Go Ticketsystem Dokumentation

Duale Hochschule Baden-Württemberg Mosbach
Vorlesung Programmieren II
Herr Prof. Dr. Helmut Neemann

09.01.2019

Matrikelnummern der Studierenden: 3040018, 6694964, 3478222

Inhaltsverzeichnis

Ersteinrichtung und Konfiguration

Der Server und das CLI lassen sich einfach mit den Batch / Shell skripten starten, die man im trivial-tickets verzeichnis finden kann.
Für das CLI kann man auch stattdessen im trivial-tickets Verzeichnis Folgendes ausführen, um die command line flags leichter setzen zu können:

```
go get -t -v ./...
go run cmd\commandLineTool\commandLineTool.go
```

Konfiguriert werden beide Programme über command line flags:
Server:

- port: Port auf dem der Server laufen wird. default ist 8443
- tickets: Pfad in dem der Server die Ticket files abspeichert / nach den Ticket files sucht. default ist "../../files/tickets"
- users: Pfad zu der JSON datei, in der die Nutzer abgespeichert werden. default ist "../../files/users/users.json"
- mails: Pfad zum Ordner, in dem die Mail files abgespeichert werden. Default ist "../../files/mails"
- cert: Pfad zum SSL Zertifikat. Default ist "../../ssl/server.cert"
- key: Pfad zur SSL Key file. Default ist "../../ssl/server.key"
- web: Pfad zum www ordner. Default ist "../../www"j

Hinweis zu den Pfaden: alle default Pfadangaben sind relativ zum Ordner `ticketsystem` (`$GOPATH/src/trivial-tickets/cmd/ticketsystem`). Es können auch absolute Pfadangaben gesetzt werden.

Command-Line-Interface (CLI)

- IPAddr: IP Adresse des Servers (der Web API). Default ist "localhost"
- port: der Port des Servers auf dem die Web API läuft. Default ist 8443
- cert: Pfad zum Zertifikat der RootCA, die das Zertifikat des Servers unterzeichnet hat. Kann natürlich auch der Pfad zum Server Zertifikat selbst sein. Default ist "./ssl/server.cert". Der Pfad ist relativ zum trivial-tickets Ordner (%GOPATH/src/github.com/mortenterhart/trivial-tickets/)

Die anderen command line flags dienen der Bedienung, nicht der Konfiguration. Sie sind im Abschnitt Bedienung>CLI beschrieben.

Es sind von vorneherein 3 Benutzer
Zugangsdaten für die Benutzeraccounts:
Name: admin Passwort: blabla
Name: max4711 Passwort: blabla
Name: tron Passwort: blabla

Bedienung

Das Ticketsystem kann über das Command Line Interface (CLI) (auch als das Kommandozeilen-Tool bekannt) oder über die Weboberfläche gesteuert werden.
Über die Weboberfläche können Tickets erstellt, betrachtet und bearbeitet werden. Dabei können die nicht angemeldeten Nutzer nur Kommentare zu den Tickets hinzufügen. Administratoren/Mitarbeiter können nachdem sie sich angemeldet haben zudem den Status eines Tickets ändern, zwei Tickets zusammenführen und den Bearbeiter des Tickets ändern.
Das CLI kommuniziert über eine web-API mit dem server. Es simuliert dabei (aus der sicht des Ticketservers) einen Mail application, die Nachrichten vom server über die API abfragt und an die jeweiligen Emailadressen versendet. Zudem können über das CLI auch neue Nachrichten an den Ticketserver eingegeben werden, die dann serverseitig verwendet werden um entweder ein neues Ticket zu erstellen, oder einen Kommentar an ein bereits bestehendes Ticket anzuhängen.

Weboberfläche

- Erstellen eines neuen Tickets: Als nicht eingeloggter Nutzer direkt die Felder auf der Startseite ausfüllen und auf “Ticket erstellen” klicken. In der darauf folgenden Ticketansicht können noch Kommentare zum Ticket hinzugefügt werden. Da sich normale Nutzer nicht einloggen, kann sich ein Nutzer beim erneuten Aufrufen der Seite leider nicht automatisch all seine Tickets anzeigen lassen. Allerdings besteht die Möglichkeit das erstellte Ticket mithilfe der TicketID wiederzufinden, indem man den Pfad “https://<IPAdr>:<Port>/ticket?id=<TicketID>” in die Adressleiste des Browsers eingibt. Zudem bekommen die Nutzer über die Mail app, die an die WebAPI angeschlossen ist eine Email mit einem Link zu ihrem Ticket
  Angemeldete Nutzer können Tickets anlegen, indem sie in der Navigation “Ticket erstellen” auswählen.
- Zuteilen eines Bearbeiters: Diese Aktion kann nur von angemeldeten Nutzern in der Weboberfläche durchgeführt werden. Dazu kann unter dem Navigationspunkt “Alle Tickets” aus einer drop-down liste für jedes noch nicht zugewiesene Ticket ein Bearbeiter ausgewählt werden. Durch einen klick auf “Zuweisen” wird die Auswahl des Bearbeiters bestätigt. Bearbeiter, die gerade den Urlaubsmodus aktiviert haben können nicht ausgewählt werden.
- Tickets anzeigen (öffnen): Eingeloggte User bekommen auf dem Dashboard die Ihnen zugewiesenen Tickets angezeigt. Dort befindet sich für jedes Ticket auch ein button über den es geöffnet werden kann. Tickets können nur von dem ihnen zugewiesenen Bearbeitern geöffnet werden. Tickets die noch keinen Bearbeiter haben, können von jedem angemeldeten Nutzer geöffnet werden (über den Navigationspunkt “Alle Tickets”)
- Bearbeitungsstatus eines Tickets ändern: Es gibt drei Bearbeitungszustände:
  - Offen
  - In Bearbeitung

    - Geschlossen

  Ein Bearbeiter kann den Status der ihm Zugewiesenen Tickets jederzeit ändern, indem er das Ticket öffnet und aus einer Drop-Down Liste den Status wählt und seine Änderungen anschließend mit einem klick auf “Speichern” bestätigt. Zudem wird der Status bei bestimmten Events auch automatisch geändert. Wird ein bearbeiter einem Ticket zugewiesen, so ändert sich dessen Status automatisch auf “In Bearbeitung”. Wählt ein Bearbeiter auf seinem Dashboard für eines seiner Tickets “Ticket freigeben” so ändert sich der Status automatisch auf “Offen”. Werden zwei Tickets zusammengeführt, so wird eines automatisch geschlossen.
- Tickets zusammenführen: Für den Fall, dass ein Kunde ausversehen ein neues Ticket anlegt, anstatt das bereits bestehende zu kommentieren, können zwei Tickets, die den selben Kunden (selbe email) und den selben Bearbeiter haben, zusammengeführt werden. Dazu muss der Bearbeiter das fälschlicherweise angelegte Ticket öffnen und in der Drop-Down Liste “Zusammenführen mit:” das gewünschte Ursprungsticket auswählen. Ist die Drop-Down Liste leer, so hat der Bearbeiter kein anderes Ticket von diesem Kunden. Nach dem der Bearbeiter den Schritt mit “Speichern” bestätigt, wird das fälschlicherweise angelegte Ticket geschlossen, und die Nachricht (und alle Kommentare) in das Ursprungsticket kopiert.
- Urlaubsmodus aktivieren: Angemeldete Nutzer können auf ihrem Dashboard durch einen Klick auf “urlaubsmodus aktivieren” / “Urlaubsmodus deaktivieren” den Urlaubsmodus an und Ausschalten. Befindet sich ein Nutzer im Urlaubsmodus, so können ihm keine neuen Tickets zugewiesen werden.
- Anmerkung zum Feld “Antworttyp”: Wann immer man als Bearbeiter ein geöffnetes Ticket ändern oder kommentieren möchte, muss man das Feld Antworttyp auswählen. Interne Kommentare werden nicht eingeloggten Usern nicht angezeigt. Zudem werden für jegliche Änderungen und Kommentare die unter dem Antworttyp “intern” gespeichert werden keine emails an den Kunden versendet. Über änderungen, bei denen das Feld auf “extern” steht wird der Kunde per Email (abzugreifen über die web API) informiert

Command-Line-Interface (CLI)

- Geführter Modus: Wird das CLI ohne f oder s flag gestartet, so wird der Nutzer durch textpromts durch ein kleines Programm geführt. Er kann auswählen, ob er eine Mail an die Web API senden will (submit), ob er zur Verfügung stehende Mails von der Web API abfragen will (fetch), oder ob er das Programm beenden will (exit). Nach dem erfolgreichen Abschluss einer der ersten beiden Optionen hat der Nutzer erneut die Auswahl zwischen diesen drei Optionen. Wird die Fetch Option ausgeführt, so werden alle vom Server zur Verfügung gestellten Nachrichten in die Kommandozeile gedruckt. Wird die Submit Option gewählt, so wird der Nutzer nacheinander aufgefordert seine email adresse, eine Ticketnummer (falls vorhanden), den Betreff und die eigentliche Nachricht einzugeben. Dabei werden die Inputs einer einfachen Überprüfung unterzogen.

- Emails abfragen: Emails können auch direkt abgefragt werden, indem man das CLI mit dem -f flag (fetch) startet. Die zur verfügung stehenden Nachrichten werden dann sofort in die kommandozeile gedruckt.
- Nachrichten an den Server schicken: Auch das Senden von Nachrichten an den Server kann komplett über flags durchgeführt werden. Dazu muss das -s (submit) flag gesetzt sein. Das CLI liest dann die Flags “email”, “tID”, “subject” und “message” and baut daraus eine Nachricht, die direkt an den Server geschickt wird. dabei erfolgt keine Überprüfung.

Betrieb des Servers (Matr. 3040018)

Zuständigkeit: Aufsetzung des Servers und der Handler, Templating
Bearbeitete Dateien / Packages:
main, files, globals, server(server, handler), session, util(filehandler, hashing), www, structs, globals

Zusammenspiel der bearbeiteten Packages / Dateien:
Von mir wurde die grundlegende Struktur des Projekts mit der Aufteilung in verschiedene verteilte packages aufgebaut. Hierzu zählt die Verbringung des package main des Ticketsystems in die Struktur cmd/ticketsystem (nach Empfehlung in Vorlesung).
Des Weiteren wurden, um Circle Imports entgegen zu wirken, die Structs in das ebenso benannte package ausgelagert. Dieselbe Vorgehensweise wurde bei globals angewandt, wobei hier keine structs, sondern globale Variablen, wie etwa die Map der Tickets enthalten sind.
Als Standard für Dateiablage, sofern nicht abweichend konfiguriert, wurde der Ordner files definiert, in welchem in weiteren Unterordnern die Tickets, sowie die Nutzerdateien gespeichert werden.
Im Package server wurde der Programmcode für den generellen Betrieb des Servers geschrieben, sowie die Handler für die verschiedenen HTTP-Requests, welche durch die Anforderungen in der Aufgabenstellung angesprochen werden.
Davon abgekapselt ist im package session die entsprechende Verwaltung und Manipulation der Session, sowie der Cookies mit Speicherung der Session-Id zu finden.
Um die Packages server und session zu Unterstützen finden sich innerhalb des Ordners util die packages filehandler und hashing. Diese agieren genau ihren Namen entsprechend und sind für sämtliche Interaktion von und mit Dateien zuständig, sowie für das hashing des Passworts beim Einloggen mittels BCrypt.
Neben dieser Backend-Logik wurden noch die Elemente innerhalb des www-Ordners zur Darstellung eingefügt.
Die eigentliche UI-Schicht wurde mittels der GO-Templates realisiert, ohne Verwendung externer Libraries für Javascript oder CSS.
Hierbei werden verschachtelt die Verschiedenen Dateien mittels der Template-Logik miteinander verknüpft. Diese füllen dynamisch bei Aufruf die HTML-Elemente durch die übergebenen Strukturen mit Daten.

Javascript ist hauptsächlich für die Navigation zuständig mit Ausnahme der Verwendung von Ajax, welches zum Zuweisen von Tickets genutzt wird.

REST-Email API (Matr. 6694964)

Zuständigkeit: Mail API und verschiedene HTTP- und JSON-Utilities
Bearbeitete Dateien/Pakete: api/api_in, api/api_out, cmd/ticketsystem, files/mails, globals, mail_events, structs, util/filehandler, util/httptools, util/jsontools

Eingehende API zur Ticketerstellung

Über eine REST-Schnittstelle des Ticketsystems, die ein externer Dienst mittels einer HTTPS-Anfrage ansprechen kann, können Nachrichten in das System eingespeist werden. Der externe Dienst, in diesem Fall unser Kommandozeilentool, kann zwar keine Emails empfangen, jedoch kann es Emails erstellen und als JSON formatiert an die Ticketerstellungs-API (siehe `api_in.ReceiveMail`) senden. Die API erwartet die Parameter `from` (der Absender), `subject` (der Betreff des Tickets) und `message` (die eigentliche Nachricht). Diese werden akribisch nach Unstimmigkeiten (z.B. fehlende oder zu viele Parameter) durchsucht und lösen bei Bedarf einen Fehler aus. Jede fehlerhafte Anfrage erwirkt hierbei einen Rückgabestatus (Response-Code) von `400 Bad Request`. Überdies wird geprüft, ob die Absender-Email-Adresse gültig ist.

Sofern die Anfrage korrekt gestellt wurde, wird ein neues Ticket mit den jeweiligen Parametern erstellt und ein OK-Status als JSON wieder an den Aufrufer zurückgegeben. Zur Bestätigung der Erstellung wird eine noreply-Email an den Absender erstellt, sodass er sich sicher sein kann, dass sein Ticket erstellt wurde. Außerdem findet er in dieser Mail einen Link auf das Ticket, da nur angemeldete Benutzer eine Liste von Tickets sehen können. Weiterhin befindet sich in der Mail ein mailto-Link, mit dem eine Email erstellt werden kann, die eine neue Antwort auf das erstellte Ticket erstellen kann. Dafür muss der Betreff der Email eine bestimmte Syntax haben:

```
[Ticket "<ticket-id>"] <Ticket-Betreff>
```

Ist diese Syntax im Betreff gegeben und die Ticket-ID existiert, wird die Nachricht an dieses Ticket angehängt. Falls das Ticket den Status “Geschlossen” hat, wird er auf “Geöffnet” zurückgesetzt. Auch in diesem Fall wird eine Benachrichtigungs-Email an den Absender erstellt. Ein weiterer Vorteil dieser Methodik ist, dass ein Email-Ping-Pong verhindert werden kann, da automatische Antwortnachrichten (wie z.B. von Abwesenheitsassistenten) an die noreply-Email-Adresse geschickt werden und so keine Antwort gesendet wird.

Ausgehende API zur Versendung von Emails

Eine weitere API, die das Ticketsystem stellt, ist die Email-Versende-API. Der externe Dienst fragt aktiv nach Emails, die beim Server erstellt wurden, nach und kann diese dann versenden. Er verifiziert das erfolgreiche Versenden der Email mit einer Bestätigung an den Server, welcher diese Email schließlich entfernen kann. Das Ticketsystem Trivial Tickets bietet mit der Funktion `api_out.SendMail` eine Funktion an, die bei unterschiedlichen sogenannten Mail Events aufgerufen wird und daraufhin eine Email anfertigt. Zu den Mail Events gehören die Erstellung und Aktualisierung von Tickets, das Erstellen neuer Kommentare, die Zuweisung eines Bearbeiters und die Freigabe eines Tickets (siehe Paket `mail_events`). Mithilfe von zugewiesenen Mailvorlagen für alle Events werden die Benachrichtigungs-Emails mit den Informationen aus dem jeweiligen Ticket angereichert und die Email in eine Datei gespeichert. Über die API-Funktion `api_out.FetchMails` können die so generierten Emails mittels einer GET-Anfrage abgerufen werden. Die Ausgabe erfolgt im JSON-Format und sieht für eine Email wie folgt aus:

```
{
      "C6F88PJr3t": {
      "from": "no-reply@trivial-tickets.com",
      "id": "C6F88PJr3t",
      "message": "Sehr geehrter Kunde, sehr geehrte Kundin,
Ihr Ticket 'NicaWCh34F' ist erfolgreich erstellt worden.
Wenn Sie eine neuen Kommentar zu diesem Ticket schreiben wollen, nutzen
Sie bitte den folgenden Link:
mailto:support@trivial-tickets.com?subject=[...]",
      "subject": "[trivial-tickets] Schwierigkeiten mit PC",
      "to": "customer@mail.com"
      }
}
```

Die vom Server generierten Emails können dann vom Kommandozeilentool an den Empfänger (den Kunden) zugestellt werden. Ist eine Email erfolgreich versendet worden, wendet sich das Kommandozeilentool erneut dem Server zu und verifiziert das Versenden durch den Aufruf der API-Funktion `api_out.VerifyMailSent`. Hierbei muss die ID der Email übertragen werden, damit der Server die Email zuordnen kann. Der Server weiß nun, dass die Email verschickt wurde und kann daher seine zwischengespeicherte Kopie löschen.

Hilfsfunktionen

Für die API wurden unter `util/httptools` und `util/jsontools` einige nützliche Funktionen zum Umgang mit HTTP-Fehlerrückgaben implementiert. So kann man bspw. mit `httptools.JsonResponse` und `httptools.JsonError` direkt JSON-Nachrichten in die HTTP-Antwort schreiben, ohne sie manuell kodieren zu müssen.

Kommandozeilentool (Matr. 3478222)

Zuständigkeit: Command Line Interface
Bearbeitete Dateien: cli/communicationToServer, cli/IO, cmd/commandLineTool, utils/cliUtils, structs

Aufbau CLI: Die Kommunikation mit dem Nutzer läuft über das IO package. Es enthält Methoden um den Nutzer mit textpromts zu Eingaben aufzufordern, die anschließend Überprüft werden. Es enthält zudem auch Methoden um z.B. die abgefragten Nachrichten auf dem Bildschirm auszugeben. Das communicationToServer package ist für die Interaktion mit der Web API zuständig. Es enthält die Methoden makeGetRequest und makePostRequest, die das Absenden eines Post oder Get Requests kapseln. Bei ihrem Aufruf muss nur noch der Pfad auf der webseite angegeben werden, nicht mehr die ganze Adresse. Zudem prüfen sie auf verschieden mögliche Fehler. Der Aufrufer muss nur noch eine Fehlerprüfung übernehmen.
Die Ablauflogik des Programms findet man in commandLineTool.go. Die main Methode entscheidet basierend auf den gesetzten flags, ob direkt eine Methode aus dem communicationToServer package aufgerufen wird, oder ob das IO package verwendet wird um den Nutzer durch die Anwendung zu führen.
cliUtils stellt Methoden zur verfügung, die sowohl von IO als auch von commandLineTool benötigt werden.