Kontrakte

[Administration]

Operation:	addBooking(chargingZone, user, maxCapacity, plugType, startTime, endTime, socStart)
Beschreibung:	Erstellen und hinzufügen einer Buchung
Vorbedingung:	Administration besteht
Nachbedingung:	Buchung wurde erstellt und der Administration hinzugefügt
Ergebnisse:	Wahrheitswert, ob die Buchung akzeptiert wurde.
Ausnahmen:	Ungültige Ladezone, Nutzer, MaxKapazität, Steckertyp, Startzeit, Endzeit
Ausgaben:	-
Typ:	Systemoperation
Querverweise:	-
Anmerkungen:	Domi

Operation:	removeBooking(id)
Beschreibung:	Entfernen einer Buchung
Vorbedingung:	Mindestens eine Buchung in Administration vorhanden; Id muss valide sein
Nachbedingung:	Buchung wurde aus der Administration entfernt.
Ergebnisse:	Wahrheitswert ob Buchung entfernt wurde
Ausnahmen:	Angeforderte Buchung nicht vorhanden
Ausgaben:	-
Typ:	Systemoperation
Querverweise:	-
Anmerkungen:	Domi

Operation:	editBooking(id, chargingZone, maxCapacity, plugType, startTime, endTime)
Beschreibung:	Bearbeiten einer Buchung
Vorbedingung:	Mindestens eine Buchung in Adminstration vorhanden
Nachbedingung:	Attribute der Buchung wurde verändert
Ergebnisse:	Wahrheitswert ob Buchung erfolgreich geändert wurde
Ausnahmen:	Angeforderte Buchung nicht vorhanden
Ausgaben:	-
Typ:	Systemoperation
Querverweise:	-
Anmerkungen:	Domi

Operation:	makeSuggestion(booking)
Beschreibung:	Erstellt eine Vorschlaf für eine Buchung
Vorbedingung:	Eine Buchungsanfrage wurde abgelehnt;
Nachbedingung:	Buchung wurde erstellt und er der Nutzer benachrichtigt Anhand der Buchung b wurde der best mögliche Ladeplatz-Vorschlag erstellt
Ergebnisse:	Ladeplatz-Vorschlag für die Buchung
Ausnahmen:	-
Ausgaben:	-
Typ:	Systemoperation
Querverweise:	-
Anmerkungen:	Uli

Operation:	updateTimetable(bookings)
Beschreibung:	Fügt die Buchungen bs in die Timetable hinzu und aktualisiert diese
Vorbedingung:	Buchungen bs wurden erfolgreich angelegt und bestätigt
Nachbedingung:	Buchungen bs wurden erstellt Buchungen bs wurden der Timetable hinzugefügt
Ergebnisse:	Neues Set an Buchungen bs
Ausnahmen:	-
Ausgaben:	**-**
Typ:	Systemoperation
Querverweise:	-
Anmerkungen:	Uli

Operation:	loadBookings()
Beschreibung:	Lädt alle Buchungen erneut ein
Vorbedingung:	-
Nachbedingung:	Buchungen wurden geladen
Ergebnisse:	Wahrheitswert ob Buchungen erfolgreich geladen wurden
Ausnahmen:	-
Ausgaben:	Gibt die Werte der Buchung an die View zurück
Typ:	Systemoperation
Querverweise:	Buchungsübersicht anzeigen
Anmerkungen:	Uli

Operation:	saveBooking()
Beschreibung:	Speichert die Änderungen an einer Buchung
Vorbedingung:	Buchung wurde erstellt
Nachbedingung:	Änderungen an der Buchung (falls vorhanden) wurden gespiechert Timetable wurde aktualisiert
Ergebnisse:	Speichert die Daten zu einer Booking ab
Ausnahmen:	-
Ausgaben:	-
Typ:	Systemoperation
Querverweise:	Buchung anlegen/bearbeiten
Anmerkungen:	Uli

Operation:	messageUser(user)
Beschreibung:	benachrichtig den User unter mehreren Umständen
Vorbedingung:	user muss existieren
Nachbedingung:	-
Ergebnisse:	-
Ausnahmen:	-
Ausgaben:	Schickt Nachricht an User
Typ:	Systemoperation
Querverweise:	-
Anmerkungen:	Thomas

Operation:	checkIn(id)
Beschreibung:	Nutzer bestätigt, dass er seine Buchung wahrnimmt; Überprüft auch ob er Zeit einhält
Vorbedingung:	Buchung mit der entsprechenden ID muss existieren StartTime liegt max. 15 Min in der Zukunft
Nachbedingung:	endTime lieg in der Zukunft
Ergebnisse:	Wahrheitswert ob Checkin erfolgreich war
Ausnahmen:	Id existiert nicht
Ausgaben:	User wurde Erfolgreich eingecheckt
Typ:	Systemoperation
Querverweise:	-
Anmerkungen:	Thomas

Operation:	checkOut(id)
Beschreibung:	Nutzer bestätigt, dass er den Ladestandtort verlässt
Vorbedingung:	Buchung mit der entsprechenden ID muss existieren; muss sich vorher einchecken
Nachbedingung:	Buchung liegt in der Vergangenheit
Ergebnisse:	Wahrheitswert ob Checkout erfolgreich war
Ausnahmen:	Id existiert nicht
Ausgaben:	User wurde Erfolgreich ausgecheckt
Typ:	Systemoperation
Querverweise:	-
Anmerkungen:	Thomas

Operation:	getPriority(email)
Beschreibung:	es wird die Priorität der Buchungsanfrage des Nutzers bestimmt
Vorbedingung:	email muss existeren
Nachbedingung:	-
Ergebnisse:	Priorität
Ausnahmen:	-
Ausgaben:	-
Typ:	Systemoperation
Querverweise:	-
Anmerkungen:	Thomas

[Timetable]

Operation:	scheduleBookings()
Beschreibung:	Verteilt die aufgegebenen Buchungsanfragen und bestätigt diese oder lehnt diese ab
Vorbedingung:	Es muss mindestens eine Buchungsanfrage existieren
Nachbedingung:	Verteilung der Buchungen hat stattgefunden
Ergebnisse:	Wahrheitswert ob Verteilung erfolgreich war
Ausnahmen:	-
Ausgaben:	Buchungsreihenfolge kann nun eingesehen werden
Typ:	Systemoperation
Querverweise:	-
Anmerkungen:	Thomas

Operation:	scheduleAdhocBookings()
Beschreibung:	Verteilt die aufgegebenen Buchungsanfragen sofort auf die noch freien Slots und bestätigt diese oder lehnt diese ab
Vorbedingung:	Es muss mindestens eine Buchungsanfrage existieren
Nachbedingung:	-
Ergebnisse:	Wahrheitswert ob Verteilung erfolgreich war
Ausnahmen:	-
Ausgaben:	Buchungsreihenfolge kann nun eingesehen werden und wurde aktualisiert
Typ:	Systemoperation
Querverweise:	-
Anmerkungen:	Thomas

[Infrastructure]

Operation:	addLocation(name)
Beschreibung:	-
Vorbedingung:	-
Nachbedingung:	Location existiert
Ergebnisse:	eine Localisation wurde kreiert und er Infrastruktur hinzugefügt
Ausnahmen:	Es wurde kein name übergeben.
Ausgaben:	-
Typ:	Systemoperation
Querverweise:	-
Anmerkungen:	Peter

Operation:	removeLocation(id)
Beschreibung:	Löscht eine Location anhand der ID
Vorbedingung:	Id muss existieren; Eine Location muss in der Liste existieren
Nachbedingung:	Die Location inklusive ihrer Id wird gelöscht, sowie alle dazugehörigen Charging Zones, Stations und plugs. Zusätzlich wird die Location aus jeder Liste in den Infrastrukturen entfernt
Ergebnisse:	Wahrheitswert ob Location erfolgreich gelöscht wurde
Ausnahmen:	Es wurde keine Location mit passender ID gefunden
Ausgaben:	-
Typ:	Systemoperation
Querverweise:	-
Anmerkungen:	Peter

Operation:	Infrastructure()
Beschreibung:	Konstruktor Ruft getInstance() auf
Vorbedingung:	-
Nachbedingung:	-
Ergebnisse:	-
Ausnahmen:	-
Ausgaben:	-
Typ:	Systemoperation
Querverweise:	-
Anmerkungen:	Thomas

Operation:	getInstance()
Beschreibung:	Erstellt ein neues Infrastructure Objekt
Vorbedingung:	Es darf kein Infrastructure Objekt existieren
Nachbedingung:	Es existiert jetzt ein Infrastructure Objekt
Ergebnisse:	Es existiert ein Infrastructure Objekt
Ausnahmen:	-
Ausgaben:	-
Typ:	Systemoperation
Querverweise:	-
Anmerkungen:	Thomas

[Location]

Operation:	addChargingZone(site, maxPowerZone)
Beschreibung:	Eine neue ChargingZone wird kreiert mit einer MaxPowerZone;
Vorbedingung:	Es müssen bereits alle Buchstaben des Alphabets bis hin zu diesem in der site bereits einer site in der Localisation zugeordnet sein
Nachbedingung:	eine site wird zugeordnet, eine MaxpowerZone wird erstellt; Die MaxPowerzone muss in einem gewissen Intervall bleiben
Ergebnisse:	eine ChargingZone mit Attributen wird erstellt
Ausnahmen:	-
Ausgaben:	-
Typ:	Systemoperation
Querverweise:	-
Anmerkungen:	Peter

Operation:	removeChargingZone(id)
Beschreibung:	entfernt die Ladezone der übergebene Id in Locations und deren Ladestationen
Vorbedingung:	es existiert mindestens eine Ladezone und deswegen eine Infrastruktur
Nachbedingung:	Ladezone mit id wurde entfernt und daher alle Ladestationen in der Ladezone
Ergebnisse:	Wahrheitswert ob entfernen erfolgreich ist
Ausnahmen:	Keine Ladezone der übergebenen id wurde gefunden
Ausgaben:	-
Typ:	Systemoperation
Querverweise:	Editing ein existierendes Szenario
Anmerkungen:	Thao

Operation:	editCharginZone(id, site)
Beschreibung:	Ladezone mit id wird geändert
Vorbedingung:	Es existiert mindestens eine Ladezone und daher eine Location und eine Infrastruktur
Nachbedingung:	Die Ladezone mit der der übergebenden Id wurde geändert
Ergebnisse:	Wahrheitswert ob ändern erfolgreich war
Ausnahmen:	Keine Ladezone mit der übergebenen Id wurde gefunden
Ausgaben:	-
Typ:	Systemoperation
Querverweise:	Editing ein existierendes Szenario
Anmerkungen:	Thao

[ChargingZone]

Operation:	addChargingStation(manufacturerg, name, maxPower)
Beschreibung:	fügt eine neue Ladestation mit Herstellername, Name, maxPower und verfügbare Stecker hinzu und fügt diese der Liste hinzu
Vorbedingung:	es existiert mindestens eine Ladezone und daher eine Location und eine Infrastruktur
Nachbedingung:	Ladezonenliste von Ladestationen haben einen neuen Eintrag -
Ergebnisse:	Wahrheitswert ob hinzufügen erfolgreich ist
Ausnahmen:	-
Ausgaben:	-
Typ:	Systemoperation
Querverweise:	Editing eine existierendes Szenario
Anmerkungen:	Thao

Operation:	removeCharingStation(id)
Beschreibung:	entfernt eine Ladestation mit der übergebenen id von einer Ladezone
Vorbedingung:	es existiert mindestens eine Ladezone und deswegen eine Location und Infrastruktur
Nachbedingung:	die Ladestation mit der übergebenen id wurde von der Ladezone entfernt und die maximale Power der Ladezonen wurde rekalkuliert
Ergebnisse:	-
Ausnahmen:	Keine Ladestation mit der übergebene id wurde gefunden
Ausgaben:	-
Typ:	Systemoperation
Querverweise:	Editing ein existierendes Szenario
Anmerkungen:	Thao

Operation:	editCharginStation(manufacturer, name, maxPower)
Beschreibung:	verändert eine existierende Ladestation der übergebenen Id
Vorbedingung:	es existert mindesetens eine Ladezone und deswegen eine Location und Infrastruktur
Nachbedingung:	-
Ergebnisse:	Wahrheitswert ob ändern erfolgreich war
Ausnahmen:	Keine Ladestation mit der übergebenen Id wurde gefunden
Ausgaben:	-
Typ:	Systemoperation
Querverweise:	Editing ein existierendes Szenario
Anmerkungen:	Thao

[ChargingStation]

Operation:	addPlug(type, power)
Beschreibung:	Fügt einen neuen Stecker zu einer Ladesäule hinzu
Vorbedingung:	Der Typ muss aus der Liste der gültigen Typen sein. Power muss größer 0 sein.
Nachbedingung:	Stecker wurde erfolgreich angelegt und ist von nun an unter einer bestimmten Id abrufbar
Ergebnisse:	Gibt zurück ob anlegen erfolgreich war
Ausnahmen:	Typ ist nicht korrekt
Ausgaben:	-
Typ:	Systemoperation
Querverweise:	-
Anmerkungen:	Thomas

Operation:	removePlug(id)
Beschreibung:	Entfernt einen bestimmten Stecker von einer Ladesäule anhand der ID
Vorbedingung:	Id muss gültig sein
Nachbedingung:	Stecker wurde erfolgreich von der Ladesäule entfernt
Ergebnisse:	Gibt Wahrheitswert zurück ob entfernen erfolgreich war
Ausnahmen:	Stecker mit der übergebenen Id wurde nicht gefunden
Ausgaben:	-
Typ:	Systemoperation
Querverweise:	-
Anmerkungen:	Thomas

[Simulation]

Operation:	addSecenario(duration, rushhours, spread, minRequests, maxRequests)
Beschreibung:	Erstellen von einem neuen Szenario in bestimmter Länge, mit der Angabe keiner, einer oder mehrerer Rushhour/s, einer Verteilung, minimaler und maximaler Anfragen
Vorbedingung:	-
Nachbedingung:	Szenario wurde erfolgreich mit allen erforderlichen Attributen angelegt und ist von nun an unter einer Bestimmten Id abrufbar
Ergebnisse:	Gibt Wahrheitswert zurück ob Szenario erfolgreich angelegt wurde
Ausnahmen:	-
Ausgaben:	Duration muss größer 0 sein; Rushhours, kann keine oder mehrere Zeitangaben beinhalten; Spread darf muss zwischen 0 und 1 liegen; minRequests muss kleiner gleich maxRequests sein und nicht negativ;
Typ:	Systemoperation
Querverweise:	-
Anmerkungen:	Thomas

Operation:	editScenario(id, completed, duration, rushours, spread, minRequests, maxRequests)
Beschreibung:	Ändern eines bestimmten Szenarios mit passenden Id; Die Länge des Szenarios, Rushhour/s, einer Verteilung, minimaler und maximaler Anfragen kann geändert werden
Vorbedingung:	Szenario mit entsprechender Id muss existieren
Nachbedingung:	Completed wird auf false gesetzt Szenario wurde erfolgreich mit allen erforderlichen Attributen geändert und ist von nun an unter der gleichen Id abrufbar
Ergebnisse:	Gibt Wahrheitswert zurück ob Szenario erfolgreich geändert wurde
Ausnahmen:	Szenario mit der entsprechenden Id wurde nicht gefunden
Ausgaben:	Duration muss größer Null sein; Rushhours, kann Null oder mehrere Zeitangaben beinhalten; Spread muss zwischen 0 und 1 liegen; minRequests muss kleiner gleich maxRequests sein und beides nicht negativ;
Typ:	Systemoperation
Querverweise:	-
Anmerkungen:	Thomas

Operation:	runScenario(id)
Beschreibung:	Es wir das entsprechende Szenario ausgeführt
Vorbedingung:	Es muss eine Location erstellt worden sein und mindestens ein Szenario muss der Simulation hinzugefügt worden sein.
Nachbedingung:	für jede Ladezone und Ladestation ist eine Auslastung pro tick Vorhanden, sowie eine Gesamtauslastung.
Ergebnisse:	Auslastung der Ladezonen und Ladestationen pro tick.
Ausnahme:	Keine Location/unvollständig Kein Szenario
Ausgaben:	-
Typ:	Systemoperation
Querverweise:	-
Anmerkung:	Tobi

Operation:	saveSzenarios()
Beschreibung:	Persistiert die Szenarien aus der Simulation
Vorbedingung:	Es müssen Szenarien in der Historie enthalten sein.
Nachbedingung:	Die Szenarien sind in einer JSON-Datei persistiert
Ergebnisse:	-
Ausnahme:	die Historie enthält keine Szenarien
Ausgaben:	-
Typ:	Systemoperation (Speicheroperation)
Querverweise:	-
Anmerkung:	Tobi

Operation:	loadSzenarios(JSON-Datei)
Beschreibung:	Liest eine JSON-Datei ein und lädt die Enthaltenen Szenarien. Die Szenarien werden der Historie hinzugefügt.
Vorbedingung:	Es müssen schon mal Szenarien gespeichert worden sein.
Nachbedingung:	Die Liste der Szenarien, die in der Historie enthalten sind, wurde um die Szenarien aus der JSON-Datei erweitert.
Ergebnisse:	-
Ausnahme:	Es gibt einen Fehler beim Einlesen der Szenarien
Ausgaben:	-
Typ:	Systemoperation (Speicheroperation)
Querverweise:	-
Anmerkung:	Tobi

Operation:	Simulation()
Beschreibung:	Konstruktor Ruft getInstance() auf
Vorbedingung:	-
Nachbedingung:	-
Ergebnisse:	-
Ausnahme:	-
Ausgaben:	-
Typ:	Systemoperation
Querverweise:	-
Anmerkung:	Thomas

Operation:	getInstance()
Beschreibung:	Erstellt ein neues Simulation Objekt
Vorbedingung:	Es darf kein Simulation Objekt existieren
Nachbedingung:	-
Ergebnisse:	Es wurde ein Simulation Objekt kreiert
Ausnahmen:	-
Ausgaben:	-
Typ:	Systemoperation
Querverweise:	-
Anmerkungen:	Thomas

[History]

Operation:	evaluateSzenario(id)
Beschreibung:	Die Auslastung der Ladezonen und Ladestationen die bei runSimulation() wird erstellt wird, wird ausgewertet und graphisch dargestellt
Vorbedingung:	runSimulation() wurde über das Szenario ausgeführt.
Nachbedingung:	-
Ergebnisse:	-
Ausnahme:	Übergebenes Szenario wurde evaluiert
Ausgaben:	die UI erhält die ausgewerteten Daten um sie zu veranschaulichen
Typ:	Systemoperation
Querverweise:	-
Anmerkungen:	Tobi

Operation:	Histroy()
Beschreibung:	Konstruktor Ruft getInstance() auf
Vorbedingung:	-
Nachbedingung:	-
Ergebnisse:	-
Ausnahme:	-
Ausgaben:	-
Typ:	Systemoperation
Querverweise:	-
Anmerkungen:	Thomas

Operation:	getInstance()
Beschreibung:	Erstellt ein neues History Objekt
Vorbedingung:	Es darf kein History Objekt existieren
Nachbedingung:	-
Ergebnisse:	Es wurde ein History Objekt kreiert
Ausnahmen:	-
Ausgaben:	-
Typ:	Systemoperation
Querverweise:	-
Anmerkungen:	Thomas