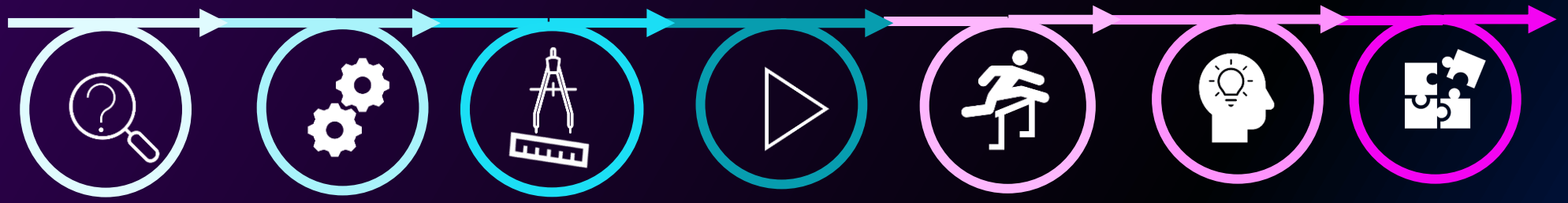
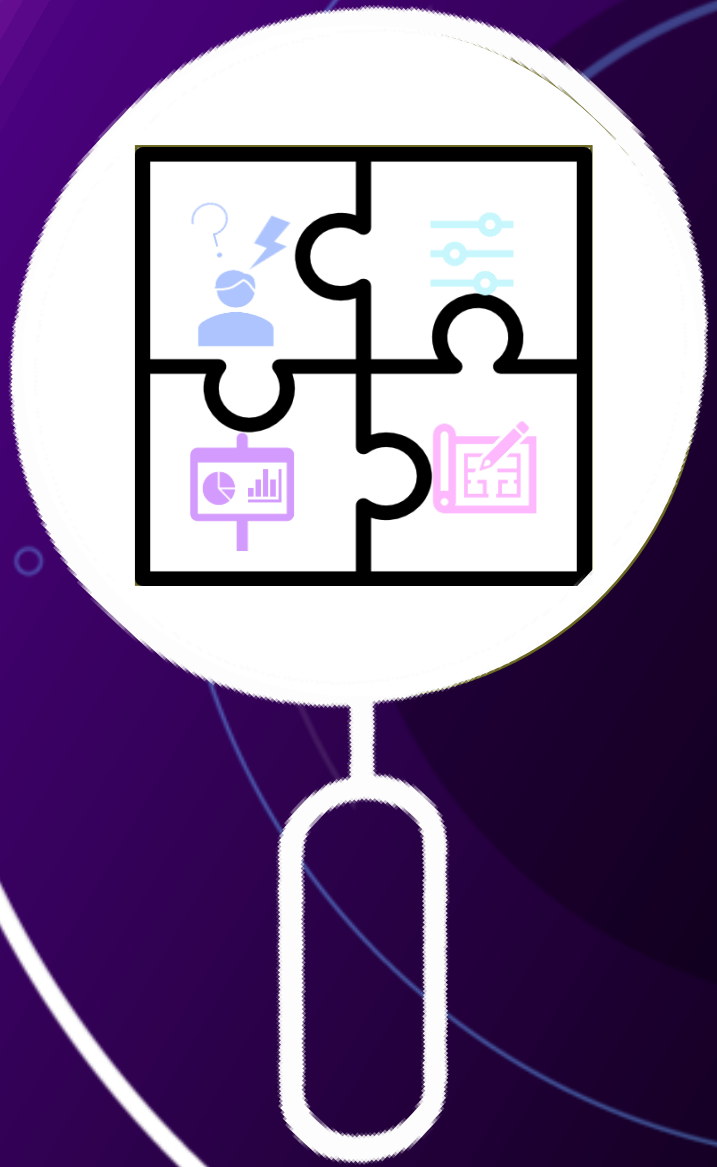
# MindKeeper

ERINNERUNGS-APP

# AGENDA

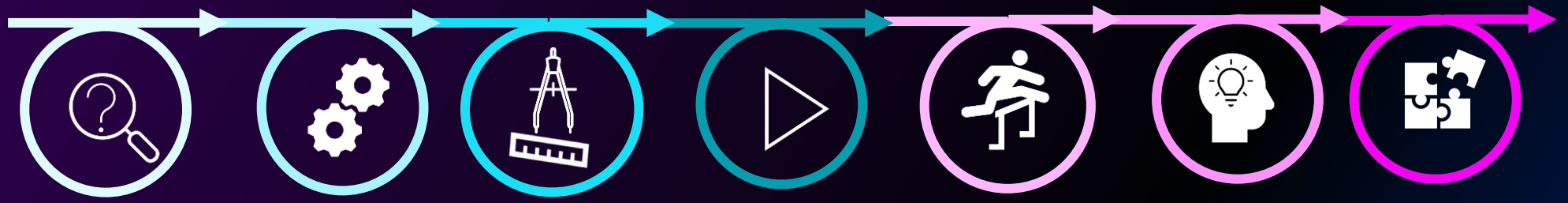


# PROBLEMSTELLUNG

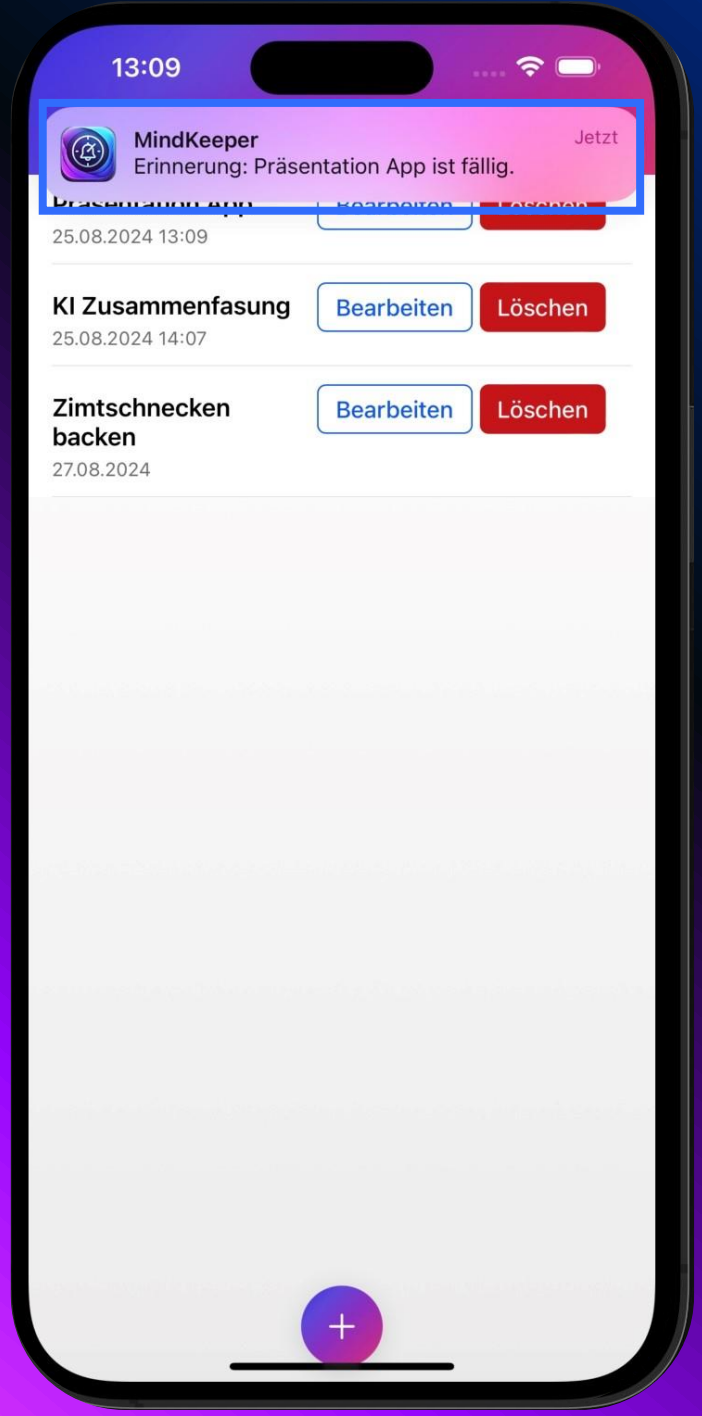


- Stress und Vergesslichkeit
- Mangelende Flexibilität bei unerwarteten Ereignissen
- Überlastung durch Informationen
- Schwierigkeiten bei langfristiger Planung

# AGENDA



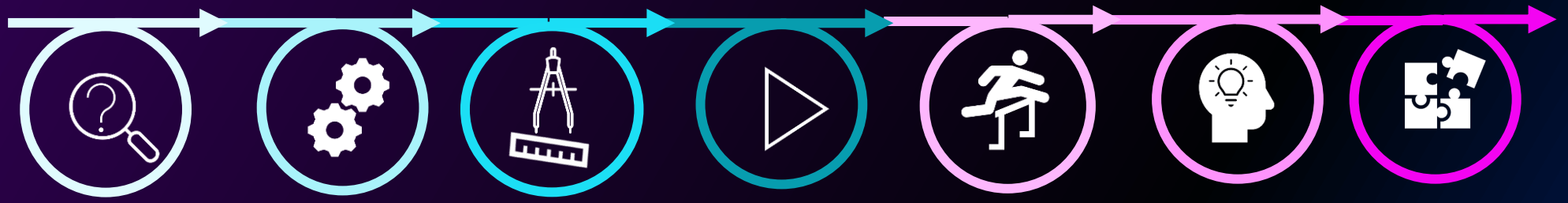
# IOS


# FUNKTIONEN

- Hinzufügen von Erinnerungen
- Bearbeiten von Erinnerungen
- Löschen von Erinnerungen
- Einplanen von Benachrichtigungen

# AGENDA



# EIGENSCHAFTEN 

Plugins:

- Local Notifications
- Preferences
- Datetime Picker vom Capawesome-Team

# Local Notification

capacitor.config.ts

```ts
import type { CapacitorConfig } from '@capacitor/cli';

const config: CapacitorConfig = {
  appId: 'io.ionic.starter',
  appName: 'MindKeeper',
  webDir: 'dist',
  plugins: {
    LocalNotifications: {
      smallIcon: "ic_stat_icon_config_sample",
      iconColor: "#488AFF",
      sound: "beep.wav",
    },
  },
};

export default config;
```

# Local Notification

Berechtigungen

requestPermissions();
checkPermissions();

```
onMounted(() => {
  requestNotificationPermissons();
  loadReminders();
});
```

```
const requestNotificationPermissons = async () => {
  try {
    const { display } = await LocalNotifications.checkPermissions();
    if (display !== 'granted') {
      await LocalNotifications.requestPermissions();
    }
  } catch (error) {
    console.error('Fehler beim Anfordern der Berechtigungen: ', error);
  }
}
```

# Local Notification

Benachrichtigung erstellen

schedule({});

```
await LocalNotifications.schedule({
  notifications: [
    {
      title: 'MindKeeper',
      body: `Erinnerung: ${reminder.text} ist fällig.`,
      id: reminder.id,
      schedule: { at: reminderDate },
      sound: 'beep.wav',
      attachments: undefined,
      actionTypeId: "",
      extra: null
    }
  ]
});
```

# Local Notification

Benachrichtigung löschen

cancel({});

```
const deleteReminder = async (id: number) => {
  await LocalNotifications.cancel({ notifications: [{ id }] });

  reminders.value = reminders.value.filter(reminder => reminder.id !== id);
  existingIds.value = reminders.value.map(reminder => reminder.id);
  await saveReminders();
};
```

# Preferences

Reminder erhalten

get({});

```
onMounted(() => {
  requestNotificationPermissons();
  loadReminders();
});
```

```
const loadReminders = async () => {
  try {
    const { value } = await Preferences.get({ key: 'reminders' });
    if (value) {
      reminders.value = JSON.parse(value);
      existingIds.value = reminders.value.map(reminder => reminder.id);
    }
  } catch (error) {
    console.error('Fehler beim Laden der Erinnerungen: ', error);
  }
};
```

# Preferences

Reminder speichern

set({});

```
const saveReminders = async () => {
  try {
    await Preferences.set({ key: 'reminders', value: JSON.stringify(reminders.value) });
  } catch (error) {
    console.error('Fehler bei der Speicherung der Erinnerungen: ', error);
  }
};
```

# Datetime Picker

## HTML-Aufruf

```
<!-- Datum -->
<ion-item lines="none" class="spaced-item" detail-none>
  <ion-label position="stacked" class="label-large">Datum</ion-label>
  <div class="datetime-container">
    <ion-button expand="block" @click="selectDate" color="light">Datum auswählen</ion-button>
    <ion-label class="datetime-label">{{ formattedDate ? formatDate(formattedDate) : 'Kein Datum ausgewählt' }}</ion-label>
  </div>
</ion-item>
```

# Datetime Picker

present({});

```
const selectDate = async () => {
  const date = new Date(reminder.value.date || Date.now());
  const { value } = await DatetimePicker.present({
    mode: 'date',
    value: date.toISOString(),
    doneButtonText: 'Fertig',
    cancelButtonText: 'Abbrechen',
    theme: 'dark',
    locale: 'de-DE',
  });
  if (value) {
    reminder.value.date = value.split('T')[0];
    formattedDate.value = reminder.value.date;
  }
};
```

# Datetime Picker

Implementierungsunterschiede für Datum- und Uhrzeit-Picker

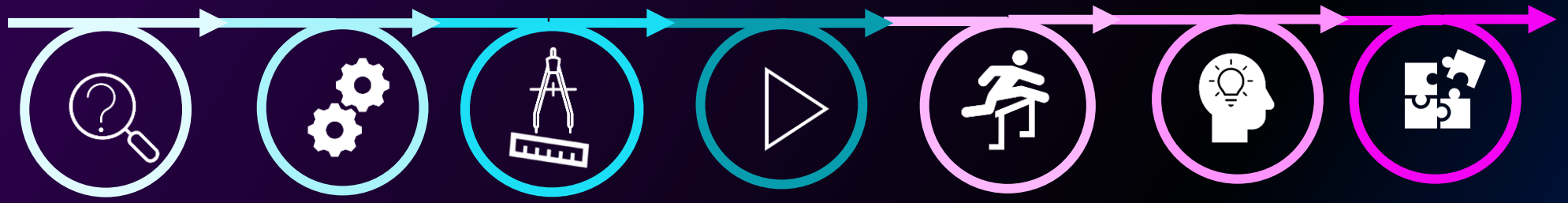
## Datum

```
const selectDate = async () => {
  const date = new Date(reminder.value.date || Date.now());
  const { value } = await DatetimePicker.present({
    mode: 'date',
    value: date.toISOString(),
    doneButtonText: 'Fertig',
    cancelButtonText: 'Abbrechen',
    theme: 'dark',
    locale: 'de-DE',
  });
  if (value) {
    reminder.value.date = value.split('T')[0];
    formattedDate.value = reminder.value.date;
  }
};
```

## Uhrzeit

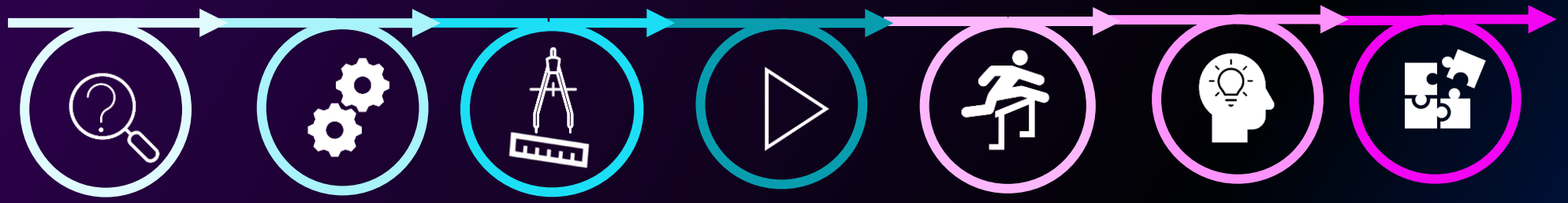
```
const selectTime = async () => {
  const time = new Date(`1970-01-01T${reminder.value.time || '00:00:00'}`);
  const { value } = await DatetimePicker.present({
    mode: 'time',
    value: time.toISOString(),
    doneButtonText: 'Fertig',
    cancelButtonText: 'Abbrechen',
    theme: 'dark',
    locale: 'de-DE',
  });
  if (value) {
    reminder.value.time = value.split('T')[1].substring(0, 5);
    formattedTime.value = reminder.value.time;
  }
};
```

# AGENDA



# LIVE DEMO

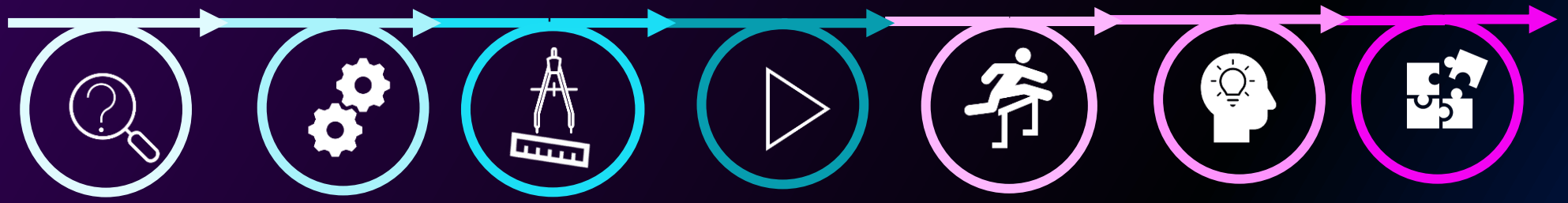
# AGENDA



# HERAUSFORDERUNGEN & GRENZEN

- iOS-spezifische Aufgaben: Entwicklerfreigabe für das Testen auf echtem Gerät
- Anpassung der Zeitangaben in der UI in deutsches Format
- Entwicklungsumgebung auf Mac einrichten

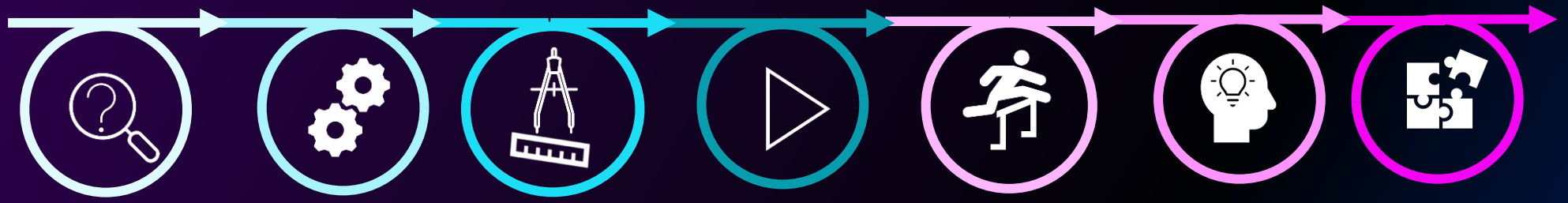
# AGENDA



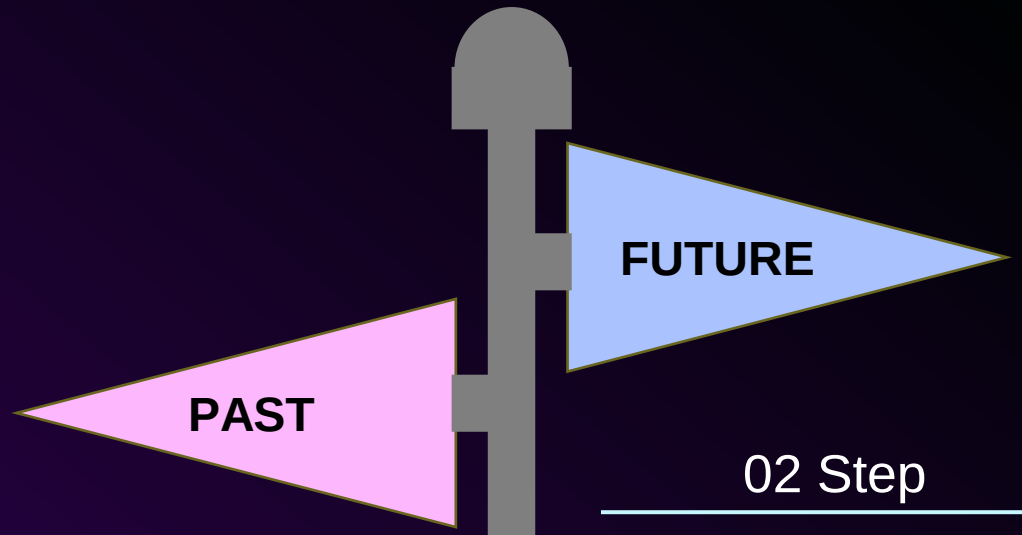
# ERFAHRUNGEN & ERKENNTNISSE

1. Kollaboratives Arbeiten in der App-Entwicklung
2. Ermittlung von Lösungen durch Recherche
3. Framework und Typescript
4. Entwickeln mit MacOS und X-Code

# AGENDA



# FAZIT & AUSBLICK




## 01 Step

- Einfach bedienbare App
- Speichert Erinnerungen welche auch bearbeitet werden.

## 02 Step

- Erinnerungen mit Freunden teilen.
- Erinnerungs-App auf andere Apps zugreifen kann (Vorlesungsplan)

# MindKeeper

## ERINNERUNGS-APP