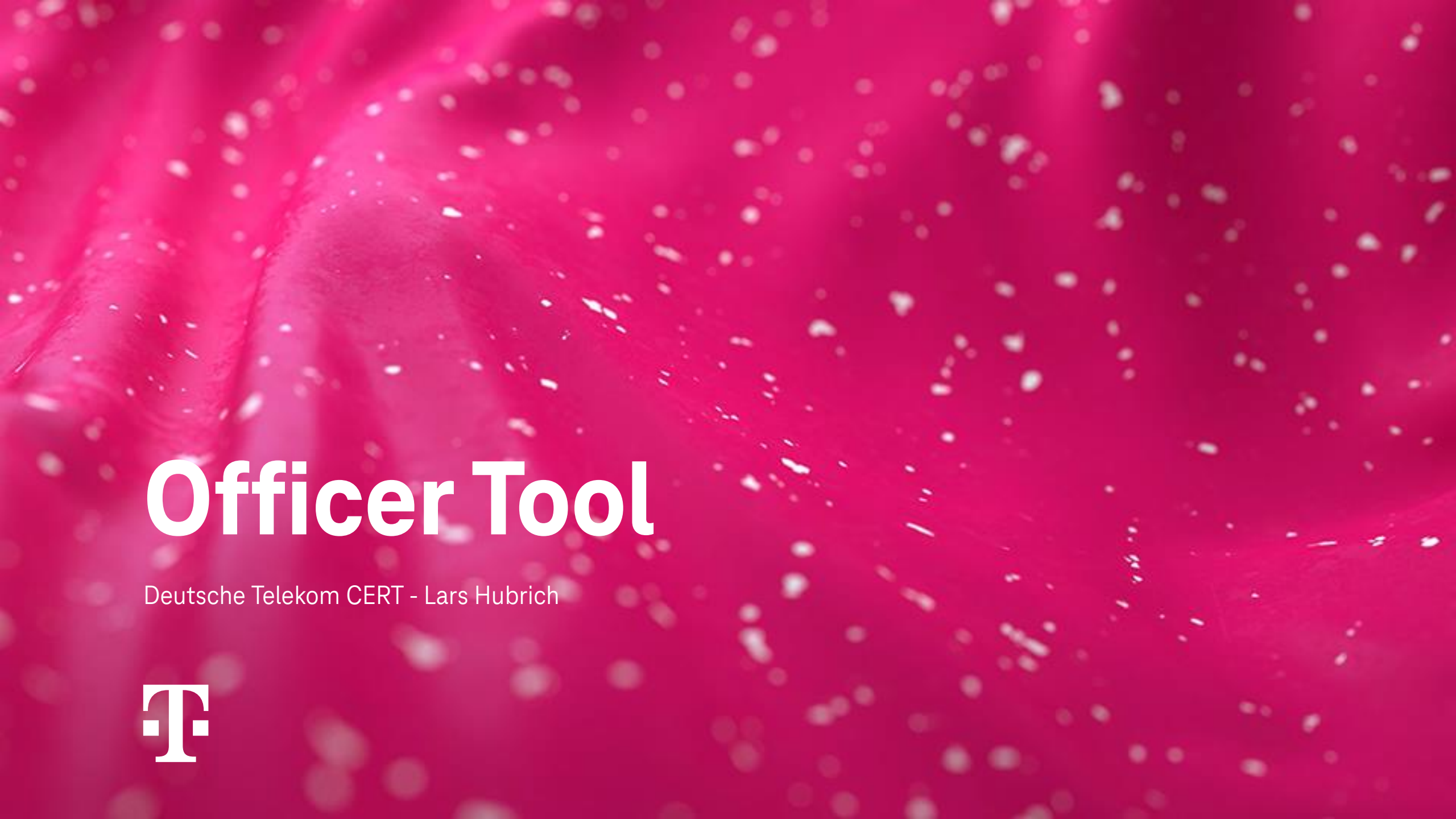


Officer Tool

Deutsche Telekom CERT - Lars Hubrich

Regelungen

Hard-Rules

- **Officer müssen in der „officer_static.json“ eingetragen sein**
  - Officer müssen den „Status“ „true“ in der „officer_static.json“ haben
- **Officer dürfen keinen Dienst an einem eingetragenen Urlaubstag haben**
- **Officer dürfen keinen Dienst an einem landesweiten oder bundeslandweiten Feiertag haben**
- **Officer dürfen nach einem Freitagsdienst kein Montagsdienst haben**
  - Wenn der Freitag ein Feiertag ist, wird der Donnerstagsdienst berücksichtigt
  - Wenn der Montag ein Feiertag ist, wird der Dienstagsdienst berücksichtigt
- **Officer dürfen niemals mehr als drei mal pro Woche Dienst haben**
- **Es muss immer mindestens eine Person mit deutschen Sprachkenntnissen Dienst haben**

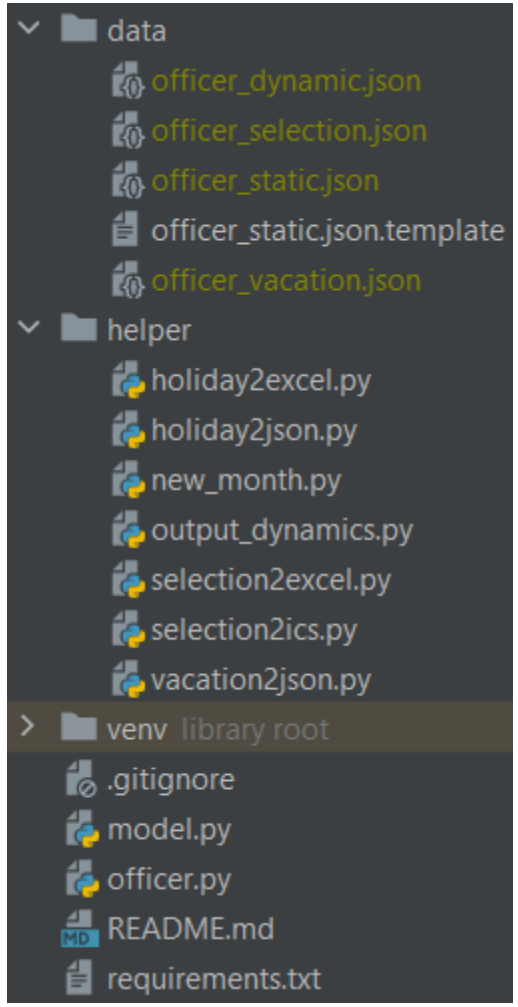
- **Hard-Rules können nicht gebrochen werden und können im schlimmsten Fall dazu führen, dass für einen Tag kein Officer / nicht genügend Officer ernannt werden**

Soft-Rules

1. **Ein Officer sollte so selten Dienst in einer Woche wie möglich haben**
2. **Ein Officer sollte so selten Dienst an einem extrem Tag wie möglich haben**
   - Extrem Tag = Montag & Freitag (oder Dienstag / Donnerstag bei Brückentagen)
3. **Ein Officer sollte so selten Dienst wie möglich haben**
4. **Nach Berücksichtigung der Regeln, sollte ein Dienst so zufällig wie möglich sein**

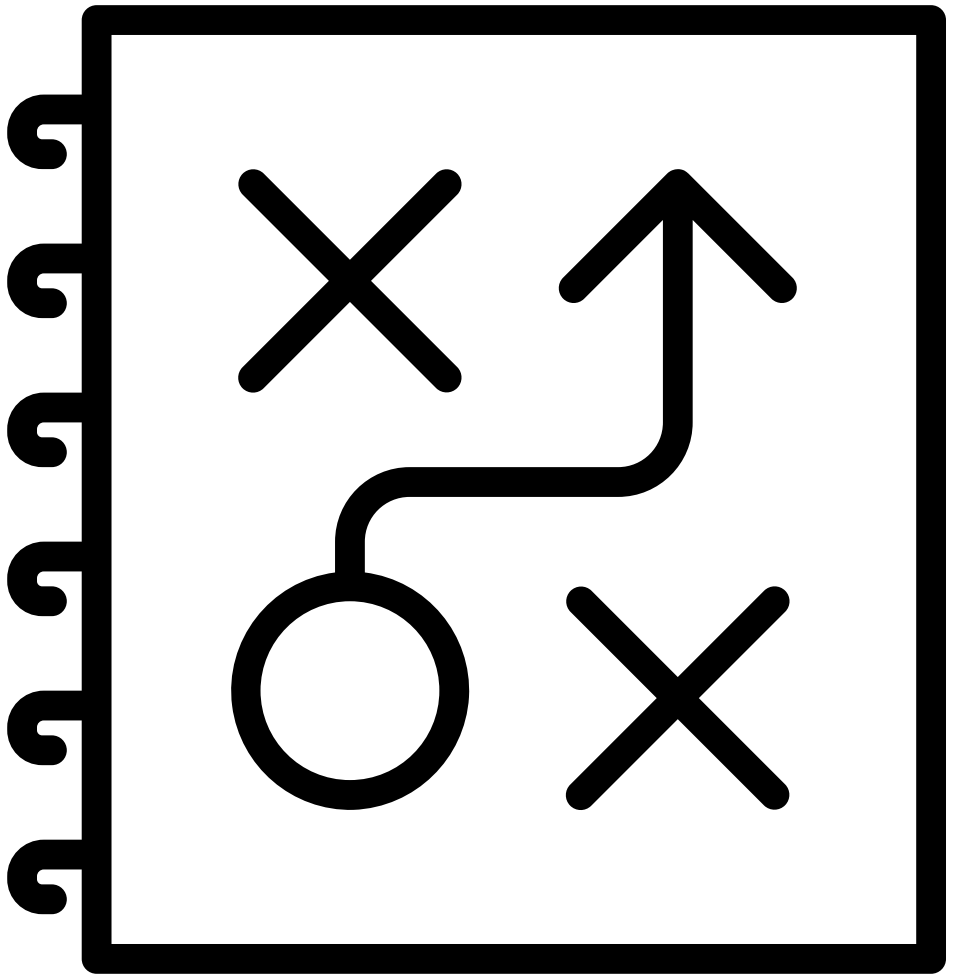
- **Nach unten hin, werden Regeln „unwichtiger“**

Verzeichnis- / Dateiübersicht



- Verzeichnis „data“
  - „officer_static.json“ muss manuell nach dem Schema in „ officer_static.json.template“ befüllt werden
  - Die restlichen Dateien entstehen während der Ausführung und müssen im Regelbetrieb nicht angefasst werden
- Verzeichnis „helper“
  - Enthält Skripte die Daten für das eigentliche Officer Skript vor- oder nachbereiten
- „officer.py“ ist das eigentliche generierende Skript

Ausführung
Step by Step



1. Monatsblätter generieren (excel_new_month.py)

- **Im gekennzeichneten Bereich muss folgendes eingetragen werden:**
  - Jahr für welches das Blatt generiert werden soll
  - Monat (Nummer mit 1 startend)
  - Monatsname
  - Dateiname (Pfad zur Officer Excel Datei)
- **Welche Voraussetzungen gibt es?**
  - Die Officer Static muss vorhanden und befüllt sein(„officer_static.json“)
  - Eine Officer Excel Datei muss existieren
  - Die Officer Excel Datei muss zumindest das TEMPLATE Blatt enthalten
- **Was macht das Skript?**
  - Es erzeugt ein neues Blatt nach TEMPLATE Schema mit Officernamen, Tagesanzahl, Tagesartanzeige, Legende und Wochenendmarkierungen

2. Feiertage generieren (holiday_lib2json.py)

- **Im gekennzeichneten Bereich muss folgendes eingetragen werden:**
  - Jahr für welches die Feiertage generiert werden sollen
- **Welche Voraussetzungen gibt es?**
  - Die Officer Static muss vorhanden und befüllt sein („officer_static.json“)
- **Was macht das Skript?**
  - Es lädt die Feiertage für die Länder und Bundesländer der Officer
  - Es exportiert diese nach „holidays.json“
  - Zusätzlich generiert es die Brückentage und exportiert diese nach „bridging_days.json“

3. Feiertage eintragen (holiday_json2excel.py)

- **Im gekennzeichneten Bereich muss folgendes eingetragen werden:**
  - Das Jahr
  - Dateiname (Pfad zur Officer Excel Datei)
- **Welche Voraussetzungen gibt es?**
  - Die Officer Static muss vorhanden und befüllt sein („officer_static.json“)
  - Die Feiertage müssen generiert sein („holidays.json“)
- **Was macht das Skript?**
  - Es trägt die Feiertage (mit der orangenen Farbe) in die Officer Tabelle ein

4. Urlaub auslesen (vacation_excel2json.py)

- **Im gekennzeichneten Bereich muss folgendes eingetragen werden:**
  - Das Jahr
  - Dateiname (Pfad zur Officer Excel Datei)
- **Welche Voraussetzungen gibt es?**
  - <keine>
- **Was macht das Skript?**
  - Es liest den Urlaub pro Officer anhand der (blauen) Farbe aus
  - Es liest weitere Abwesenheitstage (Workshop) anhand der grünen Farbe aus
  - Es speichert diese Daten nach „officer_vacation.json“

5. letzte Dienstverteilung auslesen (selection_excel2json.py)

- **Im gekennzeichneten Bereich muss folgendes eingetragen werden:**
  - Das Jahr
  - Dateiname (Pfad zur Officer Excel Datei)
- **Welche Voraussetzungen gibt es?**
  - in der Excel: Officer Dienste in der ausgewählten Zeit
- **Was macht das Skript?**
  - Es liest den Dienst pro Officer anhand der Farbe aus
  - Es speichert diese Daten nach „officer_selection.json“

6. Officer generieren (officer.py)

- **Im gekennzeichneten Bereich muss folgendes eingetragen werden:**
  - Das Startdatum (erster Tag mit Officer Dienst) (muss ein Montag sein)
  - Das Enddatum (letzter Tag mit Officer Dienst) (muss ein Freitag sein)
- **Welche Voraussetzungen gibt es?**
  - Die Officer Static muss vorhanden und befüllt sein („officer_static.json“)
  - Die Feiertage & Brückentage müssen generiert sein („holidays.json“ & „bridging_days.json“)
  - Der Urlaub kann verwendet werden („officer_vacation.json“)
  - Dynamische Officer Daten können verwendet werden („officer_dynamic.json“)
  - Die letzte Officer Auswahl kann verwendet werden („officer_selection.json“)
- **Was macht das Skript?**
  - Es generiert die Officer Dienste mit Berücksichtigung der früher genannten Regeln und exportiert diese nach „officer_selection.json“
  - Es exportiert die dynamischen Officer Daten nach „officer_dynamic.json“

7. In Excel eintragen (selection_json2excel.py)

- **Im gekennzeichneten Bereich muss folgendes eingetragen werden:**
  - Das Jahr
  - Dateiname (Pfad zur Officer Excel Datei)
- **Welche Voraussetzungen gibt es?**
  - Die Officer Auswahl muss generiert sein („officer_selection.json“)
- **Was macht das Skript?**
  - Es trägt die Officer Dienste mit einem „x“ in die Excel Tabelle ein

8. Officer Termin senden (selection_json2exchange.py)

- **Im gekennzeichneten Bereich muss folgendes eingetragen werden:**
  - Nutzername des Agenten / Servicenutzers
  - Passwort zu dem angegebenen Agenten
- **Welche Voraussetzungen gibt es?**
  - Die Officer Static muss vorhanden und befüllt sein („officer_static.json“)
  - Die Officer Auswahl muss generiert sein („officer_selection.json“)
- **Was macht das Skript?**
  - Es schickt den Officern eine Mail mit einer Termineinladung zu ihrem Dienst
  - Es speichert in dem FMB eine verknüpfte Kopie des Termins
    - Verknüpft -> Änderung im FMB am Termin ändert den Termin im eigenen Kalender (wie man es gewohnt ist)