auf der Whitelist der Organisation steht.
- Administratoren können für Freelancer spezifische Einladungs-Links generieren, die per E-Mail versendet werden.

- **Zugehörige operative Zuweisung:**
  - **Team A** (Frontend-Integration): UI-Anbindung für Login, Registrierung und die Darstellung der rollenbasierten Navigation.
  - **Team B** (Backend-Logik): Integration der Firebase-JWT-Middleware, Absicherung der Endpunkte und Implementierung der "Last Admin Protection".

- **Erforderliche Testarten & Testübersicht:**
  - *Automatisierte Unit-Tests:* Prüfung der JWT-Token-Validierung (Firebase Auth) für alle geschützten Backend-Routen.
  - *Integrationstests:* Absicherung der "Last Admin Protection". Das System muss den Versuch, das letzte aktive Administratorkonto zu löschen, mit einem HTTP-Fehler hart blockieren.

- **Definition of Done (DoD):** Die Akzeptanzkriterien sind vollständig erfüllt. Komponententests (Backend/Frontend) existieren gemäß Testkonzept, sind in die Testsuite integriert und passieren die GitLab CI/CD-Pipeline fehlerfrei. Die manuellen Systemtests wurden dokumentiert und bestanden. Der Code ist in das Release-Inkrement integriert.

- **Bilanz & Retrospektive (Ende Wochenscheibe 3):** Wochenscheibe 3 wurde erfolgreich abgeschlossen. Wir haben den Code basierend auf den kleineren Fragen zur Algorithmus-Implementierung aus der letzten Woche nach der Besprechung mit dem Kunden angepasst und die User Stories für diese Woche pünktlich und erfolgreich implementiert und abgeschlossen. Sowohl der Algorithmus als auch das Registrieren und Einladen von Nutzern sind nun voll funktionsfähig und integriert. Das Projekt steht damit weiterhin voll im Zeitplan und die formale Definition of Done (DoD) wurde erreicht.

Wochenscheibe 4: Konfliktlösung & Manuelle Eingriffe

User Story 7: Manuelle Planänderung (Task-Verschiebung)

- **User Story:** Als Nutzender möchte ich eingeplante Tasks manuell umplanen können, um flexibel auf spontane Änderungen reagieren zu können.
- **Status:** Geplant
- **Akzeptanzkriterien:**
  - Tasks können über das UI (z. B. via Editier-Menü oder Datumsänderung) manuell auf einen anderen Zeitslot verschoben werden.
  - Das Backend registriert den manuellen Eingriff und wandelt den dynamischen Task in einen fixierten Bereich (isStatic = true) um.
  - Die manuelle Planänderung löst zwingend ein sofortiges, systemweites Rescheduling der verbleibenden Aufgaben aus, bei dem verifiziert wird, dass die tägliche kognitive Kapazitätsgrenze (max. 2 bis 4 Stunden) auch nach der händischen Verschiebung eingehalten wird.
- **Zugehörige operative Zuweisung:**
  - **Team A** (Frontend-Integration): Umsetzung der UI-Komponenten zur manuellen Änderung der Task-Zeiten.
  - **Team B** (Backend-Logik): Aktualisierung des Zustandsautomaten (Umwandlung in isStatic = true) und Neuvalidierung der Limits bei manuellen Eingriffen.
- **Erforderliche Testarten & Testübersicht:**
  - *Automatisierte Unit-Tests (Backend):* Verifikation der Zustandsänderung eines Tasks. Das System muss sicherstellen, dass ein manuell geänderter Task von der Engine als fixierter Blocker gewertet wird.
  - *Manuelle Systemtests:* Prüfung der manuellen Verschiebung im Kalender-UI, wobei das System die neuen Ziel-Zeitslots zwingend auf ausreichende zeitliche Kapazität und Limit-Einhaltung prüfen muss.
- **Definition of Done (DoD):** Die Akzeptanzkriterien sind vollständig erfüllt. Komponententests (Backend/Frontend) existieren gemäß Testkonzept, sind in die Testsuite integriert und passieren die GitLab CI/CD-Pipeline fehlerfrei. Die manuellen Systemtests wurden dokumentiert und bestanden. Der Code ist in das Release-Inkrement integriert.

User Story 8: Simples Clash Management

- **User Story:** Als Nutzender möchte ich bei unlösbaren Planungskonflikten gewarnt werden, damit ich unrealistische Deadlines oder Limits manuell anpassen kann.
- **Status:** Geplant
- **Akzeptanzkriterien:**

- Stellt der Algorithmus fest, dass Deadlines aufgrund der zeitlichen Limits nicht mehr einhaltbar sind, wird der Planungsprozess kontrolliert abgebrochen.

- Das Backend wirft eine Exception, die in der Controller-Schicht übersetzt und im Frontend als simple, verständliche Toast-Nachricht ("Plan nicht erstellbar") ausgegeben wird.

- **Zugehörige operative Zuweisung:**

  - **Team B** (Backend-Logik): Exception-Handling bei algorithmischem Abbruch (Verletzung des Limits oder der Deadline).
  - **Team A** (Frontend-Integration): Abfangen des HTTP-Fehlers und Rendering der transparenten Warnmeldung ("Plan nicht erstellbar") im UI.

- **Erforderliche Testarten & Testübersicht:**

  - *Automatisierte Unit-Tests (Backend):* Exception-Handling testen. Stellt der Algorithmus fest, dass Hard-Deadlines aufgrund des strikten 2-4 Stunden Limits mathematisch nicht mehr einhaltbar sind, muss der Prozess mit einer Exception kontrolliert abbrechen.
  - *Manuelle Systemtests:* Sichtprüfung im Frontend, ob die Backend-Exception korrekt übersetzt und als Toast-Nachricht ("Plan nicht erstellbar") für den Nutzenden gerendert wird.
- **Definition of Done (DoD):** Die Akzeptanzkriterien sind vollständig erfüllt. Komponententests (Backend/Frontend) existieren gemäß Testkonzept, sind in die Testsuite integriert und passieren die GitLab CI/CD-Pipeline fehlerfrei. Die manuellen Systemtests wurden dokumentiert und bestanden. Der Code ist in das Release-Inkrement integriert.

Wochenscheibe 5: Systemstabilisierung & Endabnahme

(Hinweis: In dieser Wochenscheibe herrscht Code Freeze für neue funktionale User Stories. Der Fokus liegt kompromisslos auf den Nicht-Funktionalen Anforderungen (NFAs) und den vertraglichen Abgabekriterien zur Endabnahme.)

Zugehörige operative Zuweisung:

- **Team A** (Systemtest & Modellierung): Durchführung der automatisierten System- und Performance-Tests (Nachweis: Planungs-Engine generiert Plan für 50 Tasks strikt in < 20 Sekunden). Aktualisierung der dynamischen UML-Diagramme (Designbeschreibung) und finale Pflege des Glossars.
- **Team B** (Systemtest & Deployment): Finale Prüfung der UI auf W3C-Standard und Browserkompatibilität (Chrome, Firefox, Edge, Safari) ohne externe Styling-Plugins. Erstellung der Anwenderdokumentation und des finalen Release-Archivs (sämtliche Projektunterlagen, ZIP inkl. README) für die Endabnahme.