

# Dokumentation Go-Ticketsystem

---

Von 2057008, 2624395, 9111696

## Architekturdokumentation

### Package main

Im Package main befinden sich die main.go des Webserver, sowie die mailIn.go und mailOut.go der REST-API. Weiterhin befinden sich dort die jeweiligen Test-Dateien.

Die main.go enthält die Handler für den Webserver, welche mit dem Package backend kommunizieren, sowie das Port-Flag und Anweisungen um fehlende Ordnerstrukturen nachzubauen.

In mailIn.go und mailOut.go befinden sich Kommandozeilentools, welche dann jeweils mit den Packages api\_in und api\_out kommunizieren.

### Package api\_in

In diesem Package werden Mails empfangen und bei bestehendem Ticket wird die Mail als Kommentar an das Ticket angeheftet. Wenn kein Ticket existiert, wird ein neues Ticket erstellt. Um zu prüfen, ob entsprechende Tickets vorhanden sind, gibt eine Schnittstelle zum Package backend.

### Package api\_out

Package api\_out hat die Funktionen welche mit dem Versand von Mails zu tun haben: anzeigen der Mail- Warteschlange, versenden der Warteschlange, hinzufügen und löschen von Mails in der Warteschlange sowie Ansteuerung von Seiten eines Mailserver über die angebotene API. Die main.go greift auf das Package zu um die Schnittstelle/API darzustellen, mailOut.go um die API zu steuern.

### Package authentication

Das Package Authentication hält alle Funktionen, welche mit der Benutzerverwaltung und Benutzerauthentifikation zu tun haben: Anlegen neuer Benutzer, Abgleichen mit bestehenden Benutzern, Ausgabe von Nutzerdaten, Passwortverschlüsselung, Abgleichen von Passwörtern und Entschlüsselung von Passwörtern. Weiterhin sind Funktionen und Handler der Basic Authentication außer das Logout, welches im Frontend abgewickelt wird, in diesem Package enthalten. Jede Klasse, welche Daten über Benutzer oder die Authentifizierung von Benutzern abfragt, greift auf das Package zu.

### Package backend

Im Package backend befindet sich die Klasse ticketsBackend.go, in der sich der Großteil der Handler befindet, die von der Main aufgerufen werden und auch die Funktionen, um die Tickets zu speichern oder zu aktualisieren. Dies umfasst Tickets erstellen, Kommentare hinzufügen, Tickets freigeben, übernehmen, zuweisen und zusammenführen und Tickets schließen. Der

Urlaubsstatus des eingeloggten Users wird aus profile.html ausgelesen und in users.json gespeichert. Jede Klasse die die Map mit Tickets braucht, greift auf

### Package frontend

Das Package frontend enthält alle HTML- und CSS-Dateien, die für das Frontend benötigt werden. Zusätzlich enthält es noch das Skript logout.js, das beim Auswählen von Logout ausgeführt wird.

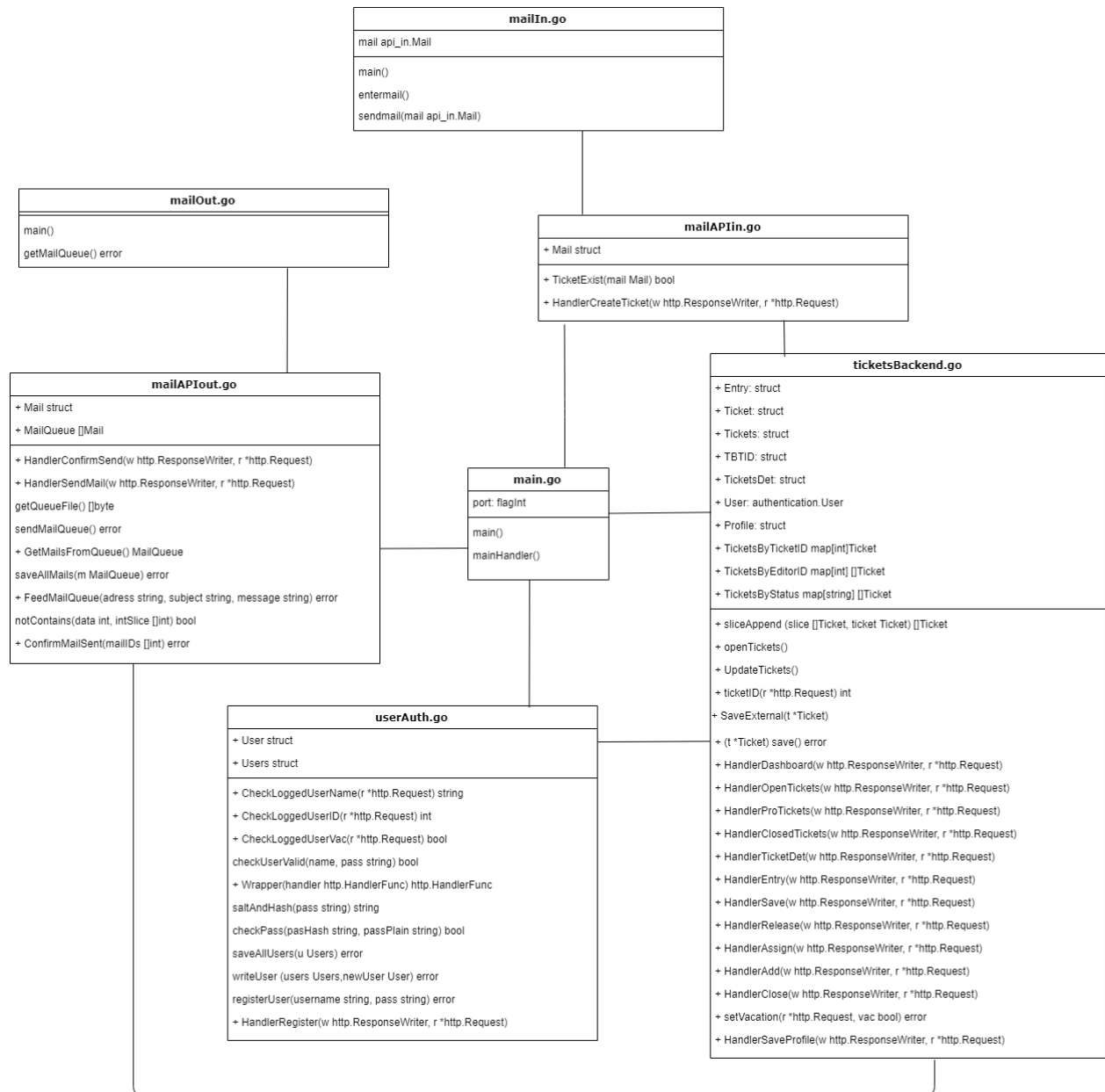
### Datenhaltung

Die Daten werden in den Ordnern "tickets", "users" und "mailQueue" abgelegt.

Tickets werden in jeweils einer eigenen JSON-Datei abgelegt, welche "ticket"+die zugehörige TicketID genannt werden. In dieser JSON-Datei befinden sich dann die Attribute des Tickets und die Liste an Einträgen.

In "users" und "mailQueue" befindet sich jeweils nur eine JSON-Datei, die alle registrierten User beziehungsweise alle Mails, welche sich in der Queue befinden, enthält.

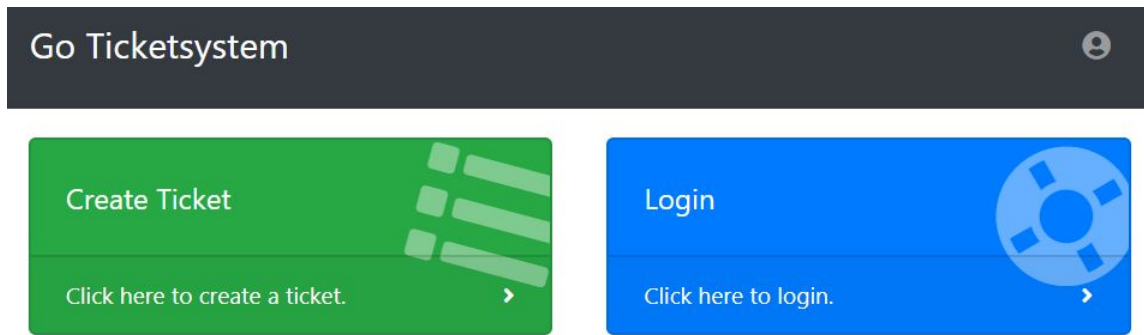
Serverzertifikat und Serverkey liegen auf Höhe von den Ordnern "cmd" und "pkg".



## Anwenderdokumentation

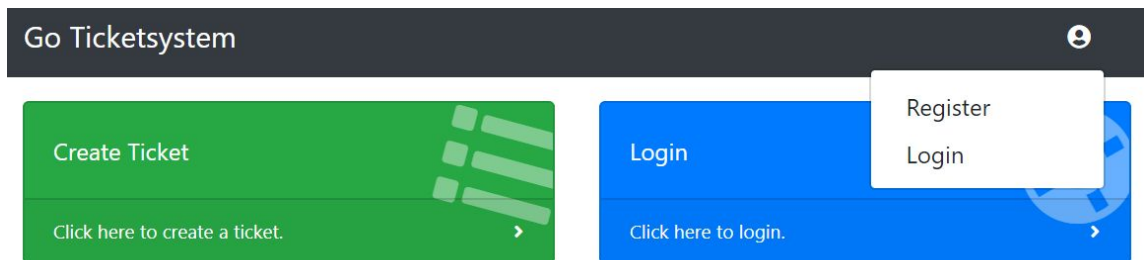
- Startseite

Auf der Startseite befinden sich zwei Schaltflächen. Die Linke kann jeder Nutzer benutzen, um ein neues Ticket zu erstellen. Die Rechte Schaltfläche ist für Nutzer gedacht, die als Bearbeiter fungieren und über Zugangsdaten verfügen. Der Login kann auch über das Benutzericon in der oberen, rechten Ecke angestoßen werden. Sollten noch keine Zugangsdaten vorhanden sein, kann hier auch das Registrierungsformular geöffnet werden.



- **Registrierung**


Von der Startseite aus kann der Benutzer oben rechts auf die Registrierung zugreifen. Als User ist standardmäßig nur ein User hinterlegt (Username: admin, Passwort: supersecret). Mit diesen Zugangsdaten kann auch ohne Registrierung auf das Ticketsystem zugegriffen werden.




Es öffnet sich ein neues Fenster indem der User seinen Namen und sein Passwort eingibt.

Register an Account


Name



Password



Confirm password



Register

Nach Bestätigung des Buttons "Register" kann sich der User mit den eingegebenen Zugangsdaten auf der Startseite anmelden.

- Ticketerstellung

Für die Ticketerstellung muss der Nutzer seine E-Mail-Adresse, den Betreff und die Beschreibung seines Problems eintragen. Anschließend kann auf speichern (Ticket wird erstellt und für alle Mitarbeiter angezeigt) oder auf abbruch (alle Eingaben werden gelöscht) klicken. Unabhängig davon gelangt der Nutzer wieder auf die Startseite.



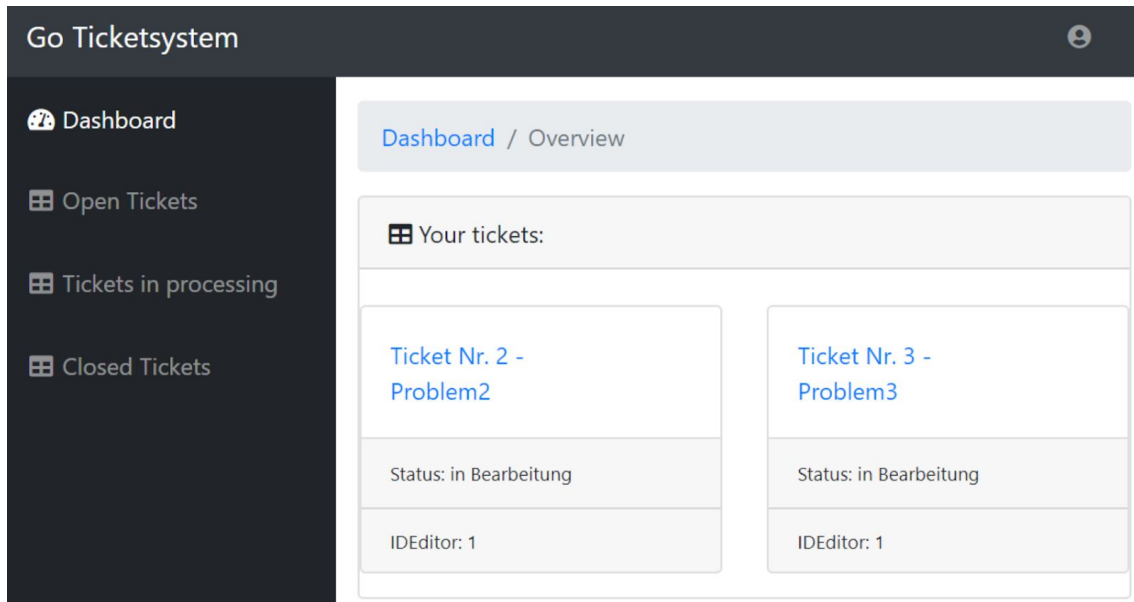
## Ticket erstellen

Text

Speichern

Abbruch

- **Dashboard (für Bearbeiter)**  
Nach erfolgreichem Login gelangt der Nutzer auf sein persönliches Dashboard. Hier werden alle Tickets mit Status "in Bearbeitung" angezeigt, die seiner ID zugeordnet sind. Für die Detailansicht eines Tickets muss der Bearbeiter lediglich auf den Betreff des Tickets klicken.



- Weitere Auflistungen der Tickets (für Bearbeiter)  
Im linken Menü kann ein Bearbeiter zwischen verschiedenen Ansichten wechseln. Zur Auswahl steht eine Ansicht mit den offenen Tickets, eine Ansicht mit den Tickets, die bei dem eingeloggten Benutzer in Bearbeitung sind und eine Ansicht mit den geschlossenen Tickets.
- Detailansicht eines Tickets (für Bearbeiter)  
In der Detailansicht eines Ticket kann ein Bearbeiter den gesamten bisherigen Verlauf des Tickets einsehen. Dies beinhaltet alle Kommentare von Ticketersteller und Bearbeitern mit Datum. Hier ist es einem Bearbeiter auch möglich ein offenes Ticket zu übernehmen oder ein Ticket, das ihm bereits zugeordnet ist freizugeben oder einem anderen Bearbeiter zuzuweisen. Es stehen für die Zuweisung nur die Bearbeiter zur Verfügung, die nicht den Status "im Urlaub" haben. Weiterhin können 2 Tickets zusammengeführt werden. Hierfür ist jedoch die Bedingung dass beide Tickets denselben Bearbeiter zugeordnet sind. Ist dies der Fall werden die Kommentare des ausgewählten Tickets an das aktuelle angehängt und das ausgewählte Ticket wird mit einem Systemkommentar auf den Status "geschlossen" gesetzt.





Dashboard

Open Tickets

Tickets in processing

Closed Tickets

## Ticket Nr. 3 - Problem3

in Bearbeitung

Bearbeiter: 1

2018-12-30 - hans@wurst.com:  
Blablabla1.1

2018-12-30 - John Doe2:  
Blablabla1.2

2018-12-30 - hans@wurst.com:  
Blablabla1.3

2019-01-01 - asd:  
asdasd

2019-01-05 - System:  
Das Ticket wurde wegen Zusammenführung geschlossen. Die Einträge wurden in Ticket Nr. 2 übertragen.

[Kommentar hinzufügen](#)[Ticket freigeben](#)

### Ticket anderem Bearbeiter zuweisen:

[Ticket zuweisen](#)[Ticket schließen](#)

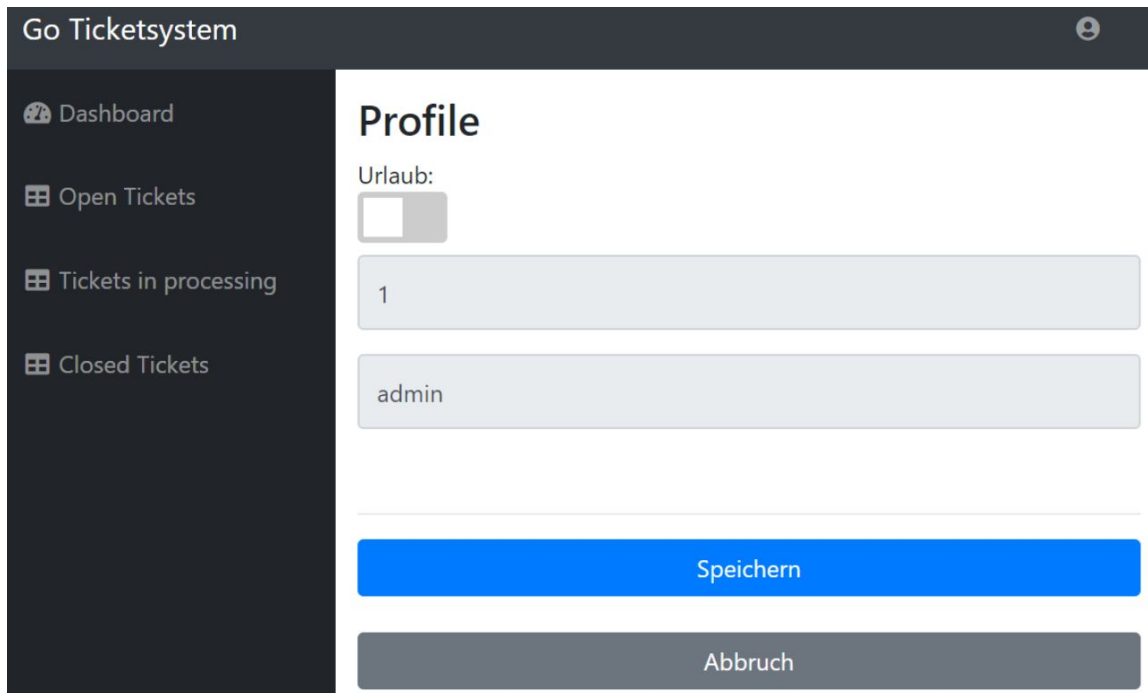
### Tickets zusammenführen:

Das aktuelle Ticket bleibt erhalten, während die Kommentare des ausgewählten Tickets angehängt und das Ticket geschlossen wird. Die Aktion wird nur ausgeführt, wenn beide Tickets denselben Bearbeiter haben.

[Ticket zusammenführen](#)

- Profil (für Bearbeiter)

Auf allen Seiten nach dem Login steht dem Nutzer oben rechts das Benutzericon zur Ansicht seines Profils zu Verfügung. Auf der Profilsseite wird der aktuelle Urlaubsstatus, die ID und der Name angezeigt. ID und Name können nicht geändert werden, der Urlaubsstatus kann jedoch je nach Bedarf gesetzt werden. Die Änderung des Status muss mit Klick auf Speichern bestätigt werden. Sobald ein Bearbeiter seinen Status auf Urlaub gesetzt hat, können ihm keine neuen Tickets zugewiesen werden.



Go Ticketsystem

Dashboard

Open Tickets

Tickets in processing

Closed Tickets

## Profile

Urlaub:

☐

1

admin

Speichern

Abbruch

## Dokumentation des Betriebs

### Inbetriebnahme

Für eine lauffähige Anwendung muss die zip-Datei entpackt werden und Golang auf dem Rechner installiert sein. Die verwendete Go-Version ist 1.11.4 für dieses Projekt.

Um die Anwendung auszuführen werden folgende Packages benötigt: Bcrypt, Assert und Errors. Diese können mit `go get "github.com/pkg/errors"`, `go get "github.com/stretchr/testify/assert"` und `go get "golang.org/x/crypto/bcrypt"` geladen werden. Es ist anfangs nur ein User für das Ticketsystem hinterlegt (Username: admin, Passwort: supersecret).

### Starten mit Flags

Es steht zur Konfiguration des Ports das Flag "port" zur Verfügung. Mit "port" wird der Port auf dem der Server gestartet wird, gesetzt. Der Default ist 443.

## REST-API für Mail-Empfang

Um die API mit Hilfe eines Kommandozeilentools zu nutzen steht im Ordner cmd die Datei mailIn.go zur Verfügung. Wird diese gestartet muss der Nutzer zunächst mit "y" bestätigen, dass er eine Nachricht senden will, danach wird die Absenderadresse, der Betreff und der Text der Nachricht abgefragt. Jede Eingabe muss der Nutzer mit Enter bestätigen.

```
Do you want to send a mail to Ticketsystem? (y/n)
y
Please enter your e-mail-address:
test@test.com
Please enter the subject of email:
Test API
Please enter the text of email:
This is a test.
Message successfully created.
```

Zum Schluss erhält der Nutzer noch eine Erfolgsmeldung, dass die Nachricht erfolgreich erstellt und versendet wurde.

## REST-API für Mail-Versand

Der Mailversand über die REST-API kann über ein Kommandozeilentool gesteuert werden. Dieses befindet sich in der Klasse mailOut.go. Nach dem Start stehen dem Nutzer drei Befehle zur Verfügung:

'show' - Zeigt alle Mails an, die sich in der Warteschlange befinden.

'send' - Simuliert das versenden der Mails in der Warteschlange durch den Mailserver. Die IDs der gesendeten Mails werden angegeben (mit Kommata getrennt, ohne Leerzeichen) und anschließend aus der Warteschlange gelöscht.

'exit' - Verlässt die Anwendung

## Beiträge der Gruppenmitglieder

2057008

- Authentifizierung (userAuth.go, Logout.js, Eingriffe in relevante andere Klassen [handler.go, main.go])  
Implementierung der Authentifizierung über das Basic Authentication-Verfahren. Bau des Wrapper-Handler Konstrukts, über das die gesicherten Teile der Website Zugangsgeschützt sind. Erstellung der Logout-Funktion im Rahmen des Basic Authentication-Verfahrens für die Browser Internet Explorer, Edge, Chrome und Firefox. Einbindung des "saltings", "hashings", dem Entschlüsseln und dem Abgleich der Passwörter.
- Benutzerverwaltung (userAuth.go)  
Anmeldung und Speicherung registrierter Nutzer in einer Datei, Passwörter sind nicht im Klartext. Funktionen, die anderen Teilen der Anwendung ausgeben können, welcher User gerade angemeldet ist, wurden ebenfalls eingebaut.
- Mail-Out API (mailAPIOut.go, mailOut.go)  
Bietet eine REST-API für einen Mailserver um zu sendende Mails abzufragen und eine Versandbestätigung anzunehmen. Konsumiert eine REST-API auf der Seite des Mailservers, um die Mails an ihn weiterzugeben. Außerdem wurde noch ein Kommandozeilentool eingerichtet, mit dem die Mail-Warteschlange angezeigt werden kann und das versenden der Mails simuliert werden kann.

2624395

- Flag (main.go)  
Flag zum Festlegen des Ports
- Directory (main.go)  
Erstellung der benötigten Folder "tickets", "users" und "mailQueue", falls diese nicht existent sind
- Backend (ticketsBackend.go)  
Umwandlung des Cachings von globaler Variable [ ]Tickets zu maps

9111696

- Frontend  
Erstellung der HTML-Seiten unter Zuhilfenahme von Bootstrap CSS und Einfügen von dynamischen Funktionen, um je nach Ticketstatus andere Buttons anzuzeigen.
- Backend
  - Ticketerstellung und Bearbeitung (ticketsBackend.go)

Bei gespeicherter Ticketerstellung wird ein Ticket im Ordner Tickets hinzugefügt und ist dann für alle Bearbeiter bei den Tickets mit Status "offen" zu sehen. Funktionalitäten zum Kommentar hinzufügen, Tickets freigeben und übernehmen, Tickets anderen Bearbeitern zuweisen und Tickets zusammenführen.

- Urlaubsmodus (ticketsBackend.go)  
Speichern des Zustandes des On-Off-Sliders in der User-Datei.
- Mail-In API (mailAPIin.go, mailIn.go)  
Kommandozeilentool, dass die REST-API, die empfangene Mails überprüft, aufruft. Entweder Erstellung eines Kommentars bei vorhandenem Ticket oder Erstellung eines neuen Tickets.