Kontrakte

1. Model

Administration

1. addBooking
2. removeBooking
3. clean
4. AdHocDistribution.distribute
5. StandardDistribution.distribute
6. Timer.triggerBookingDistribution
7. run
8. run(Booking[0..*] :bookings)
9. filter
10. notify
11. toggleCheck(booking IBooking)

Operation: addBooking(Booking: booking)

Beschreibung: Hinzufügen einer Buchung zur Buchungsliste der Schedule.

Vorbedingung: Schedule und validierte Buchung existieren. (Sind nicht null)

Nachbedingung: Die Liste bookings von Schedule wurde um die übergebene valide Buchung erweitert.

Ergebnisse: Wahrheitswert entsprechend ob die Operation erfolgreich war oder nicht.

Ausnahmen: Schedule enthält die Buchung bereits.

Ausgaben: ---

Typ: Systemoperation

Querverweise: ---

Anmerkungen: ---

Operation: removeBooking(Booking: booking)

Beschreibung: Entfernt eine Buchung aus der Buchungsliste einer Schedule.

Vorbedingung: Schedule und zu entfernende Buchung existieren.

Nachbedingung: Buchung wurde aus der Schedule entfernt.

Ergebnisse: Wahrheitswert entsprechend ob die Operation erfolgreich war oder nicht.

Ausnahmen: Schedule enthält die Buchung nicht.

Ausgaben: ---

Typ: Systemoperation

Querverweise: ---

Anmerkungen: ---

Operation: clean(DateTime: now)

Beschreibung: Entferne alle abgelaufenen Buchungen aus der Buchungsliste der Schedule.

Vorbedingung: Schedule existiert.

Nachbedingung: Schedule enthält keine abgelaufenen Buchungen mehr, d. h. für jede Buchung buchung gilt: buchung.getEndDate() >= now.

Ergebnisse: Wahrheitswert entsprechend ob die Operation erfolgreich war oder nicht.

Ausnahmen: ---

Ausgaben: ---

Typ: Systemoperation

Querverweise: ---

Anmerkungen: ---

Operation: StandardDistribution.distribute(Booking[1..*]: bookings, Schedule: schedule)

Beschreibung: Verteile alle vorhandenen Buchungen in die Schedule.

Vorbedingung: Schedule existiert. Nichtleere Liste zu verteilender und valider Buchungen auf der entsprechenden Schedule existiert.

Nachbedingung: Übergebene Buchungen, wurden nun in die Buchungsliste der entsprechenden Schedule verteilt. Attribut “station” der Buchungen wurde bei der Zuweisung gesetzt. Der Startzeitpunkt der Buchungen wurde auf den Beginn des Ladevorgangs gesetzt und der Endzeitpunkt entsprechend auf das Ende.

Ergebnisse: Wahrheitswert entsprechend ob die Operation erfolgreich war oder nicht.

Ausnahmen: Eine der Buchungen ist bereits in der Schedule enthalten. (Operation addBooking(Booking: booking) gibt den Wahrheitswert false zurück.)

Ausgaben: ---

Typ: Systemoperation

Querverweise: ---

Anmerkungen: ---

Operation: AdHocDistribution.distribute(Booking[1..*]: bookings, Schedule: schedule)

Beschreibung: Verteile alle AdHoc-Buchungen in die Schedule.

Vorbedingung: Schedule existiert. Eine valide AdHoc Buchung auf dem zur Schedule gehörenden Standort wurde durchgeführt.

Nachbedingung: Die AdHoc Buchung wurde der Schedule hinzufefügt. Attribut “station” der Buchung wurde gesetzt. Der Startzeitpunkt der Buchungen wurde auf den Beginn des Ladevorgangs gesetzt und der Endzeitpunkt auf das Ende des Laden gesetzt

Ergebnisse: Wahrheitswert entsprechend ob die Operation erfolgreich war oder nicht.

Ausnahmen: Die Buchung ist bereits in der Schedule enthalten. (Operation addBooking(Booking: booking) gibt den Wahrheitswert false zurück.)

Ausgaben: ---

Typ: Systemoperation

Querverweise: ---

Anmerkungen: ---

Operation: Timer.triggerBookingDistribution()

Beschreibung: Aktiviere den Distributor regelmäßig zu einem vorgegebenen Zeitpunkt um normale Buchungen zu verteilen.

Vorbedingung: Distributor und zugehörige Schedule existieren. Es gibt zu verteilende Buchungen.

Nachbedingung: Die Methode run() wurde auf dem Distributor zum vorgegebenen Zeitpunkt aufgerufen.

Ergebnisse: Wahrheitswert entsprechend ob die Operation erfolgreich war oder nicht.

Ausnahmen: Es gibt keine Buchungen zum Verteilen. Die Kapazitäten sind erschöpft.

Ausgaben: ---

Typ: Systemoperation

Querverweise: ---

Anmerkungen: ---

Operation: AdHoc.triggerBookingDistribution()

Beschreibung: Aktiviere den Distributor um AdHoc Buchungen zu verteilen. Setze die Verteilungsstrategie des Distributors auf AdHoc.

Vorbedingung: Distributor und zugehörige Schedule existieren. Es wurde eine valide AdHoc Buchung aufgegeben.

Nachbedingung: Die Methode run() wurde auf dem Distributor zum vorgegebenen Zeitpunkt aufgerufen.

Ergebnisse: Wahrheitswert entsprechend ob die Operation erfolgreich war oder nicht.

Ausnahmen: Es gibt keine AdHoc-Buchung zum Verteilen. Die Kapazitäten sind erschöpft.

Ausgaben: ---

Typ: Systemoperation

Querverweise: ---

Anmerkungen: ---

Operation: run()

Beschreibung: Führe die Buchungsverteilung durch. Sorgt für die Benachrichtigung per E-Mail.

Vorbedingung: Es wurde eine Verteilungsstrategie gesetzt.

Nachbedingung: Es gibt im Buchungscache keine Buchungen mehr, die das Filterkriterium vom zugehörigen Filter erfüllen. Diese sind in dem Zugehörigen schedule. Der Notification Manager wurde für jede Booking informiert.

Ergebnisse: Wahrheitswert entsprechend ob die Operation erfolgreich war oder nicht.

Ausnahmen: ---

Ausgaben: ---

Typ: Systemoperation

Querverweise: ---

Anmerkung: ---

Operation: run(Booking[0..*] :bookings)

Beschreibung: Führe die Buchungsverteilung durch. Sorgt für die Benachrichtigung per E-Mail.

Vorbedingung: Es wurde eine Verteilungsstrategie gesetzt.

Nachbedingung: Im SChedule sind falls möglich die Bookings aus bookings zu finden. Der Notification Manager wurde für jede Booking informiert.

Ergebnisse: Wahrheitswert entsprechend ob die Operation erfolgreich war oder nicht.

Ausnahmen: ---

Ausgaben: ---

Typ: Systemoperation

Querverweise: ---

Anmerkung: ---

Operation: filter(Booking[1..*]: bookings, DateTime: date)

Beschreibung: Filter die Buchungsliste nach einem bestimmten Merkmal.

Vorbedingung: Die Buchungsliste ist nicht leer.

Nachbedingung: Es existiert eine Buchungsliste mit den gefilterten Buchungen.

Ergebnisse: Eine Liste and Buchungen, die das Filterkriterium erfüllen.

Ausnahmen: Kein Objekt erfüllt das Filterkriterium.

Ausgaben: ---

Typ: Systemoperation

Querverweise: ---

Anmerkung: O(n)

Operation: notify(Booking: booking, string: eventName)

Beschreibung: reicht die Informationen an die update von seinem listener weiter

Vorbedingung: Booking existiert, eventName existiert

Nachbedingung: ---

Ergebnisse: ---

Ausnahmen: ---

Ausgaben: ---

Typ: Systemoperation

Querverweise: ---

Anmerkung: ---

Operation: toggleCheck(IBooking: booking)

Beschreibung: setzt die Buchung auf aktiv bzw auf inaktiv und benachrichtigt den Benutzer

Vorbedingung: Booking existiert, es ist der richtige zeitraum für diese Aktion

Nachbedingung: aktiv der booking hat sich geändert

Ergebnisse: ---

Ausnahmen: ---

Ausgaben: ---

Typ: Systemoperation

Querverweise: ---

Anmerkung: ---

Simulation

1. generateDateTimeValues
2. addVehicle
3. deleteVehicle
4. addRushhour
5. deleteRushhour
6. updateWorkload
7. generateBooking
8. triggerBookinDistribution
9. Rushhour.run
10. calculataLocationWorkload
11. calculateSationWorkload
12. Simulation.run
13. init
14. ExecutedSzenario

Operation: generateDateTimeValues(DateTime: start, DateTime: end, int: bookings)

Beschreibung: generiert im bereich von start zu end bookings Zeitpunkte die normalverteilt sind.

Vorbedingung: start existiert, end existiert und ist größer als start, bookings existiert und ist größer 0

Nachbedingung: ---

Ergebnisse: Eine Liste von DateTimes zwischen start und end (Normalverteilt)

Ausnahmen: ---

Ausgaben: ---

Typ: Systemoperation

Querverweise: ---

Anmerkung: ---

Operation: addVehicle(Vehicle: vehicle)

Beschreibung: fügt vehicle der Liste der enthaltenen Vehicle hinzu

Vorbedingung: vehicle existiert und ist valide.

Nachbedingung: vehicle wurde der Liste der Vehicle hinzugefügt

Ergebnisse: Wahrheitswert entsprechend ob die Operation erfolgreich war oder nicht.

Ausnahmen: ---

Ausgaben: ---

Typ: Systemoperation

Querverweise: ---

Anmerkung: ---

Operation: deleteVehicle(Vehicle: vehicle)

Beschreibung: Entfernt vehicle aus der Liste der enthaltenen Vehicle.

Vorbedingung: vehicle existiert und ist valide.

Nachbedingung: vehicle wurde der Liste der Vehicle entfernt.

Ergebnisse: Wahrheitswert entsprechend ob die Operation erfolgreich war oder nicht.

Ausnahmen: vehicle ist nicht in der Liste der Vehicle.

Ausgaben: ---

Typ: Systemoperation

Querverweise: ---

Anmerkung: ---

Operation: addRushhour(Rushhour: rushhour)

Beschreibung: Fügt rushhour der Liste der enthaltenen Rushhour hinzu.

Vorbedingung: rushhour existiert und ist valide.

Nachbedingung: rushhour wurde der Liste der Rushhour hinzugefügt.

Ergebnisse: Wahrheitswert entsprechend ob die Operation erfolgreich war oder nicht.

Ausnahmen: rushhour ist bereits in der Liste der Rushhour enthalten.

Ausgaben: ---

Typ: Systemoperation

Querverweise: ---

Anmerkung: ---

Operation: deleteRushhour(Rushhour: rushhour)

Beschreibung: Entfernt rushhour aus der Liste der enthaltenen Rushhour.

Vorbedingung: rushhour existiert und ist valide.

Nachbedingung: rushhour wurde der Liste der rushhour entfernt.

Ergebnisse: Wahrheitswert entsprechend ob die Operation erfolgreich war oder nicht.

Ausnahmen: rushhour ist nicht in der Liste der Rushhour .

Ausgaben: ---

Typ: Systemoperation

Querverweise: ---

Anmerkung: ---

Operation: updateWorkload(double: location, double[1..*]: station)

Beschreibung: aktualisiert den Workload, erweitert stationWorkload und locationWorkload um jeweils einen eintrag

Vorbedingung: location existiert und ist valide. station ist valide und existiert.

Nachbedingung: location ist der neue letzte eintrag von locationWorkload station ist der neue letzte eintrag von stationWorkload

Ergebnisse: Wahrheitswert entsprechend ob die Operation erfolgreich war oder nicht.

Ausnahmen: ---

Ausgaben: ---

Typ: Systemoperation

Querverweise: ---

Anmerkung: ---

Operation: generateBookings(Scenario: scenario)

Beschreibung: Erstellt eine Liste von Booking, die zum Scenario passen für capacity, plugs, socStart, socEnd wird ein Vehicle aus Szenario.vehicles random gewählt. Für start und end wird folgendermaßen vorgegangen: Es werden für jeden Tag Szenario.bookingCountPerDay Bookings erstellt, diese haben eine linear verteilte startTime und eine zufällig generierten endTime, sollte es Rushour geben, drücken diese die Verteilung enger zusammen (mithilfe der run). Die priority wird zufallsgeneriert und mit probabilityVIP wird sie auf VIP (mit probalbilityAdhoc wird eine Adhoc erstellt statt einer booking, ohne Einteilung) gesetzt mit probabilityGuest wird sie auf GUEST gesetzt in jedem anderen fall auf EMPLOYEE. Alle Buchungen stehen auf active = true.

Vorbedingung: Scenario ist valide und existiert

Nachbedingung: ---

Ergebnisse: Liste der erstellten Bookings

Ausnahmen: Die summe der Rushour.bookings ist größer als der bookingCountPerDay (es werden bookings an die Rushours per first come first serve verteilt).

Ausgaben: ---

Typ: Systemoperation

Querverweise: ---

Anmerkung: ---

Operation: triggerBookingDistribution()

Beschreibung: Wählt die Bookings aus die Verteilt werden sollen und ruft dann Distributer.run(bookings) auf.

Vorbedingung: PendingBookings existiert und enthält noch Bookings

Nachbedingung: Die ausgewählten Bookings sind nicht mehr in pendingBookings und sind wenn es möglich zugeteilt.

Ergebnisse: Wahrheitswert entsprechend ob die Operation erfolgreich war oder nicht.

Ausnahmen: ---

Ausgaben: ---

Typ: Systemoperation

Querverweise: ---

Anmerkung: Zum Auswahlverfahren es müssen Bookings der nächsten 30 Tage (von tickCount*tickLength + ExecutedScenario.start) ausgewählt werden allerdings nicht alle, hierbei muss für den auswahl algorithmus folgendes gelten: Es werden auf jedenfall alle Bookings für den nächsten Tag ausgewählt, er muss schnell sein, darf NICHT randomisiert sein und Organische Verteilung vorweisen. Ein Vorschlag um das zu erreichen ist jede Booking auszuwählen die in PendingBookings einen Index haben der eine Primzahl ist.

Operation: Rushhour.run()

Beschreibung: ruft generateDateTimeValues auf und übergibt bookings, end, start

Vorbedingung: ---

Nachbedingung: ---

Ergebnisse: gibt Rückgabewert von generateDateTimeValue zurück.

Ausnahmen: ---

Ausgaben: ---

Typ: Systemoperation

Querverweise: ---

Anmerkung: ---

Operation:calculateLocationWorkload(now: DateTime)

Beschreibung: berechnet den Workload der Location in der Zeitspann der länge tickLength von tickCount*tickLength+scenario.start innerhalb der Simulation und fügt diese dem array scenario.locationWorkload zu

Vorbedingung: tickLength, tickCount, scenario existieren und scenario hat start und ist valide

Nachbedingung: scenario.locationWorkload hat einen eintrag mehr

Ergebnisse: Wahrheitswert entsprechend ob die Operation erfolgreich war oder nicht.

Ausnahmen: ---

Ausgaben: ---

Typ: Systemoperation

Querverweise: ---

Anmerkung: maximal O(n)!

Operation:calculateStationWorkload(now: DateTime)

Beschreibung: berechnet den Workload der Stations in der Zeitspann der länge tickLength von tickCount*tickLength+scenario.start innerhalb der Simulation und fügt diese dem array scenario.locationWorkload zu

Vorbedingung: tickLength, tickCount, scenario existieren und scenario hat start und ist valide

Nachbedingung: scenario.stationWorkload hat einen eintrag mehr

Ergebnisse: gibt die Auslastung der Stations zurück

Ausnahmen: ---

Ausgaben: ---

Typ: Systemoperation

Querverweise: ---

Anmerkung: die Reihenfolge innerhalb des rückgabe arrays entspricht der Reihenfolge wenn man Location.zones[0].stations[0] über Location.zones[0].stations[n] hinzu Location.zones[n].stations[n] durchläuft.

Operation: Szenario.run()

Beschreibung: Es beginnt eine schleife in der immer triggerBookingDistribution (sollten pendingBookings nicht leer sein) danach die adhocBuchungen für diesen Tag, calculateLocationWorkload und dann calculateStationWorkload aufgerufen wird und dann tickCount um eins erhöht wird. Diese Schleife endet sobald tickCount > scenario.duration ist nun werden noch die nicht erfüllten Bookings berechnet und gesetzt

Vorbedingung: scenario existiert und ist valide

Nachbedingung: szenario ist vollständig simuliert, in pendingBookings ist keine Booking mehr.

Ergebnisse: Wahrheitswert entsprechend ob die Operation erfolgreich war oder nicht.

Ausnahmen: ---

Ausgaben: ---

Typ: Systemoperation

Querverweise: ---

Anmerkung: maximal O(n)!

Operation: init()

Beschreibung: ruft clear des Szenarios auf, filtert aus bookings alle Adhoc bookings erstellt damit eine Liste und setzt diese liste als adHocBookings und den rest als PendingBookings. Und ruft run auf.

Vorbedingung: scenario existiert und ist valide

Nachbedingung: szenario ist vollständig simuliert, in pendingBookings ist keine Booking mehr.

Ergebnisse: ---

Ausnahmen: ---

Ausgaben: ---

Typ: Systemoperation

Querverweise: ---

Anmerkung:

Operation: clear()

Beschreibung: Erstellt eine Tiefe kopie der liste generatedBookings, und ersetzt damit bookings außerdem werden locationWorkload und stationWorkload geleert. Und fulfilledRequest wird auf 0 gesetzt.

Vorbedingung: generatedBookings existiert und enthält mindestens eine Buchung

Nachbedingung: das Szenario ist auf den Zustand vor der Simulation zurückgesetzt

Ergebnisse: ---

Ausnahmen: ---

Ausgaben: ---

Typ: Systemoperation

Querverweise: ---

Anmerkung:

Operation: ExecutedSzenario(Szenario:szenario)

Beschreibung: Konstruktor von ExecutedSzenario übernimmt die Daten aus dem szenario kopiert diese ruft erzeugt mithilfe des Generator die generatedBooking Liste. Inizialisiert die Listen.

Vorbedingung: szenario existiert und ist valide

Nachbedingung: ExecutedSzenario existiert

Ergebnisse: ---

Ausnahmen: ---

Ausgaben: ---

Typ: Systemoperation

Querverweise: ---

Anmerkung:

History

1. analyze
2. createSuggestion
3. calcBookingSuccessRate
4. calcUnnecessaryWorkload
5. calcNecessaryWorkload
6. calcPlugDistributionAccepted
7. calcPlugDistributionDeclined
8. Suggestion

Operation: analyze(IEvaluatable: scenario):

Beschreibung: Erstellt eine Evaluation, seztzt es als globale variable und befüllt diese.

Vorbedingung: scenario existiert und ist valide.

Nachbedingung: eine neue valide Evaluation existiert.

Ergebnisse: die erstellte Evaluation

Ausnahmen: ---

Ausgaben: ---

Typ: Systemoperation

Querverweise: ---

Anmerkung:

Operation: createSuggestion()

Beschreibung: Erstellt eine neue Suggestion. Berechnet dafür die Art und wie viele Stationen/ Zonen hinzu-/ wegkommen

Vorbedingung: die benötigte werte sind in der evaluation gespeichert, diese exitiert und ist Valide.

Nachbedingung: eine neue valide Suggestion existiert.

Ergebnisse: die erstellte Suggestion

Ausnahmen: ---

Ausgaben: ---

Typ: Systemoperation

Querverweise: ---

Anmerkung:

Operation:calcBookingSuccessRate()

Beschreibung: berechnet die Booking Succsess Rate

Vorbedingung: Die benötigten Werte sind über die Evaluation abfragbar

Nachbedingung: ---

Ergebnisse: die Berechnete rate zwischen 0 und 100

Ausnahmen: die Rate liegt nicht zwischen 0 und 100

Ausgaben: ---

Typ: Systemoperation

Querverweise: ---

Anmerkung: ---

Operation:calcUnnecessaryWorkload()

Beschreibung: berechnet die UnnecessaryWorkload

Vorbedingung: Die benötigten Werte sind über die Evaluation abfragbar

Nachbedingung: ---

Ergebnisse: der berechnete wert zwischen 0 und 100

Ausnahmen: der Wert liegt nicht zwischen 0 und 100

Ausgaben: ---

Typ: Systemoperation

Querverweise: ---

Anmerkung: ---

Operation: calcPlugDistributionAccepted()

Beschreibung: berechnet die verteilung der Plugs unter den Akzeptierten Buchungen

Vorbedingung: Die benötigten Werte sind über die Evaluation abfragbar

Nachbedingung: Der berechnete Array hat die Länge PlugTyp.Count

Ergebnisse: Der berechnete Array

Ausnahmen: ---

Ausgaben: ---

Typ: Systemoperation

Querverweise: ---

Anmerkung: ---

Operation: calcPlugDistributionDeclined()

Beschreibung: berechnet die verteilung der Plugs unter den nicht Akzeptierten Buchungen

Vorbedingung: Die benötigten Werte sind über die Evaluation abfragbar

Nachbedingung: Der berechnete Array hat die Länge PlugTyp.Count

Ergebnisse: Der berechnete Array

Ausnahmen: ---

Ausgaben: ---

Typ: Systemoperation

Querverweise: ---

Anmerkung: ---

Operation: Suggestion(int stations, int: zones)

Beschreibung: Konstruktor der Klasse Suggestion generiert den suggestion string mithilfe der übergabeparameter

Vorbedingung: ---

Nachbedingung: eine neue Suggestion existiert.

Ergebnisse: ---

Ausnahmen: ---

Ausgaben: ---

Typ: Systemoperation

Querverweise: ---

Anmerkung: ---

Infrastructure

1. addZone
2. deleteZone
3. addStation
4. deleteStation
5. addPlug
6. deletePlug
7. CompareTo

Operation:	addZone(Zone: zone) : bool
Beschreibung:	Hinzufügen einer Zone zu Zonenliste einer Location.
Vorbedingung:	Eine hinzuzufügende valide Zone existiert .
Nachbedingung:	Die Zonenliste wurde um Zone erweitert.
Ergebnisse:	Wahrheitswert entsprechend ob die Operation erfolgreich war oder nicht.
Ausnahmen:	Zone ist bereits enthalten.
Ausgaben:	---
Typ:	Systemoperation
Querverweise:	---
Anmerkung:	---

Operation:	deleteZone(Zone: zone): bool
Beschreibung:	Entfernen einer Zone aus der Zonenliste.
Vorbedingung:	Zone existiert.
Nachbedingung:	Zone ist aus der Zonenliste entfernt.
Ergebnisse:	Wahrheitswert entsprechend ob die Operation erfolgreich war oder nicht.
Ausnahmen:	Zone ist nicht in der Zonenliste enthalten.
Ausgaben:	---
Typ:	Systemoperation
Querverweise:	---
Anmerkung:	---

Operation:	addStation(Station:station) : bool
Beschreibung:	Hinzufügen von Station zur Stationenliste.
Vorbedingung:	Valide Station existiert.
Nachbedingung:	Station ist in der Stationenliste enthalten.
Ergebnisse:	Wahrheitswert entsprechend ob die Operation erfolgreich war oder nicht.
Ausnahmen:	Station ist bereits enthalten.

Ausgaben:	Systemoperation
Typ:	---
Querverweise:	---
Anmerkung:	

Operation:	deleteStation(Station:station): bool
Beschreibung:	Entfernen einer Station aus der Stationenliste”.
Vorbedingung:	Valide Station existiert.
Nachbedingung:	Station wurde aus der Stationenliste entfernt.
Ergebnisse:	Wahrheitswert entsprechend ob die Operation erfolgreich war oder nicht.
Ausnahmen:	Station ist nicht in der Stationenliste enthalten.
Ausgaben:	---
Typ:	Systemoperation
Querverweise:	---
Anmerkung:	---

Operation:	addPlug(Plug: plug): bool
Beschreibung:	Hinzufügen von Plug zu der Plugliste.
Vorbedingung:	Valider Plug existiert.
Nachbedingung:	Plug wurde zu der Plugliste hinzugefügt.
Ergebnisse:	Wahrheitswert entsprechend ob die Operation erfolgreich war oder nicht.
Ausnahmen:	Plug ist bereits in der Plugliste enthalten.
Ausgaben:	---
Typ:	Systemoperation
Querverweise:	---
Anmerkung:	---

Operation:	deletePlug(Plug: plug) : bool
Beschreibung:	Entfernen von Plug der Liste “plugs”.
Vorbedingung:	Plug existiert.
Nachbedingung:	Plug wurde aus der Liste “plugs” entfernt.
Ergebnisse:	Wahrheitswert entsprechend ob die Operation erfolgreich war oder nicht.
Ausnahmen:	Plug ist nicht in der Liste “plugs” enthalten.
Ausgaben:	---
Typ:	Systemoperation
Querverweise:	---
Anmerkung:	---

Operation:	compareTo(object obj) : bool
Beschreibung:	vergleicht das obj mit der Zone.
Vorbedingung:	obj ist eine Zone
Nachbedingung:	---
Ergebnisse:	Wahrheitswert entsprechend ob der char bei site gleich war
Ausnahmen:	---
Ausgaben:	---
Typ:	Systemoperation
Querverweise:	---
Anmerkung:	---

Communication

1. update()
2. generateMessageAccepted
3. generateMessageDeclined
4. generateMessageCheckIn
5. generateMessageCheckOut

6. sendMessage

Operation:update(Booking :booking, string: eventName)

Beschreibung: entscheidet mithilfe des eventNames welche Nachricht an die E-mail Adresse bookings.user geschickt werden soll und sorgt dafür das diese verschickt wird

Vorbedingung: booking ist valide, eventName ist ein eventName und existiert

Nachbedingung: der Messenger verschickt eine E-Mail

Ergebnisse: ---

Ausnahmen: booking.User ist nicht gesetzt (es passiert einfach nichts)

Ausgaben: ---

Typ: ---

Querverweise: ---

Anmerkungen: ---

Operation: generateMessageAccepted(Booking :booking)

Beschreibung:generiert mithilfe von commands und den details aus booking die Nachricht für den User

Vorbedingung: booking ist valide und existiert

Nachbedingung: ---

Ergebnisse: Nachricht für den User

Ausnahmen: ---

Ausgaben: ---

Typ: ---

Querverweise: ---

Anmerkungen: ---

Operation: generateMessageDeclined(Booking :booking)

Beschreibung:generiert mithilfe von commands und den details aus booking die Nachricht für den User

Vorbedingung: booking ist valide und existiert

Nachbedingung: ---

Ergebnisse: Nachricht für den User

Ausnahmen: ---

Ausgaben: ---

Typ: ---

Querverweise: ---

Anmerkungen: ---

Operation: generateMessageCheckIn(Booking :booking)

Beschreibung:generiert mithilfe von commands und den details aus booking die Nachricht für den User

Vorbedingung: booking ist valide und existiert

Nachbedingung: ---

Ergebnisse: Nachricht für den User

Ausnahmen: ---

Ausgaben: ---

Typ: ---

Querverweise: ---

Anmerkungen: ---

Operation: generateMessageCheckOut(Booking :booking)

Beschreibung:generiert mithilfe von commands und den details aus booking die Nachricht für den User

Vorbedingung: booking ist valide und existiert

Nachbedingung: ---

Ergebnisse: Nachricht für den User

Ausnahmen: ---

Ausgaben: ---

Typ: ---

Querverweise: ---

Anmerkungen: ---

Operation: sendMessage(message: string, user: string)

Beschreibung: sendet die Nachricht message an die E-Mail user

Vorbedingung: message existiert, user existiert und ist eine E-Mail

Nachbedingung: Der richtige Nutzer erhält eine E-Mail.

Ergebnisse: ---

Ausnahmen: ---

Ausgaben: ---

Typ: ---

Querverweise: ---

Anmerkungen: ---

User

1. getUserPriority

Operation:getUserTyp(String: email)

Beschreibung: Ließt aus einem Textdokument die Priorität des Nutzers. Ist diese nicht enthalten wird die Rolle Gast zurück gegeben.

Vorbedingung: Email existiert

Nachbedingung: ---

Ergebnisse: Rolle des Users

Ausnahmen: ---

Ausgaben: ---

Typ: ---

Querverweise: ---

Anmerkungen: ---

2. Controller

HomeController

1. Index
2. Imprint
3. Help
4. Login

Operation: Index()

Beschreibung: gibt die Home Seite zurück.

Vorbedingung: ---

Nachbedingung: Der User sieht die Home Seite

Ergebnisse: Die Seite Index wird zurückgegeben

Ausnahmen: ---

Ausgaben: ---

Typ: ---

Querverweise: ---

Anmerkungen: ---

Operation: Imprint()

Beschreibung: gibt die Impressum Seite zurück.

Vorbedingung: ---

Nachbedingung: Der User sieht die Impressum Seite

Ergebnisse: Die Seite Imprint wird zurückgegeben

Ausnahmen: ---

Ausgaben: ---

Typ: ---

Querverweise: ---

Anmerkungen: ---

Operation: Help()

Beschreibung: gibt die Help Seite zurück.

Vorbedingung: ---

Nachbedingung: Der User sieht die Help Seite

Ergebnisse: Die Seite Help wird zurückgegeben

Ausnahmen: ---

Ausgaben: ---

Typ: ---

Querverweise: ---

Anmerkungen: ---

Operation: Login(string: email)

Beschreibung: Login vorgang des Users

Vorbedingung: email ist nicht leer, email ist eine valide email

Nachbedingung: Rolle wurde als SessionVariable gesetzt, der Nutzer sieht die Admin.Dashboard Seite sollte er die Rolle PLANER haben, sonst sieht er die Booking.Index Seite.

Ergebnisse: ---

Ausnahmen: ---

Ausgaben: ---

Typ: ---

Querverweise: ---

Anmerkungen: ---

BookingController

Index
Create
Post
Edit
Delete
ToggleCheck

Operation: Index()

Beschreibung: Erstellt ein DashboardViewModel und gibt die Booking.Index Seite zurück, sollte seine Rolle PLANER sein landet er auf der Admin.Dashboard Seite.

Vorbedingung:

Nachbedingung: Der User sieht die Booking.Index Seite auf der seine Buchungen angezeigt werden

Ergebnisse: Die Seite Booking.Index wird zurückgegeben

Ausnahmen: Der User hat sich noch nicht eingeloggt

Ausgaben: ---

Typ: ---

Querverweise: ---

Anmerkungen: ---

Operation: Create()

Beschreibung: Erstellt die BookingCreateViewModel und gibt die Booking.Create Seite zurück

Vorbedingung: Der User ist eingeloggt, und ist berechtigt diese Seite zu sehen

Nachbedingung: Der User sieht die Booking.Create Seite auf der er eine neue Buchung erstellen kann.

Ergebnisse: Die Seite Booking.Create wird zurückgegeben

Ausnahmen: ---

Ausgaben: ---

Typ: ---

Querverweise: ---

Anmerkungen: ---

Operation: Post(Booking: booking)

Beschreibung: gibt die Booking.Index Seite zurück und fügt die booking in den Cache hinzu

Vorbedingung: Der User ist eingeloggt, booking existiert.

Nachbedingung: Der User sieht die Booking.Index Seite auf der er seine Buchungen sehen kann, auf der auch die neue Buchung zu sehen ist.

Ergebnisse: Die Seite Booking.Index wird zurückgegeben

Ausnahmen: booking ist nicht valide.

Ausgaben: ---

Typ: ---

Querverweise: ---

Anmerkungen: ---

Operation: Edit(Booking: booking)

Beschreibung: gibt die Booking.Create Seite zurück

Vorbedingung: Der User ist eingeloggt, booking existiert, ist im cache und ist valide

Nachbedingung: Der User sieht die Booking.Create Seite auf der er seine Buchung bearbeiten kann. Das Formular ist ausgefüllt mit den Werten aus booking. booking ist nicht mehr im Cache

Ergebnisse: Die Seite Booking.Create wird zurückgegeben

Ausnahmen: ---

Ausgaben: ---

Typ: ---

Querverweise: ---

Anmerkungen: ---

Operation: Delete(Booking: booking)

Beschreibung: gibt die Booking.Index Seite zurück.

Vorbedingung: Der User ist eingeloggt, booking existiert, ist im cache und ist valide

Nachbedingung: Der User sieht die Booking.Index auf der er seine Buchungen sehen kann, booking ist nicht mehr zu sehen. Booking ist nicht mehr im Cache

Ergebnisse: Die Seite Booking.Create wird zurückgegeben

Ausnahmen: ---

Ausgaben: ---

Typ: ---

Querverweise: ---

Anmerkungen: ---

Operation: ToggleCheck(Booking: booking)

Beschreibung: Regelt den Check-In, Check-Out Vorgang.

Vorbedingung: Der User ist eingeloggt, booking existiert, es ist der richtige Zeitpunkt für diese Aktion.

Nachbedingung: Der User sieht die Booking.Index Seite auf der er seine Buchungen sehen kann. booking.active ist true sollte es vorher false gewesen sein und false sollte es vorher true gewesen sein.

Ergebnisse: Die Seite Booking.Index wird zurückgegeben

Ausnahmen: ---

Ausgaben: ---

Typ: ---

Querverweise: ---

Anmerkungen: ---

AdminController

1. Index()

Operation: Index() **Beschreibung:** Erstellt ein DashboardViewModel und gibt die Admin.Dashboard Seite zurück, sollte seine Rolle nicht PLANER sein landet er auf der Booking.Index Seite. **Vorbedingung:** --- **Nachbedingung:** Der User sieht die Admin.Dashboard Seite auf der alle Buchungen angezeigt werden **Ergebnisse:** Die Seite Admin.Dashboard wird zurückgegeben **Ausnahmen:** Der User hat sich noch nicht eingeloggt (zur Home.Index weiterleiten) **Ausgaben:** --- **Typ:** --- **Querverweise:** --- **Anmerkungen:** ---

3. ViewModel

4. Persistence