

---

# Smart Booking

# Administrations-Handbuch

1	Überblick .....	3
2	Clientseite – Struktureller Aufbau .....	4
3	SQL-Datenbank.....	6
3.1	ER-Modell .....	6
3.2	Relationales DB-Modell.....	6
3.3	Datenbank-Code.....	7
4	Security Konzept.....	17
5	Model – View – Controller .....	19
6	Exception-Handling .....	19
	Interfaces.....	20
6.1	ICommand .....	20
6.2	<i>ICommandResolver</i> .....	20
6.3	IDbConnection.....	20
6.4	IRequest.....	21
6.5	IResponse .....	21
6.6	ISecurityObject .....	22
7	Klassenbeschreibungen.....	22
7.1	DbConnection.....	22
7.2	FileSystemCommandResolver .....	23
7.3	FrontController.....	24
7.4	HttpRequest .....	24
7.5	HttpResponse .....	25
7.6	SecurityObject .....	26
7.7	TemplateView.....	27
8	Commands.....	28

8.1	Cmd_Standard.....	28
8.2	cmdFill_XY .....	29
8.3	cmdGet_XY.....	43
8.4	cmdHandle_XY .....	47
8.5	cmdInsert_XY.....	48
8.6	cmdLogout.....	56
8.7	cmdMail.....	57
8.8	cmdUpdate.....	58
9	Applikations-Register (Tabs).....	60
9.1	Anfragen .....	60
9.2	Auswertungen .....	86
9.3	Buchungen.....	92
9.4	Jugendherberge.....	110
9.5	Kunden .....	120
9.6	Leistungen .....	136
9.7	Package.....	144
9.8	Partner.....	155
9.9	User .....	164
10	Application.....	171
10.1	Javascript.....	171
10.2	CSS-Styles .....	179
10.3	Template.....	183
10.4	Serverpolling for Changes .....	184
11	Frontend .....	185
11.1	Javascript.....	185
11.2	CSS-Styles .....	190
11.3	Template.....	192
12	Login Form.....	192
12.1	Javascript.....	192
12.2	CSS-Styles .....	193
12.3	Template.....	193

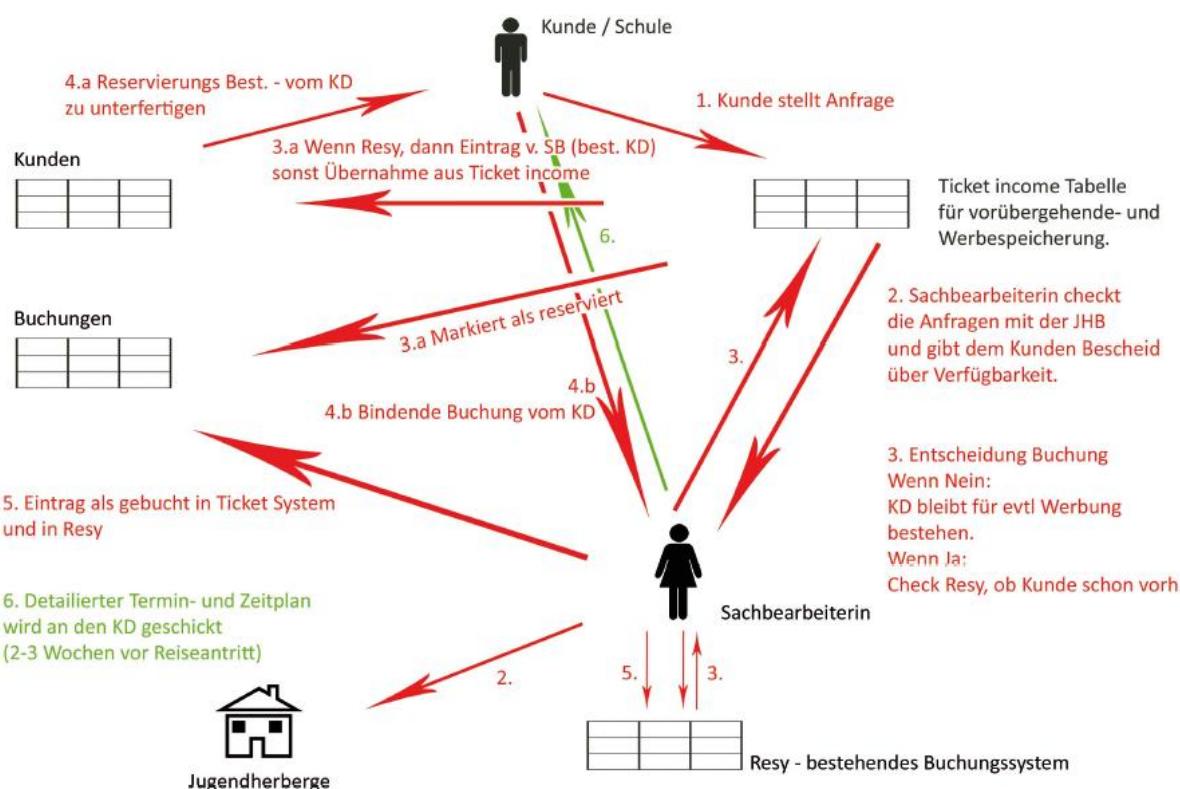
## 1 Überblick

Sehr geehrter Administrator,

wir benützen für die Darstellung unserer Applikation auf der Clientseite ein auf Javascript basierendes Framework namens ExtJS. Die Serverschickt wurde in PHP ausgeführt und wird von der Clientseite über asynchrone AJAX-Calls angesprochen. Die Kommunikation zwischen Server und Client erfolgt im JSON-Format. Gespeichert werden die Informatinen in einer mySQL 5.0 Datenbank.

Der Aufbau unserer Software folgt weitgehend dem Muster der objektorientierten Programmierung und unterliegt dem Paradigma des Model – View – Controller (MVC) Prinzips.

Hier sehen Sie eine Zusammenfassung des Prozessablaufes einer Buchung, begonnen von der Anfragestellung des Konsumenten über die Bezahlung bis zur Leistungskonsumierung, um die Prozesskette besser zu verstehen:



## 2 Clientseite – Struktureller Aufbau

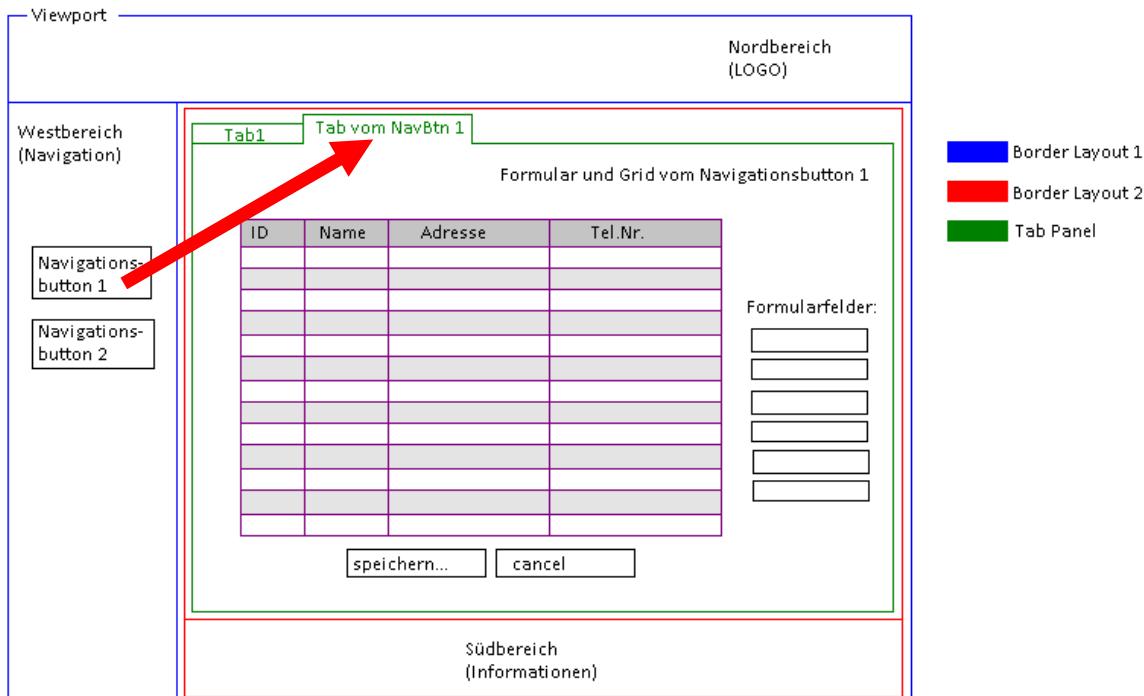
Der Viewport (das ist die Trägerkomponente des gesamten Dokumentes [document.body]) wird in ein Border-Layout unterteilt. Dieses Borderlayout beinhaltet einen Nordbereich, indem das Logo steht, einen Westbereich für die Navigation und natürlich einen Center-Bereich. Das Zentrum dieses Borderlayouts ist für sich betrachtet nochmals ein eigenes Border Layout. Diesmal mit einem Süd-Bereich für zusätzliche Infos und wieder einem Center-Bereich. In diesen Center-Bereich kommt ein TabPanel (eine Trägerkomponente für Registerkarten). Nach betätigen eines Buttons in der Navigation wird eine Funktion aufgerufen, die für den aufgerufenen Part alle notwendigen Stores und Panels generiert, die Steuerelemente zum Aufrufen der PHP-Commands bereitstellt und das Gesamtergebnis dieser Funktion in den TabPanel als neue Registerkarte einhängt. Nach dem Schließen einer Registerkarte wird der komplette Speicherbereich der Stores und Panels wieder freigegeben.

Hier ist der Aufbau des Gerüsts und der Navigation das Hauptaugenmerk. Die Steuerelemente, Grids und Stores der einzelnen Bereiche, die über die Navigationsbuttons annavigiert werden, finden Sie im jeweiligen "NAME DES MENÜPUNKTES.JS"-File



Durch drücken eines Navigationsbuttons wird eine Funktion aufgerufen, die die benötigten Stores, Grids, EventHandler, Steuerelemente und Formularelemente erzeugt und die Stores über den asynchronen Aufruf einer am Server gespeicherten Prozedur lädt. Diese erzeugten

Elemente werden in einem Panel optisch ansprechend dargestellt. Dieses Panel wird anschließend als neuer Tab in das oben dargestellte, grüne Haupt-Tab-Panel eingehängt.

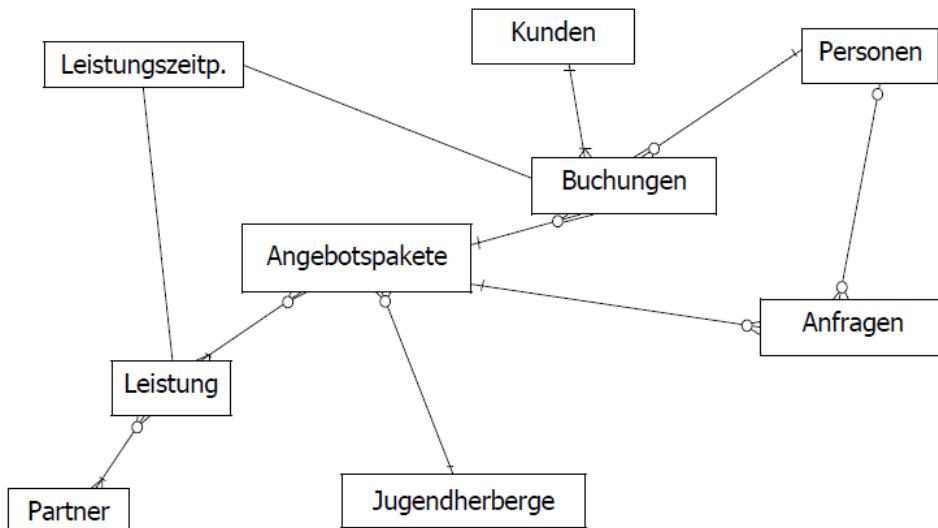


Zusätzlich muss erwähnt werden, dass die Navigationsbuttons von Beginn weg das Attribut „visible = false“ vordefiniert haben. Erst beim erfolgreichen Abfragen der Userrechte vom Server werden diese Buttons (je nach Rechten) auf „visible = true“ gesetzt. So steuern wir das Erscheinungsbild der Benutzeroberfläche, angepasst auf die Usergruppe. Natürlich ist ein sichtbar / unsichtbarsetzen von Buttons nicht unser gesamtes Security-Prinzip. Dieses wird später erläutert und befindet sich in der Serverschicht.

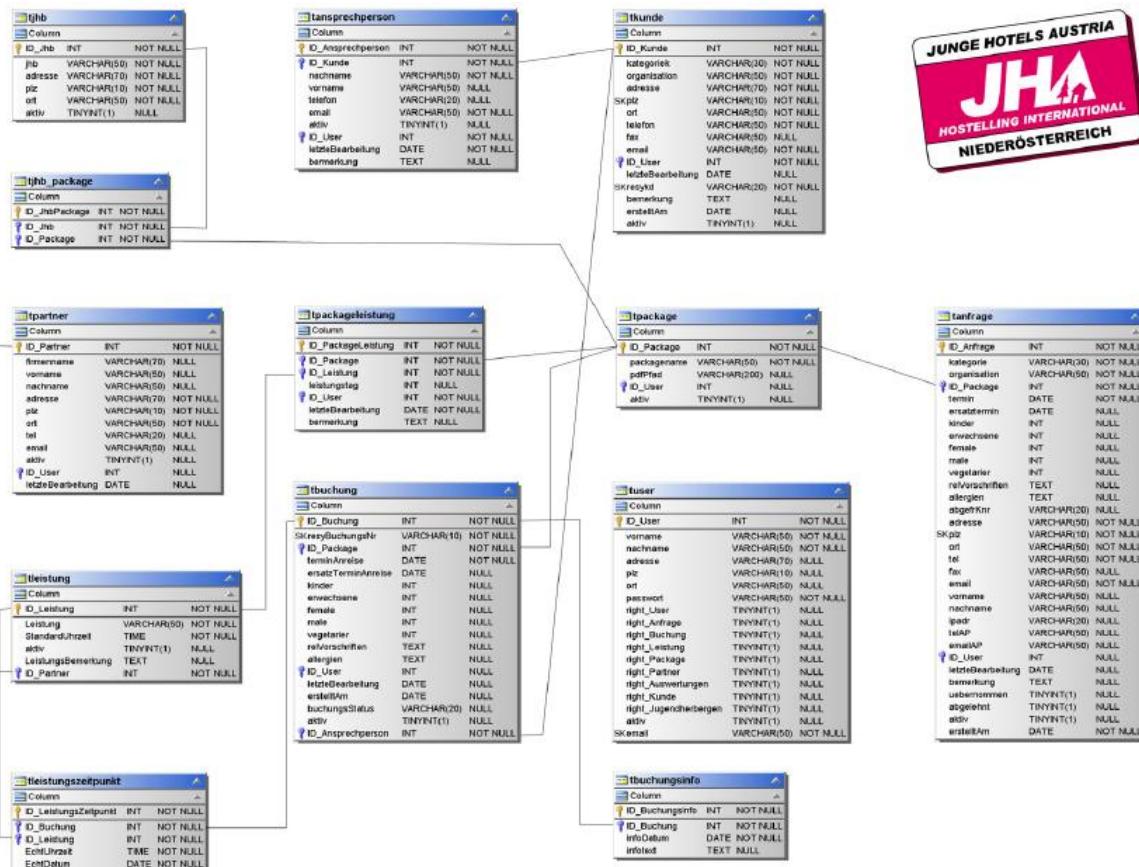
Der clientseitige Source Code (Javascript) für die Komponenten, die für den Aufbau relevant sind lautet wie folgt:

## 3 SQL-Datenbank

### 3.1 ER-Modell



### 3.2 Relationales DB-Modell



### 3.3 Datenbank-Code

```

/*****
***** CREATE DATABASE *****/
/***** USE `smartbooking`; */

/*****
***** CREATE TABLES *****/
/***** CREATE TABLE `tAnfrage` (
`ID_Anfrage` int(11) NOT NULL auto_increment,
`kategorie` varchar(30) collate latin1_general_ci NOT NULL,
`organisation` varchar(50) collate latin1_general_ci default NULL,
`ID_Package` int(11) NOT NULL,
`termin` date NOT NULL,
`ersatztermin` date default NULL,
`kinder` int(11) default NULL,
`erwachsene` int(11) default NULL,
`female` int(11) default NULL,
`male` int(11) default NULL,
`vegetarier` int(11) default NULL,
`relVorschriften` text collate latin1_general_ci,
`allergien` text collate latin1_general_ci,
`abgefrKnr` varchar(20) collate latin1_general_ci default NULL,
`adresse` varchar(50) collate latin1_general_ci NOT NULL,
`plz` varchar(10) collate latin1_general_ci NOT NULL,
`ort` varchar(50) collate latin1_general_ci NOT NULL,
`tel` varchar(50) collate latin1_general_ci default NULL,
`fax` varchar(50) collate latin1_general_ci default NULL,
`email` varchar(50) collate latin1_general_ci NULL,
`vorname` varchar(50) collate latin1_general_ci NOT NULL,
`nachname` varchar(50) collate latin1_general_ci NOT NULL,
`ipadr` varchar(20) collate latin1_general_ci default NULL,
`telAP` varchar(50) collate latin1_general_ci NOT NULL,
`emailAP` varchar(50) collate latin1_general_ci NOT NULL,
`ID_User` int(11) default NULL,
`letzteBearbeitung` date default NULL,
`bemerkung` text collate latin1_general_ci,
`uebernommen` tinyint(1) default '0',
`abgelehnt` tinyint(1) default '0',
`aktiv` tinyint(1) default '1',
`erstelltAm` date NOT NULL,
PRIMARY KEY (`ID_Anfrage`),
KEY `fk_tAnfrage_tPackage`(`ID_Package`),
KEY `IX_tAnfrage_organisation`(`organisation`),
KEY `IX_tAnfrage_nachname`(`nachname`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1_general_ci;

CREATE TABLE `tAnsprechperson` (
`ID_Ansprechperson` int(11) NOT NULL auto_increment,
`ID_Kunde` int(11) NOT NULL,
`nachname` varchar(50) collate latin1_general_ci NOT NULL,
`vorname` varchar(50) collate latin1_general_ci default NULL,
`telefon` varchar(20) collate latin1_general_ci default NULL,
`email` varchar(50) collate latin1_general_ci NOT NULL,
`aktiv` tinyint(1) default '1',
`ID_User` int(11) NOT NULL,
`letzteBearbeitung` date NOT NULL,
`bemerkung` text collate latin1_general_ci,
PRIMARY KEY (`ID_Ansprechperson`),
KEY `fk_tAnsprechperson_IDKunde`(`ID_Kunde`),

```

```
KEY `IX_AnspprechpersonNachname`(`nachname`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1_general_ci;
```

```
CREATE TABLE `tBuchung` (
`ID_Buchung` int(11) NOT NULL auto_increment,
`resyBuchungsNr` varchar(10) collate latin1_general_ci NOT NULL,
`ID_Package` int(11) NOT NULL,
`terminAnreise` date NOT NULL,
`ersatzTerminAnreise` date default NULL,
`kinder` int(11) default NULL,
`erwachsene` int(11) default NULL,
`female` int(11) default NULL,
`male` int(11) default NULL,
`vegetarier` int(11) default NULL,
`relVorschriften` text collate latin1_general_ci,
`allergien` text collate latin1_general_ci,
`ID_User` int(11) default NULL,
`letzteBearbeitung` date default NULL,
`erstelltAm` date default NULL,
`buchungsStatus` varchar(20) collate latin1_general_ci default NULL,
`aktiv` tinyint(1) default '1',
`ID_Anspprechperson` int(11) NOT NULL,
PRIMARY KEY (`ID_Buchung`),
KEY `IX_resyB`(`resyBuchungsNr`),
KEY `IX_status`(`buchungsStatus`),
KEY `FK_tbuchung_tansprechperson`(`ID_Anspprechperson`),
KEY `FK_tbuchung_tpakage`(`ID_Package`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1_general_ci;
```

```
CREATE TABLE `tBuchungsinfo` (
`ID_Buchungsinfo` int(11) NOT NULL auto_increment,
`ID_Buchung` int(11) NOT NULL,
`infoDatum` date NOT NULL,
`infotext` text collate latin1_general_ci,
`ID_User` int(11) NOT NULL,
PRIMARY KEY (`ID_Buchungsinfo`),
KEY `FK_tbuchungsinfo`(`ID_Buchung`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1_general_ci;
```

```
CREATE TABLE `tJhb` (
`ID_Jhb` int(11) NOT NULL auto_increment,
`jhB` varchar(50) collate latin1_general_ci NOT NULL,
`adresse` varchar(70) collate latin1_general_ci NOT NULL,
`plz` varchar(10) collate latin1_general_ci NOT NULL,
`ort` varchar(50) collate latin1_general_ci NOT NULL,
`aktiv` tinyint(1) default '1',
PRIMARY KEY (`ID_Jhb`),
KEY `IX_tJHB_jhb`(`jhB`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1_general_ci;
```

```
CREATE TABLE `tJhb_Package` (
`ID_Jhb` int(11) NOT NULL,
`ID_Package` int(11) NOT NULL,
PRIMARY KEY (`ID_Jhb`,`ID_Package`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1_general_ci;
```

```
CREATE TABLE `tKunde` (
`ID_Kunde` int(11) NOT NULL auto_increment,
`kategoriek` varchar(30) collate latin1_general_ci NOT NULL,
`organisation` varchar(50) collate latin1_general_ci NOT NULL,
`adresse` varchar(70) collate latin1_general_ci NOT NULL,
`plz` varchar(10) collate latin1_general_ci NOT NULL,
```

```

`ort` varchar(50) collate latin1_general_ci NOT NULL,
`telefon` varchar(50) collate latin1_general_ci NOT NULL,
`fax` varchar(50) collate latin1_general_ci default NULL,
`email` varchar(50) collate latin1_general_ci NOT NULL,
`ID_User` int(11) NOT NULL,
`letzteBearbeitung` date default NULL,
`resykd` varchar(20) collate latin1_general_ci NOT NULL,
`bemerkung` text collate latin1_general_ci,
`erstelltAm` date default NULL,
`aktiv` tinyint(1) default '1',
PRIMARY KEY (`ID_Kunde`),
KEY IX_resyKd(`resykd`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1_general_ci;

```

```

CREATE TABLE `tLeistung` (
`ID_Leistung` int(11) NOT NULL auto_increment,
`Leistung` varchar(50) collate latin1_general_ci NOT NULL,
`StandardUhrzeit` time NOT NULL,
`aktiv` tinyint(1) default '1',
`LeistungsBemerkung` text collate latin1_general_ci,
`ID_Partner` int(11) NOT NULL,
PRIMARY KEY (`ID_Leistung`),
KEY IX_tLeistung_Leistung(`Leistung`),
KEY FK_tleistung_tPartner(`ID_Partner`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1_general_ci;

```

```

CREATE TABLE `tLeistungsZeitpunkt` (
`ID_LeistungsZeitpunkt` int(11) NOT NULL auto_increment,
`ID_Buchung` int(11) NOT NULL,
`ID_Leistung` int(11) NOT NULL,
`EchtUhrzeit` time NOT NULL,
`EchtDatum` date NOT NULL,
PRIMARY KEY (`ID_LeistungsZeitpunkt`),
KEY FK_tLeistungsZeitpunkt_tBuchung(`ID_Buchung`),
KEY FK_tLeistungsZeitpunkt_tLeistung(`ID_Leistung`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1_general_ci;

```

```

CREATE TABLE `tPackage` (
`ID_Package` int(11) NOT NULL auto_increment,
`packagename` varchar(50) collate latin1_general_ci NOT NULL,
`pdfPfad` varchar(200) collate latin1_general_ci default NULL,
`ID_User` int(11) default NULL,
`letzteBearbeitung` date NOT NULL,
`aktiv` tinyint(1) default '1',
PRIMARY KEY (`ID_Package`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1_general_ci;

```

```

CREATE TABLE `tPackageleistung` (
`ID_Package` int(11) NOT NULL,
`ID_Leistung` int(11) NOT NULL,
`leistungstag` int(11) default NULL,
`ID_User` int(11) NOT NULL,
`letzteBearbeitung` date NOT NULL,
`bermerkung` text collate latin1_general_ci,
PRIMARY KEY (`ID_Package`,`ID_Leistung`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1_general_ci;

```

```

CREATE TABLE `tPartner` (
`ID_Partner` int(11) NOT NULL auto_increment,
`firmenname` varchar(70) collate latin1_general_ci default NULL,
`vorname` varchar(50) collate latin1_general_ci default NULL,

```

```

`nachname` varchar(50) collate latin1_general_ci default NULL,
`adresse` varchar(70) collate latin1_general_ci NOT NULL,
`plz` varchar(10) collate latin1_general_ci NOT NULL,
`ort` varchar(50) collate latin1_general_ci NOT NULL,
`tel` varchar(20) collate latin1_general_ci default NULL,
`email` varchar(50) collate latin1_general_ci default NULL,
`aktiv` tinyint(1) default '1',
`ID_User` int(11) default NULL,
`letzteBearbeitung` date default NULL,
PRIMARY KEY (`ID_Partner`),
KEY `IX_tPartner_firmenname`(`firmenname`),
KEY `IX_tPartner_nachname`(`nachname`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1_general_ci;

CREATE TABLE `tUser` (
`ID_User` int(11) NOT NULL auto_increment,
`vorname` varchar(50) collate latin1_general_ci NOT NULL,
`nachname` varchar(50) collate latin1_general_ci NOT NULL,
`email` varchar(50) collate latin1_general_ci NOT NULL,
`adresse` varchar(70) collate latin1_general_ci default NULL,
`plz` varchar(10) collate latin1_general_ci default NULL,
`ort` varchar(50) collate latin1_general_ci default NULL,
`passwort` varchar(50) collate latin1_general_ci NOT NULL,
`right_User` tinyint(1) default '1',
`right_Anfrage` tinyint(1) default '1',
`right_Buchung` tinyint(1) default '1',
`right_Leistung` tinyint(1) default '1',
`right_Package` tinyint(1) default '1',
`right_Partner` tinyint(1) default '1',
`right_Auswertungen` tinyint(1) default '1',
`right_Kunde` tinyint(1) default '1',
`right_Jugendherbergen` tinyint(1) default '1',
`aktiv` tinyint(1) default '1',
PRIMARY KEY (`ID_User`),
UNIQUE KEY `UK_eMail`(`email`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1_general_ci;

insert into
`tUser`(`ID_User`, `vorname`, `nachname`, `email`, `adresse`, `plz`, `ort`, `passwort`, `right_User`, `right_Anfrage`, `right_Buchung`, `right_Leistung`, `right_Package`, `right_Partner`, `right_Auswertungen`, `right_Kunde`, `right_Jugendherbergen`, `aktiv`) values
(1,'Michael','Wagner','michael.wagner@rockt.it',NULL,NULL,NULL,'e37f0136aa3ffaf149b351f6a4c948e9',1,1,1,1,1,1,1,1,1,1),(2,'Rolan d','Widmayer','widmayer.r@aon.at',NULL,NULL,NULL,'e37f0136aa3ffaf149b351f6a4c948e9',1,1,1,1,1,1,1,1,1,1);

/********************* ADD CONSTRAINTS TO TABLES *****/
ALTER TABLE tAnfrage ADD CONSTRAINT `fk_tAnfrage_tPackage` FOREIGN KEY (`ID_Package`) REFERENCES `tPackage`(`ID_Package`)
ON UPDATE CASCADE;
ALTER TABLE tAnsprechperson ADD CONSTRAINT `fk_tAnsprechperson_IDKunde` FOREIGN KEY (`ID_Kunde`) REFERENCES `tKunde`(`ID_Kunde`)
ON UPDATE CASCADE;
ALTER TABLE tBuchung ADD CONSTRAINT `FK_tbuchung_tpackage` FOREIGN KEY (`ID_Package`) REFERENCES `tPackage`(`ID_Package`)
ON UPDATE CASCADE;
ALTER TABLE tBuchung ADD CONSTRAINT `FK_tbuchung_tansprechperson` FOREIGN KEY (`ID_Ansprechperson`) REFERENCES
`tAnsprechperson`(`ID_Ansprechperson`)
ON UPDATE CASCADE;
ALTER TABLE tBuchungsinfo ADD CONSTRAINT `FK_tbuchungsinfo` FOREIGN KEY (`ID_Buchung`) REFERENCES `tBuchung`(`ID_Buchung`)
ON UPDATE CASCADE;
ALTER TABLE tJhb_Package ADD CONSTRAINT `FK_tJhb_tJhb_Package` FOREIGN KEY (`ID_Jhb`) REFERENCES `tJhb`(`ID_Jhb`)
ON UPDATE CASCADE;
ALTER TABLE tJhb_Package ADD CONSTRAINT `FK_tPackage_tJhb_Package` FOREIGN KEY (`ID_Package`) REFERENCES `tPackage`(`ID_Package`)
ON UPDATE CASCADE;
ALTER TABLE tLeistung ADD CONSTRAINT `FK_tleistung_tPartner` FOREIGN KEY (`ID_Partner`) REFERENCES `tPartner`(`ID_Partner`)
ON UPDATE CASCADE;

```

```

ALTER TABLE tLeistungszeitpunkt ADD CONSTRAINT `FK_tLeistungsZeitpunkt_tBuchung` FOREIGN KEY ( `ID_Buchung` ) REFERENCES
`tBuchung` ( `ID_Buchung` ) ON UPDATE CASCADE;
ALTER TABLE tLeistungszeitpunkt ADD CONSTRAINT `FK_tLeistungsZeitpunkt_tLeistung` FOREIGN KEY ( `ID_Leistung` ) REFERENCES
`tLeistung` ( `ID_Leistung` ) ON UPDATE CASCADE;
ALTER TABLE tPackageleistung ADD CONSTRAINT `FK_PackageLeistung_IDLeistung` FOREIGN KEY ( `ID_Leistung` ) REFERENCES
`tLeistung` ( `ID_Leistung` ) ON UPDATE CASCADE;
ALTER TABLE tPackageleistung ADD CONSTRAINT `FK_PackageLeistung_IDPackage` FOREIGN KEY ( `ID_Package` ) REFERENCES
`tPackage` ( `ID_Package` ) ON UPDATE CASCADE;

```

```

/********************* VIEWS *****/

```

```

DELIMITER $$

CREATE VIEW `vAnfrage` AS
SELECT
`tAnfrage`.`ID_Anfrage`      AS `ID_Anfrage`,
`tAnfrage`.`kategorie`       AS `kategorie`,
`tAnfrage`.`organisation`    AS `organisation`,
`tPackage`.`packagename`     AS `packagename`,
`tAnfrage`.`termin`          AS `termin`,
`tAnfrage`.`ersatztermin`    AS `ersatztermin`,
`tAnfrage`.`kinder`          AS `kinder`,
`tAnfrage`.`erwachsene`      AS `erwachsene`,
`tAnfrage`.`female`          AS `female`,
`tAnfrage`.`male`            AS `male`,
`tAnfrage`.`vegetarier`      AS `vegetarier`,
`tAnfrage`.`relVorschriften` AS `relVorschriften`,
`tAnfrage`.`allergien`       AS `allergien`,
`tAnfrage`.`abgefrKnr`        AS `abgefrKnr`,
`tAnfrage`.`adresse`         AS `adresse`,
`tAnfrage`.`plz`              AS `plz`,
`tAnfrage`.`ort`              AS `ort`,
`tAnfrage`.`tel`              AS `tel`,
`tAnfrage`.`fax`              AS `fax`,
`tAnfrage`.`email`            AS `email`,
`tAnfrage`.`vorname`          AS `vorname`,
`tAnfrage`.`nachname`         AS `nachname`,
`tAnfrage`.`ipadr`             AS `ipadr`,
`tAnfrage`.`telAP`             AS `telAP`,
`tAnfrage`.`emailAP`           AS `emailAP`,
`tAnfrage`.`uebernommen`      AS `uebernommen`,
`tAnfrage`.`abgelehnt`        AS `abgelehnt`,
CONCAT( `tUser`.`vorname` , '_latin1' , `tUser`.`nachname` ) AS `username`,
`tAnfrage`.`letzteBearbeitung` AS `letzteBearbeitung`,
`tAnfrage`.`bemerkung`        AS `bemerkung`,
`tAnfrage`.`erstelltAm`        AS `erstelltAm`
FROM (( `tAnfrage` LEFT JOIN `tUser` ON (( `tAnfrage`.`ID_User` = `tUser`.`ID_User` ))) 
LEFT JOIN `tPackage` ON (( `tPackage`.`ID_Package` = `tAnfrage`.`ID_Package` )))
WHERE ( `tAnfrage`.`aktiv` = 1 )$$
DELIMITER ;

```

```

/********************* STORED PROCEDURES *****/

```

```

DELIMITER $$

CREATE PROCEDURE `procInsertAnfrage` (
  IN kategorie varchar(30),
  IN organisation varchar(50),

```

```

IN ID_Package int,
IN termin date,
IN ersatztermin date,
IN kinder int,
IN erwachsene int,
IN female int,
IN male int,
IN vegetarier int,
IN relVorschriften text,
IN allergien text,
IN abgefrKnr varchar(20),
IN adresse varchar(50),
IN plz varchar(10),
IN ort varchar(50),
IN tel varchar(50),
IN fax varchar(50),
IN email varchar(50),
IN vorname varchar(50),
IN nachname varchar(50),
IN ipadr varchar(20),
IN telAP varchar(50),
IN emailAP varchar(50),
IN ID_User int,
IN letzteBearbeitung date,
IN bemerkung text,
IN erstelltAm date)
BEGIN
    INSERT INTO tAnfrage (kategorie, organisation, ID_Package, termin, ersatztermin, kinder,
    erwachsene, female, male, vegetarier, relVorschriften, allergien, abgefrKnr, adresse,
    plz, ort, tel, fax, email, vorname, nachname, ipadr, telAP, emailAP, ID_User, letzteBearbeitung,
    bemerkung, erstelltAm)
    VALUES (kategorie, organisation, ID_Package, termin, ersatztermin, kinder,
    erwachsene, female, male, vegetarier, relVorschriften, allergien, abgefrKnr, adresse,
    plz, ort, tel, fax, email, vorname, nachname, ipadr, telAP, emailAP, ID_User, letzteBearbeitung,
    bemerkung, erstelltAm);
END $$

DELIMITER ;

DELIMITER $$

CREATE PROCEDURE `procInsertPartner`(
    IN ID_Partner int,
    IN firmenname varchar(70),
    IN vorname varchar(50),
    IN nachname varchar(50),
    IN adresse varchar(70),
    IN plz varchar(10),
    IN ort varchar(50),
    IN tel varchar(20),
    IN email varchar(50),
    IN ID_User int,
    IN letzteBearbeitung date)
BEGIN
    INSERT INTO tPartner (ID_Partner, firmenname, vorname, nachname, adresse,
    plz, ort, tel, email, ID_User, letzteBearbeitung)
    VALUES (ID_Partner, firmenname, vorname, nachname, adresse,
    plz, ort, tel, email, ID_User, letzteBearbeitung);
END $$

DELIMITER ;

DELIMITER $$

CREATE PROCEDURE `procInsertJhb`(
    IN jhb varchar(50),
    IN adresse varchar(70),
    IN plz varchar(10),
    IN ort varchar(50))

```

```

BEGIN
    INSERT INTO tJhb (jhb, adresse, plz, ort, aktiv)
        VALUES (jhb, adresse, plz, ort, true);
END$$
DELIMITER ;

DELIMITER $$

CREATE PROCEDURE `procInsertJhbPack`(
    IN ID_Jhb int(11),
    IN ID_Package int(11))
BEGIN
    INSERT INTO tJhb_Package (ID_Jhb, ID_Package)
        VALUES (ID_Jhb, ID_Package);
END$$
DELIMITER ;

DELIMITER $$

CREATE PROCEDURE `procInsertPackage`(
    IN packagename varchar(50),
    IN pdfPfad varchar(200),
    IN ID_User int)
BEGIN
    INSERT INTO tPackage(packagename, pdfPfad, ID_User, aktiv)
        VALUES (packagename, pdfPfad, ID_User, true);
END$$
DELIMITER ;

DELIMITER $$

CREATE PROCEDURE `procInsertPackLeistung`(
    IN ID_Package int,
    IN ID_Leistung int)
BEGIN
    INSERT INTO tPackageleistung(ID_Package, ID_Leistung)
        VALUES (ID_Package, ID_Leistung);
END$$
DELIMITER ;

DELIMITER $$

CREATE PROCEDURE `procInsertKunde`(
    IN kategorie VARCHAR(30),
    IN organisation VARCHAR(50),
    IN adresse VARCHAR(50),
    IN plz VARCHAR(10),
    IN ort VARCHAR(50),
    IN telefon VARCHAR(50),
    IN fax VARCHAR(50),
    IN email VARCHAR(50),
    IN resykd VARCHAR(20),
    IN bemerkung TEXT,
    IN ID_User INT,
    IN letzteBearbeitung DATE,
    IN erstelltAm DATE)
BEGIN
    INSERT INTO tKunde (kategoriek, organisation,
        adresse, plz, ort, telefon, fax, email, ID_User, resykd, letzteBearbeitung,
        bemerkung, erstelltAm)
        VALUES (kategorie, organisation,
            adresse, plz, ort, telefon, fax, email, ID_User, resykd, letzteBearbeitung,
            bemerkung, erstelltAm);
END $$

DELIMITER ;

```

```

DELIMITER $$

CREATE PROCEDURE `procInsertAnsprechperson`(
    IN ID_Kunde INT,
    IN nachname VARCHAR(50),
    IN vorname VARCHAR(50),
    IN telefon VARCHAR(20),
    IN email VARCHAR(50),
    IN ID_User INT,
    IN letzteBearbeitung DATE,
    IN bemerkung TEXT)
BEGIN
    INSERT INTO tAnsprechperson (ID_Kunde,nachname,vorname,telefon,email,ID_User,letzteBearbeitung,bemerkung)
    VALUES (ID_Kunde,nachname,vorname,telefon,email,ID_User,letzteBearbeitung,bemerkung);
END $$

DELIMITER ;

DELIMITER $$

CREATE PROCEDURE `procInsertBuchung`(
    IN resyBuchungsNr VARCHAR(10),
    IN ID_Package int,
    IN terminAnreise DATE,
    IN ersatzTerminAnreise DATE,
    IN kinder INT,
    IN erwachsene INT,
    IN female INT,
    IN male INT,
    IN vegetarier INT,
    IN relVorschriften TEXT,
    IN allergien TEXT,
    IN ID_User INT,
    IN letzteBearbeitung DATE,
    IN erstelltAm DATE,
    IN buchungsStatus VARCHAR(20),
    IN ID_Anprechperson int)
BEGIN
    INSERT INTO tBuchung (resyBuchungsNr, ID_Package, terminAnreise, ersatzTerminAnreise, kinder, erwachsene, female, male, vegetarier, relVorschriften, allergien, ID_User, letzteBearbeitung, erstelltAm, buchungsStatus, ID_Anprechperson)
    VALUES (resyBuchungsNr, ID_Package, terminAnreise, ersatzTerminAnreise, kinder, erwachsene, female, male, vegetarier, relVorschriften, allergien, ID_User, letzteBearbeitung, erstelltAm, buchungsStatus, ID_Anprechperson);
END $$

DELIMITER ;

DELIMITER $$

CREATE PROCEDURE `procInsertLeistung` (
    IN Leistung VARCHAR(50),
    IN StandardUhrzeit TIME,
    IN LeistungsBemerkung TEXT,
    IN ID_Partner INT)
BEGIN
    INSERT INTO tLeistung (Leistung, StandardUhrzeit, LeistungsBemerkung, ID_Partner, aktiv)
    VALUES (Leistung, StandardUhrzeit, LeistungsBemerkung, ID_Partner, TRUE);
END$$

DELIMITER ;

DELIMITER $$

CREATE PROCEDURE `procInsertUser`(
    IN adresse VARCHAR(70),
    IN email VARCHAR(50),
    IN nachname VARCHAR(50),
    IN vorname VARCHAR(50),
    IN ort VARCHAR(50),
    IN plz VARCHAR(10),
    IN passwd VARCHAR(50))
BEGIN

```

```


INSERT INTO tUser (adresse, email, nachname, vorname, ort, plz, passwort, aktiv)
VALUES (adresse, email, nachname, vorname, ort, plz, passwd, TRUE);
END$$
DELIMITER ;

DELIMITER $$

CREATE PROCEDURE `procInsertLeistungsZeitpunkt`(
    IN ID_Buchung INT,
    IN ID_Leistung INT,
    IN zeit time,
    IN tag date)
BEGIN
    INSERT INTO tLeistungszeitpunkt (ID_Buchung, ID_Leistung, EchtUhrzeit, EchtDatum)
    VALUES (ID_Buchung, ID_Leistung, zeit, tag);
END$$
DELIMITER ;

DELIMITER $$

CREATE PROCEDURE `procInsertBuchungsHinweis`(
    IN ID_Buchung INT,
    IN infoDatum date,
    IN infotext TEXT,
    IN ID_User INT)
BEGIN
    INSERT INTO tBuchungsinfo (ID_Buchung, infodatum, infotext, ID_User)
    VALUES (ID_Buchung, infodatum, infotext, ID_User);
END$$
DELIMITER ;

/*
***** ANLEGEN VON TESTDATEN *****
***** ANLEGEN VON TESTDATEN *****
***** ANLEGEN VON TESTDATEN *****

insert into `tPackage`(`ID_Package`, `packagename`, `pdfPfad`, `ID_User`, `aktiv`, `letzteBearbeitung`)
values ('/packages/jugendaktive_annaberg.pdf',2,1,'2010-06-03'),(2,'Come Together Annaberg','/packages/cometogether_a.pdf',2,1,'2010-06-03'),(3,'Sportwoche Mini Annaberg','06-03'),(4,'Wasserratten Bad Grosspertholz','/package/wasserratten_bg.pdf',2,1,'2010-06-03'),(5,'Grosspertholz','/packages/creatmeyouth_bg.pdf',2,1,'2010-06-03'),(6,'Meine StÄfÄrke Drosendorf','/packages/meinestaerke_d.pdf',2,1,'2010-06-03'),(7,'Kommunikation und Team Lackenhoef','/packages/kom_team_l.pdf',2,1,'2010-06-03'),(8,'Vertrauen und sichern Lackenhoef','06-03'),(9,'Gruppendynamic Melk','/packages/gruppendymanik_m.pdf',2,1,'2010-06-03'),(10,'Melk','/pakckages/sozialeslernen_m.pdf',2,1,'2010-06-03'),(11,'We are Media Tulln','/packages/wearemedia_t.pdf',2,1,'2010-06-03')

insert into `tPartner`(`ID_Partner`, `firmenname`, `vorname`, `nachname`, `adresse`, `plz`, `ort`, `tel`, `email`, `values
(1,'Pirkers Lebzelterei','Monika','Schwaighofer','Hauptplatz ','3882','Mariazell','03882 2003'),(2,'Outdoor OG','Martin','Veith','Endergasse 57/5/5','1120','Wien','+43 650 9533551','office@ma03.at'),(3,'Reiterhof Schagi','Martin','Pfeffer','Langseitenrotte 21','3223','Wienerbruck','02728 3003'),(4,'Annabergerhaus','Veronika','Hinteregger','Annarotte ','3222','Annaberg','0664 4432000'),(5,'Tennisschule - Mariazellerland','Wolfgang','GlÄfÄnzl','M 22463','1,2,'2010-06-03'),(6,'Tennisschule - Mariazellerland','Wolfgang','GlÄfÄnzl','M 22463','1,2,'2010-06-03'),(7,'EIBL Lifte TÄfÄrnitz Ges.m.b.H.','Peter','Schackmann','Eiblstra. 8245','office@eibllife.at',1,2,'2010-06-03'),(8,'Mariazeller Schwebebahn','','','WienerstraÃ 2555','betriebsleitung@mariazell-buergeralpe.at',1,2,'2010-06-03'),(9,'Kameltheater Kernhof','1','3195','Kernhof','0664 1111012 ','kameltheater@aon.at',1,2,'2010-06-03'),(10,'Naturlehrpfad Annaberg','Helmuth','Widmayer','Annarotte 177','3222','Annaberg','02728 77045','widmayer. City St. PÄfÄrlten','','','SchieÃfÄstattring 15','3100','St. PÄfÄrlten','02742 3526610','1,2,'2010-03'),(11,'Heidemarie','Lammer','Hauptplatz ','8630','Mariazell','03882 2595505','1,2,'2010-03'),(12,'BetriebsfÄfÄhrungs-GmbH','','','Albrechtstr. 12','3950','GmÃfÄnd','02852/20 20 30'),(13,'Waldviertel Tourismus','Elisabeth','Ederndorfer','Sparkassenplatz 4','3910','Zwettl','+43 676 77 69 808','office@franztrischler.at',1,2,'2010-06-03'),(14,'info@waldviertel.at',1,2,'2010-06-03'),(15,'Lindhorn Kinderclub','Margarete','Patterer','+43 676 77 69 808','info@projektwochen.at',1,2,'2010-06-03'),(16,'Kletterhalle Obergrafendorf','DI Mag. Christoph ','Zarfli','Franz Werfelgasse 22','8045','Graz','+43 676 777 16 23','1,2,'2010-06-03'),(17,'M

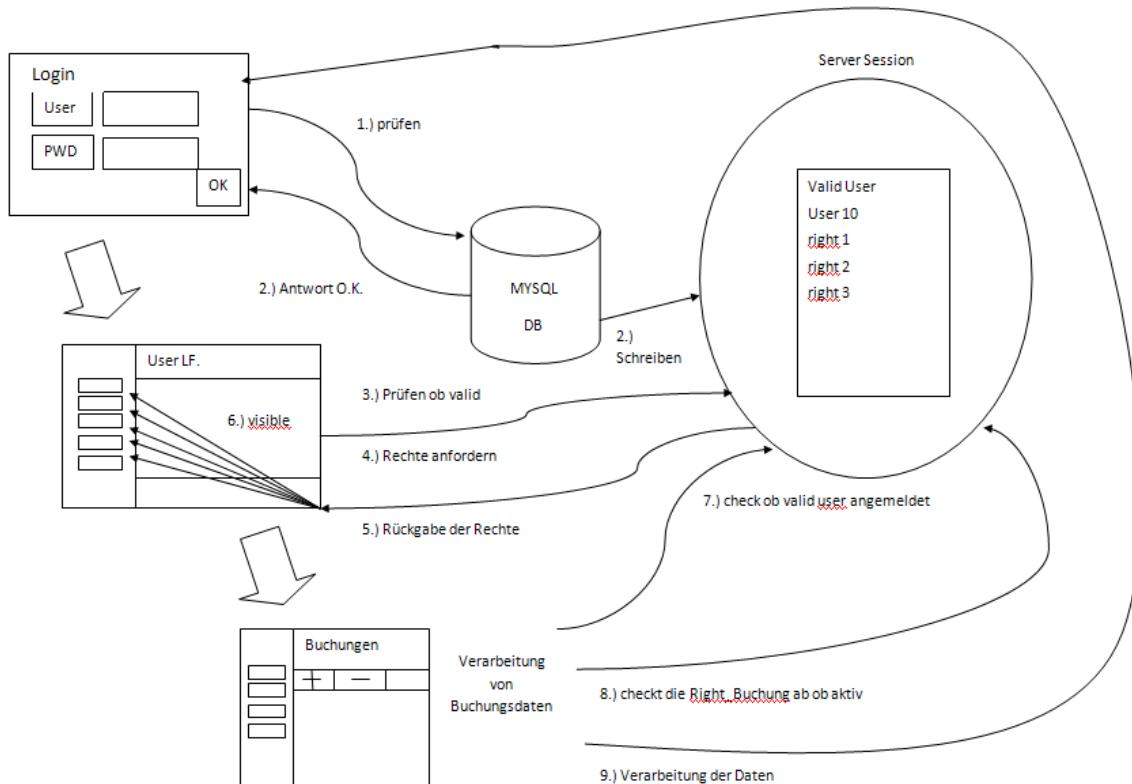

```

```

insert into `tLeistung`(`ID_Leistung`, `Leistung`, `StandardUhrzeit`, `aktiv`, `LeistungsBemerkung`, `ID_Partner`) values (1,'Lebkuchen verzieren','10:00:00',1,"1),(2,'Reiten','10:00:00',1,"3),(3,'Sommerrodelbahn','10:00:00',1,"7),(4,'High und Lowropes Elemente','09:00:00',1,"2),(5,'ProblemlÄfÄ¶lungen fÄfÄ¶r die Gruppe','13:00:00',1,"2),(6,'Lagerfeuer','20:00:00',1,"2),(7,'Angeleitete und spielerische Reflexion','09:00:00',1,"2),(8,'Schatzkammer','09:00:00',1,"12),(9,'Holzknechtland - BÄfÄ¶geralpe','13:00:00',1,"8),(10,'Kameltheater','11:00:00',1,"9),(11,'FÄfÄ¶hrung Naturlehrpfad','14:00:00',1,"10),(12,'Tennis','10:00:00',1,"6),(13,'Lagerfeuer','19:30:00',1,"5),(14,'Beachvolleyball','09:00:00',1,"2),(15,'Ruthsche,Schwimmen, Action','13:00:00',1,"13),(16,'Schwimmtraining','09:00:00',1,"13),(17,'Videoleabend','19:30:00',1,"5),(18,'Spieleabend','19:00:00',1,"5),(19,'Aqua Fitness','09:00:00',1,"13),(20,'Kinderakademie Waldviertel','13:00:00',1,"14),(21,'BÄfÄ¶hne frei','09:00:00',1,"14),(22,'Fassldorf','13:00:00',1,"14),(23,'Modellieren','09:30:00',1,"14),(24,'ProblemlÄfÄ¶lungsaufgaben','13:00:00',1,"2),(25,'High- und Lowropes Elemente','09:00:00',1,"2),(26,'Yoga und bioenergetische ÄfÄ¶bungen','13:30:00',1,"2),(27,'Outdoortag','09:00:00',1,"15),(28,'Fackelwanderung','19:30:00',1,"15),(29,'Vom niedrigen Seilelement bis hoch hinaus','13:30:00',1,"15),(30,'Spiel & SpaÄfÄ, reden ÄfÄ¶ber das','13:30:00',1,"15),(31,'Kommunikationsspiele','09:00:00',1,"15),(32,'Bouldern bis Toprpe','13:00:00',1,"15),(33,'First Aid Paket','09:00:00',1,"15),(34,'Gruppendynamische Aufgabenstellung','13:30:00',1,"15),(35,'Kletterschein','09:00:00',1,"15),(36,'Klettern','09:00:00',1,"16),(37,'Indoor Sky Walk','13:30:00',1,"16),(38,'Gruppendynamische Komm. Aufgabe','13:00:00',1,"16),(39,'Kegeln','09:00:00',1,"16),(40,'Hochseilgarten mit Flying Fox','09:00:00',1,"16),(41,'Klettern auf der Riesenleiter','13:30:00',1,"16),(42,'Trendsportfimmacht','20:00:00',1,"17),(43,'Workshops im Bereich Fotografie und Film','13:00:00',1,"17),(44,'Videoschnitt und Photoshop','09:00:00',1,"17),(45,'Lichtschipiele','16:00:00',1,"17),(46,'Fotoschau','09:00:00',1,"17),(47,'vb','00:00:00',1,'cvb',10);
insert into `tJhb`(`ID_Jhb`, `jhbc`, `adresse`, `plz`, `ort`, `aktiv`) values (1,'Annaberg','Annarotte 77','3222','Annaberg',1),(2,'Bad Grosspertholz',Nr. 177,'3972','Bad Grosspertholz',1),(3,'Drosendorf','BadstraÃfÄ,e 25','2095','Drosendorf',1),(4,'Lackenhof','ÄfÄétscherweg 3','3295','Lackenhof',1),(5,'Melk','Abt Karl StraÃfÄ,e 42','3390','Melk',1),(6,'Tulln','Marc Aurel Park 1','3430','Tulln',1);
insert into `tKunde`(`ID_Kunde`, `kategoriek`, `organisation`, `adresse`, `plz`, `ort`, `telefon`, `fax`, `email`, `ID_User`, `letzteBearbeitung`, `bemerkung`, `erstelltAm`, `aktiv`) values (1,'Schule','VS Altengbach','Schulgasse 3','3033','Altengbach','0664 73 66 99 02',"2,"2,"2010-06-03","0000370001258"),(2,'Schule','VS Altengbach','Schulgasse 3','3033','Altengbach','0664 73 66 99 02',"2,"2,"2010-06-03","0000370001258"),(3,'Schule','BG MÄfÄ¶dling','Untere Bachgasse 4','2340','MÄfÄ¶dling','02236'),(4,'Schule','VS Texing','Texing 19','3242','Texing',"2,"2,"2010-06-03","000036004141"),(5,'Schule','RG3','RadetzkystraÃfÄ,e 2A','1030','Wien',"2,"2,"2010-06-03","0000370001156"),(6,'Schule','BRG 4','Waltergasse 7','1040','Wien',"2,"2,"2010-06-03","000035002085"),(7,'Schule','PTS Wien West','SchopenhauerstraÃfÄ,e 81','1180','Wien',"2,"2,"2010-06-03","000037001161"),(8,'Schule','Sporthauptschule Scheibbs','Feldgasse 3','3270','Scheibbs',"2,"2,"2010-06-03","000000151893"),(9,'Schule','VS Altengbach','Schulgasse 3','3033','Altengbach','0664 73 66 99 02',"2,"2,"2010-06-03");
insert into `tAnsprechperson`(`ID_Ansprechperson`, `ID_Kunde`, `nachname`, `vorname`, `telefon`, `email`, `aktiv`, `ID_User`, `letzteBearbeitung`, `bemerkung`) values (1,1,'Neuhold','Maria','0664 73 66 99 02','annaberg@noejhw.at',1,2,"2010-06-03"),(2,3,'Zsak','Maria','02236','annaberg@noejhw.at',1,2,"2010-06-03"),(3,4,'Thier','Eva',027557256,'roland@widmayer.at'),(4,5,'Tzaferis','Christa','0699 11998286','roland@widmayer.at'),(5,6,'Michalek','Regine','01 5053447','roland@widmayer.at'),(6,7,'Greinsteiner','Martina','01 645 0127','roland@widmayer.at'),(7,8,'Rinner','Sandra','07482 42266,0','roland@widmayer.at'),(8,9,'Schule','VS Altengbach','Schulgasse 3','3033','Altengbach','0664 73 66 99 02',"2,"2,"2010-06-03");
insert into `tBuchung`(`ID_Buchung`, `resyBuchungsNr`, `ID_Package`, `terminAnreise`, `ersatzTerminAnreise`, `kinder`, `erwachsene`, `female`, `male`, `vegetarier`, `relVorschriften`, `allergien`, `ID_User`, `letzteBearbeitung`, `erstelltAm`, `buchungsStatus`, `aktiv`, `ID_Ansprechperson`) values (1,'1000370002',1,"2010-06-07"),(2,'1000370003',2,"2010-06-20"),(3,'1900-05-31',45,4,20,25,0,"2,"2,"2010-06-03"),(4,'1900-05-31',15,2,10,7,0,"2,"2,"2010-06-03"),(5,'1900-05-31',15,2,10,7,0,"2,"2,"2010-06-03"),(6,'1900-05-31',15,2,10,7,0,"2,"2,"2010-06-03"),(7,'1900-05-31',15,2,10,7,0,"2,"2,"2010-06-03"),(8,'1900-05-31',15,2,10,7,0,"2,"2,"2010-06-03"),(9,'1900-05-31',15,2,10,7,0,"2,"2,"2010-06-03"),(10,'1900-05-31',15,2,10,7,0,"2,"2,"2010-06-03");
insert into `tBuchungsinfo`(`ID_Buchungsinfo`, `ID_Buchung`, `infoDatum`, `infotext`, `ID_User`) values (1,3,"2010-06-05",'hfijfj'),1);
insert into `tAnfrage`(`ID_Anfrage`, `kategorie`, `organisation`, `ID_Package`, `termin`, `ersatztermin`, `kinder`, `erwachsene`, `female`, `male`, `vegetarier`, `relVorschriften`, `allergien`, `abgegrKnr`, `adresse`, `plz`, `ort`, `tel`, `fax`, `email`, `vorname`, `nachname`, `ipadr`, `telAP`, `emailAP`, `ID_User`, `letzteBearbeitung`, `bemerkung`, `uebernommen`, `abgelehnt`, `aktiv`, `erstelltAm`) values (22,'Schule','VS Altengbach',1,"2010-06-07"),(23,'Schule','BG MÄfÄ¶dling',2,"2028-06-20"),(24,'Schule','VS Wieberg / Rehberg',1,"2010-09-13"),(25,'Schule','VS Enzersdorf',1,"2010-09-15"),(26,'Schule','BAKIP 10',2,"2010-09-15"),(27,'Schule','VS Texing',2,"2010-10-06"),(28,'Schule','VS Enzersdorf',1,"2010-09-15"),(29,'Schule','VS Texing',2,"2010-10-06"),(30,'Schule','VS Texing',2,"2010-10-06"),(31,'Schule','VS Texing',2,"2010-10-06"),(32,'Schule','VS Texing',2,"2010-10-06"),(33,'Schule','VS Texing',2,"2010-10-06"),(34,'Schule','VS Texing',2,"2010-10-06"),(35,'Schule','VS Texing',2,"2010-10-06"),(36,'Schule','VS Texing',2,"2010-10-06"),(37,'Schule','VS Texing',2,"2010-10-06"),(38,'Schule','VS Texing',2,"2010-10-06"),(39,'Schule','VS Texing',2,"2010-10-06"),(40,'Schule','VS Texing',2,"2010-10-06"),(41,'Schule','VS Texing',2,"2010-10-06"),(42,'Schule','VS Texing',2,"2010-10-06"),(43,'Schule','VS Texing',2,"2010-10-06"),(44,'Schule','VS Texing',2,"2010-10-06"),(45,'Schule','VS Texing',2,"2010-10-06"),(46,'Schule','VS Texing',2,"2010-10-06"),(47,'Schule','VS Texing',2,"2010-10-06"),(48,'Schule','VS Texing',2,"2010-10-06"),(49,'Schule','VS Texing',2,"2010-10-06"),(50,'Schule','VS Texing',2,"2010-10-06"),(51,'Schule','VS Texing',2,"2010-10-06"),(52,'Schule','VS Texing',2,"2010-10-06"),(53,'Schule','VS Texing',2,"2010-10-06"),(54,'Schule','VS Texing',2,"2010-10-06"),(55,'Schule','VS Texing',2,"2010-10-06"),(56,'Schule','VS Texing',2,"2010-10-06"),(57,'Schule','VS Texing',2,"2010-10-06"),(58,'Schule','VS Texing',2,"2010-10-06"),(59,'Schule','VS Texing',2,"2010-10-06"),(60,'Schule','VS Texing',2,"2010-10-06"),(61,'Schule','VS Texing',2,"2010-10-06"),(62,'Schule','VS Texing',2,"2010-10-06"),(63,'Schule','VS Texing',2,"2010-10-06"),(64,'Schule','VS Texing',2,"2010-10-06"),(65,'Schule','VS Texing',2,"2010-10-06"),(66,'Schule','VS Texing',2,"2010-10-06"),(67,'Schule','VS Texing',2,"2010-10-06"),(68,'Schule','VS Texing',2,"2010-10-06"),(69,'Schule','VS Texing',2,"2010-10-06"),(70,'Schule','VS Texing',2,"2010-10-06"),(71,'Schule','VS Texing',2,"2010-10-06"),(72,'Schule','VS Texing',2,"2010-10-06"),(73,'Schule','VS Texing',2,"2010-10-06"),(74,'Schule','VS Texing',2,"2010-10-06"),(75,'Schule','VS Texing',2,"2010-10-06"),(76,'Schule','VS Texing',2,"2010-10-06"),(77,'Schule','VS Texing',2,"2010-10-06"),(78,'Schule','VS Texing',2,"2010-10-06"),(79,'Schule','VS Texing',2,"2010-10-06"),(80,'Schule','VS Texing',2,"2010-10-06"),(81,'Schule','VS Texing',2,"2010-10-06"),(82,'Schule','VS Texing',2,"2010-10-06"),(83,'Schule','VS Texing',2,"2010-10-06"),(84,'Schule','VS Texing',2,"2010-10-06"),(85,'Schule','VS Texing',2,"2010-10-06"),(86,'Schule','VS Texing',2,"2010-10-06"),(87,'Schule','VS Texing',2,"2010-10-06"),(88,'Schule','VS Texing',2,"2010-10-06"),(89,'Schule','VS Texing',2,"2010-10-06"),(90,'Schule','VS Texing',2,"2010-10-06"),(91,'Schule','VS Texing',2,"2010-10-06"),(92,'Schule','VS Texing',2,"2010-10-06"),(93,'Schule','VS Texing',2,"2010-10-06"),(94,'Schule','VS Texing',2,"2010-10-06"),(95,'Schule','VS Texing',2,"2010-10-06"),(96,'Schule','VS Texing',2,"2010-10-06"),(97,'Schule','VS Texing',2,"2010-10-06"),(98,'Schule','VS Texing',2,"2010-10-06"),(99,'Schule','VS Texing',2,"2010-10-06"),(100,'Schule','VS Texing',2,"2010-10-06"),(101,'Schule','VS Texing',2,"2010-10-06"),(102,'Schule','VS Texing',2,"2010-10-06"),(103,'Schule','VS Texing',2,"2010-10-06"),(104,'Schule','VS Texing',2,"2010-10-06"),(105,'Schule','VS Texing',2,"2010-10-06"),(106,'Schule','VS Texing',2,"2010-10-06"),(107,'Schule','VS Texing',2,"2010-10-06"),(108,'Schule','VS Texing',2,"2010-10-06"),(109,'Schule','VS Texing',2,"2010-10-06"),(110,'Schule','VS Texing',2,"2010-10-06"),(111,'Schule','VS Texing',2,"2010-10-06"),(112,'Schule','VS Texing',2,"2010-10-06"),(113,'Schule','VS Texing',2,"2010-10-06"),(114,'Schule','VS Texing',2,"2010-10-06"),(115,'Schule','VS Texing',2,"2010-10-06"),(116,'Schule','VS Texing',2,"2010-10-06"),(117,'Schule','VS Texing',2,"2010-10-06"),(118,'Schule','VS Texing',2,"2010-10-06"),(119,'Schule','VS Texing',2,"2010-10-06"),(120,'Schule','VS Texing',2,"2010-10-06"),(121,'Schule','VS Texing',2,"2010-10-06"),(122,'Schule','VS Texing',2,"2010-10-06"),(123,'Schule','VS Texing',2,"2010-10-06"),(124,'Schule','VS Texing',2,"2010-10-06"),(125,'Schule','VS Texing',2,"2010-10-06"),(126,'Schule','VS Texing',2,"2010-10-06"),(127,'Schule','VS Texing',2,"2010-10-06"),(128,'Schule','VS Texing',2,"2010-10-06"),(129,'Schule','VS Texing',2,"2010-10-06"),(130,'Schule','VS Texing',2,"2010-10-06"),(131,'Schule','VS Texing',2,"2010-10-06"),(132,'Schule','VS Texing',2,"2010-10-06"),(133,'Schule','VS Texing',2,"2010-10-06"),(134,'Schule','VS Texing',2,"2010-10-06"),(135,'Schule','VS Texing',2,"2010-10-06"),(136,'Schule','VS Texing',2,"2010-10-06"),(137,'Schule','VS Texing',2,"2010-10-06"),(138,'Schule','VS Texing',2,"2010-10-06"),(139,'Schule','VS Texing',2,"2010-10-06"),(140,'Schule','VS Texing',2,"2010-10-06"),(141,'Schule','VS Texing',2,"2010-10-06"),(142,'Schule','VS Texing',2,"2010-10-06"),(143,'Schule','VS Texing',2,"2010-10-06"),(144,'Schule','VS Texing',2,"2010-10-06"),(145,'Schule','VS Texing',2,"2010-10-06"),(146,'Schule','VS Texing',2,"2010-10-06"),(147,'Schule','VS Texing',2,"2010-10-06"),(148,'Schule','VS Texing',2,"2010-10-06"),(149,'Schule','VS Texing',2,"2010-10-06"),(150,'Schule','VS Texing',2,"2010-10-06"),(151,'Schule','VS Texing',2,"2010-10-06"),(152,'Schule','VS Texing',2,"2010-10-06"),(153,'Schule','VS Texing',2,"2010-10-06"),(154,'Schule','VS Texing',2,"2010-10-06"),(155,'Schule','VS Texing',2,"2010-10-06"),(156,'Schule','VS Texing',2,"2010-10-06"),(157,'Schule','VS Texing',2,"2010-10-06"),(158,'Schule','VS Texing',2,"2010-10-06"),(159,'Schule','VS Texing',2,"2010-10-06"),(160,'Schule','VS Texing',2,"2010-10-06"),(161,'Schule','VS Texing',2,"2010-10-06"),(162,'Schule','VS Texing',2,"2010-10-06"),(163,'Schule','VS Texing',2,"2010-10-06"),(164,'Schule','VS Texing',2,"2010-10-06"),(165,'Schule','VS Texing',2,"2010-10-06"),(166,'Schule','VS Texing',2,"2010-10-06"),(167,'Schule','VS Texing',2,"2010-10-06"),(168,'Schule','VS Texing',2,"2010-10-06"),(169,'Schule','VS Texing',2,"2010-10-06"),(170,'Schule','VS Texing',2,"2010-10-06"),(171,'Schule','VS Texing',2,"2010-10-06"),(172,'Schule','VS Texing',2,"2010-10-06"),(173,'Schule','VS Texing',2,"2010-10-06"),(174,'Schule','VS Texing',2,"2010-10-06"),(175,'Schule','VS Texing',2,"2010-10-06"),(176,'Schule','VS Texing',2,"2010-10-06"),(177,'Schule','VS Texing',2,"2010-10-06"),(178,'Schule','VS Texing',2,"2010-10-06"),(179,'Schule','VS Texing',2,"2010-10-06"),(180,'Schule','VS Texing',2,"2010-10-06"),(181,'Schule','VS Texing',2,"2010-10-06"),(182,'Schule','VS Texing',2,"2010-10-06"),(183,'Schule','VS Texing',2,"2010-10-06"),(184,'Schule','VS Texing',2,"2010-10-06"),(185,'Schule','VS Texing',2,"2010-10-06"),(186,'Schule','VS Texing',2,"2010-10-06"),(187,'Schule','VS Texing',2,"2010-10-06"),(188,'Schule','VS Texing',2,"2010-10-06"),(189,'Schule','VS Texing',2,"2010-10-06"),(190,'Schule','VS Texing',2,"2010-10-06"),(191,'Schule','VS Texing',2,"2010-10-06"),(192,'Schule','VS Texing',2,"2010-10-06"),(193,'Schule','VS Texing',2,"2010-10-06"),(194,'Schule','VS Texing',2,"2010-10-06"),(195,'Schule','VS Texing',2,"2010-10-06"),(196,'Schule','VS Texing',2,"2010-10-06"),(197,'Schule','VS Texing',2,"2010-10-06"),(198,'Schule','VS Texing',2,"2010-10-06"),(199,'Schule','VS Texing',2,"2010-10-06"),(200,'Schule','VS Texing',2,"2010-10-06"),(201,'Schule','VS Texing',2,"2010-10-06"),(202,'Schule','VS Texing',2,"2010-10-06"),(203,'Schule','VS Texing',2,"2010-10-06"),(204,'Schule','VS Texing',2,"2010-10-06"),(205,'Schule','VS Texing',2,"2010-10-06"),(206,'Schule','VS Texing',2,"2010-10-06"),(207,'Schule','VS Texing',2,"2010-10-06"),(208,'Schule','VS Texing',2,"2010-10-06"),(209,'Schule','VS Texing',2,"2010-10-06"),(210,'Schule','VS Texing',2,"2010-10-06"),(211,'Schule','VS Texing',2,"2010-10-06"),(212,'Schule','VS Texing',2,"2010-10-06"),(213,'Schule','VS Texing',2,"2010-10-06"),(214,'Schule','VS Texing',2,"2010-10-06"),(215,'Schule','VS Texing',2,"2010-10-06"),(216,'Schule','VS Texing',2,"2010-10-06"),(217,'Schule','VS Texing',2,"2010-10-06"),(218,'Schule','VS Texing',2,"2010-10-06"),(219,'Schule','VS Texing',2,"2010-10-06"),(220,'Schule','VS Texing',2,"2010-10-06"),(221,'Schule','VS Texing',2,"2010-10-06"),(222,'Schule','VS Texing',2,"2010-10-06"),(223,'Schule','VS Texing',2,"2010-10-06"),(224,'Schule','VS Texing',2,"2010-10-06"),(225,'Schule','VS Texing',2,"2010-10-06"),(226,'Schule','VS Texing',2,"2010-10-06"),(227,'Schule','VS Texing',2,"2010-10-06"),(228,'Schule','VS Texing',2,"2010-10-06"),(229,'Schule','VS Texing',2,"2010-10-06"),(230,'Schule','VS Texing',2,"2010-10-06"),(231,'Schule','VS Texing',2,"2010-10-06"),(232,'Schule','VS Texing',2,"2010-10-06"),(233,'Schule','VS Texing',2,"2010-10-06"),(234,'Schule','VS Texing',2,"2010-10-06"),(235,'Schule','VS Texing',2,"2010-10-06"),(236,'Schule','VS Texing',2,"2010-10-06"),(237,'Schule','VS Texing',2,"2010-10-06"),(238,'Schule','VS Texing',2,"2010-10-06"),(239,'Schule','VS Texing',2,"2010-10-06"),(240,'Schule','VS Texing',2,"2010-10-06"),(241,'Schule','VS Texing',2,"2010-10-06"),(242,'Schule','VS Texing',2,"2010-10-06"),(243,'Schule','VS Texing',2,"2010-10-06"),(244,'Schule','VS Texing',2,"2010-10-06"),(245,'Schule','VS Texing',2,"2010-10-06"),(246,'Schule','VS Texing',2,"2010-10-06"),(247,'Schule','VS Texing',2,"2010-10-06"),(248,'Schule','VS Texing',2,"2010-10-06"),(249,'Schule','VS Texing',2,"2010-10-06"),(250,'Schule','VS Texing',2,"2010-10-06"),(251,'Schule','VS Texing',2,"2010-10-06"),(252,'Schule','VS Texing',2,"2010-10-06"),(253,'Schule','VS Texing',2,"2010-10-06"),(254,'Schule','VS Texing',2,"2010-10-06"),(255,'Schule','VS Texing',2,"2010-10-06"),(256,'Schule','VS Texing',2,"2010-10-06"),(257,'Schule','VS Texing',2,"2010-10-06"),(258,'Schule','VS Texing',2,"2010-10-06"),(259,'Schule','VS Texing',2,"2010-10-06"),(260,'Schule','VS Texing',2,"2010-10-06"),(261,'Schule','VS Texing',2,"2010-10-06"),(262,'Schule','VS Texing',2,"2010-10-06"),(263,'Schule','VS Texing',2,"2010-10-06"),(264,'Schule','VS Texing',2,"2010-10-06"),(265,'Schule','VS Texing',2,"2010-10-06"),(266,'Schule','VS Texing',2,"2010-10-06"),(267,'Schule','VS Texing',2,"2010-10-06"),(268,'Schule','VS Texing',2,"2010-10-06"),(269,'Schule','VS Texing',2,"2010-10-06"),(270,'Schule','VS Texing',2,"2010-10-06"),(271,'Schule','VS Texing',2,"2010-10-06"),(272,'Schule','VS Texing',2,"2010-10-06"),(273,'Schule','VS Texing',2,"2010-10-06"),(274,'Schule','VS Texing',2,"2010-10-06"),(275,'Schule','VS Texing',2,"2010-10-06"),(276,'Schule','VS Texing',2,"2010-10-06"),(277,'Schule','VS Texing',2,"2010-10-06"),(278,'Schule','VS Texing',2,"2010-10-06"),(279,'Schule','VS Texing',2,"2010-10-06"),(280,'Schule','VS Texing',2,"2010-10-06"),(281,'Schule','VS Texing',2,"2010-10-06"),(282,'Schule','VS Texing',2,"2010-10-06"),(283,'Schule','VS Texing',2,"2010-10-06"),(284,'Schule','VS Texing',2,"2010-10-06"),(285,'Schule','VS Texing',2,"2010-10-06"),(286,'Schule','VS Texing',2,"2010-10-06"),(287,'Schule','VS Texing',2,"2010-10-06"),(288,'Schule','VS Texing',2,"2010-10-06"),(289,'Schule','VS Texing',2,"2010-10-06"),(290,'Schule','VS Texing',2,"2010-10-06"),(291,'Schule','VS Texing',2,"2010-10-06"),(292,'Schule','VS Texing',2,"2010-10-06"),(293,'Schule','VS Texing',2,"2010-10-06"),(294,'Schule','VS Texing',2,"2010-10-06"),(295,'Schule','VS Texing',2,"2010-10-06"),(296,'Schule','VS Texing',2,"2010-10-06"),(297,'Schule','VS Texing',2,"2010-10-06"),(298,'Schule','VS Texing',2,"2010-10-06"),(299,'Schule','VS Texing',2,"2010-10-06"),(300,'Schule','VS Texing',2,"2010-10-06"),(301,'Schule','VS Texing',2,"2010-10-06"),(302,'Schule','VS Texing',2,"2010-10-06"),(303,'Schule','VS Texing',2,"2010-10-06"),(304,'Schule','VS Texing',2,"2010-10-06"),(305,'Schule','VS Texing',2,"2010-10-06"),(306,'Schule','VS Texing',2,"2010-10-06"),(307,'Schule','VS Texing',2,"2010-10-06"),(308,'Schule','VS Texing',2,"2010-10-06"),(309,'Schule','VS Texing',2,"2010-10-06"),(310,'Schule','VS Texing',2,"2010-10-06"),(311,'Schule','VS Texing',2,"2010-10-06"),(312,'Schule','VS Texing',2,"2010-10-06"),(313,'Schule','VS Texing',2,"2010-10-06"),(314,'Schule','VS Texing',2,"2010-10-06"),(315,'Schule','VS Texing',2,"2010-10-06"),(316,'Schule','VS Texing',2,"2010-10-06"),(317,'Schule','VS Texing',2,"2010-10-06"),(318,'Schule','VS Texing',2,"2010-10-06"),(319,'Schule','VS Texing',2,"2010-10-06"),(320,'Schule','VS Texing',2,"2010-10-06"),(321,'Schule','VS Texing',2,"2010-10-06"),(322,'Schule','VS Texing',2,"2010-10-06"),(323,'Schule','VS Texing',2,"2010-10-06"),(324,'Schule','VS Texing',2,"2010-10-06"),(325,'Schule','VS Texing',2,"2010-10-06"),(326,'Schule','VS Texing',2,"2010-10-06"),(327,'Schule','VS Texing',2,"2010-10-06"),(328,'Schule','VS Texing',2,"2010-10-06"),(329,'Schule','VS Texing',2,"2010-10-06"),(330,'Schule','VS Texing',2,"2010-10-06"),(331,'Schule','VS Texing',2,"2010-10-06"),(332,'Schule','VS Texing',2,"2010-10-06"),(333,'Schule','VS Texing',2,"2010-10-06"),(334,'Schule','VS Texing',2,"2010-10-06"),(335,'Schule','VS Texing',2,"2010-10-06"),(336,'Schule','VS Texing',2,"2010-10-06"),(337,'Schule','VS Texing',2,"2010-10-06"),(338,'Schule','VS Texing',2,"2010-10-06"),(339,'Schule','VS Texing',2,"2010-10-06"),(340,'Schule','VS Texing',2,"2010-10-06"),(341,'Schule','VS Texing',2,"2010-10-06"),(342,'Schule','VS Texing',2,"2010-10-06"),(343,'Schule','VS Texing',2,"2010-10-06"),(344,'Schule','VS Texing',2,"2010-10-06"),(345,'Schule','VS Texing',2,"2010-10-06"),(346,'Schule','VS Texing',2,"2010-
```

## 4 Security Konzept

Das Security-Konzept von Smartbooking ist im Folgenden kurz schematisch dargestellt:

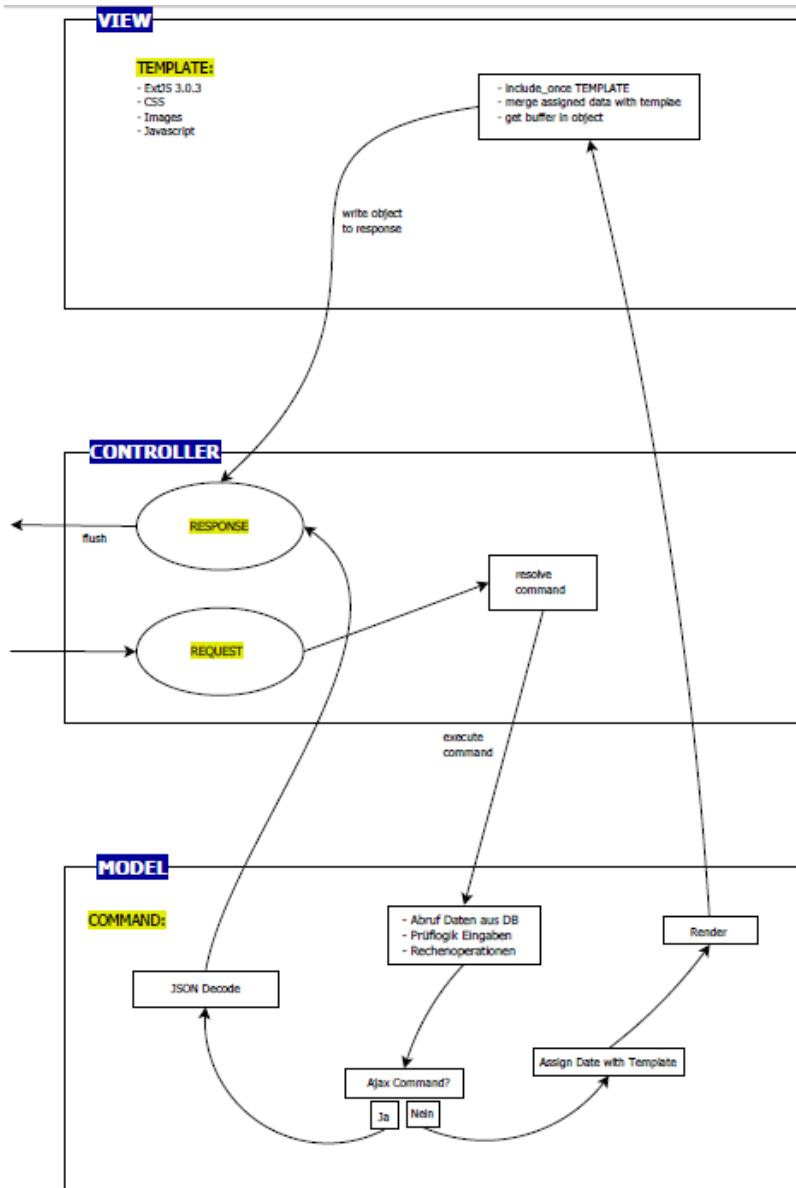


Beim ersten Kontakt mit dem User wird ein Login-Fenster dargestellt und eine Benutzerauthentifizierung wird durchgeführt. Sobald der User seine Credentials eingegeben hat, werden das Passwort und die Mailadresse mit den gespeicherten Daten in der Datenbank kontrolliert. Das Passwort ist md5 codiert und wird erst am Server umgewandelt. Das bedeutet eine Übertragung des Passwortes vom Client zum Server in Klartext. Zur Lösung dieses Problems ist eine HTTPS-Verschlüsselung vorgesehen. Nachdem die Daten erfolgreich mit der Datenbank abgeglichen wurden, werden die entsprechenden Rechte des Users in die Serversession gespeichert. Danach wird der User zur eigentlichen Applikation weitergeleitet. Beim Aufbau der Applikation werden die Rechte, die in der Session gespeichert sind, abgefragt. Je nach Berechtigung werden die Navigationsbuttons auf „visible = true“ gesetzt und somit wird das Interface an die Berechtigungen des Users angepasst. Bei jedem weiteren Speichervorgang am Server, sowie beim Abrufen von Daten wird zuerst in der Session kontrolliert, ob ein valider User angemeldet ist. Danach wird geprüft, ob dieser User die entsprechenden Berechtigungen für die angeforderte Datenaktion besitzt. Falls das alles zutrifft, werden die Daten verarbeitet.

Programmiertechnisch wurde die angestrebte Lösung durch definieren einer Klasse „SecurityObject“ erreicht, in dessen Constructor-Methode die Prüfung auf den angemeldeten User in der Session durchgeführt wird. Es existiert in dieser Klasse auch noch eine Methode, mit der ein Rechtevergleich durchgeführt werden kann.

## 5 Model – View – Controller

Smartbooking beruht auf dem Model-View-Controller Prinzip. Das bedeutet, dass die Logikebene strikt von der Präsentationsebene getrennt ist. Eine bildliche Darstellung dieser Logik im speziellen Fall sieht wie folgt aus:



## 6 Exception-Handling

Smartbooking besitzt ein komplexes und ausgereiftes Exception-Handling. Es folgt der Theorie, dass Fehler wie Blasen im Programm aufsteigen und in einer ansprechenden und verständlichen Form an den Benutzer gemeldet werden, ohne das Programm abstürzen zu lassen. Sollte die JS-Applikation bereits laufen, erkennt der Server diesen Umstand und sendet die Exception-Meldung im JSON-Format an den Client. Dort wird die Exception

erkannt und über die Funktion HandleException in geeigneter Form an den Benutzer weitergegeben. Sollte die Applikation noch nicht am Client laufen und es tritt ein Fehler am Server auf, wird die Fehlermeldung auf die gleiche Art und Weise erzeugt, aber nicht per JSON weitergegeben. Stattdessen wird ein eigenes View „Fehler“ mit dem Fehlertext aufgerufen und an den Client als Response zurückgegeben. Serverseitig findet sich das Exceptionhandling (also der try-catch Block) in der index.php, dem zentralen Einstiegspunkt in die Anwendung. Geworfen werden die Exceptions dort, wo sie auftreten (also z.B. beim Verbinden zum DB-Server, oder beim Verbinden zum SMTP-Server, oder beim Absetzen eines SQL-Statements). Aus Gründen der Zentralisierung wird in der Klasse DBConnection nach jedem versuchten Absetzen eines Statements auf Fehler geprüft und im Fehlerfall eine Exception geworfen.

## 7 Interfaces

Um die Wiederverwendbarkeit des Codes zu erhöhen, und um die Abhängigkeiten zwischen den Codebausteinen zu verkleinern, wurden die wichtigsten Klassen als Interfaces beschrieben. Das bringt einige erhebliche Vorteile mit sich.

### 7.1 ICommand

```
<?php
//Leitidee: Jede Seite im Webspace wird zu einem Command
//Ausgenommen davon ist eine einzige Seite - der zentrale Einstiegspunkt
// Unter PHP am Apache Webserver ist der typische Name dieser Seite index.php

//Das Commandinterface spezifiziert die Anforderung an Klassen,
//die für Seiteninhalte zuständig sind.

//Vorteil:
// - Neue Commandklassen können mit geringer Abhängigkeit hinzugefügt werden
// - Bestehende Commandklassen können ohne große Tiefen- und Breitenwirkung
// ausgetauscht werden.

interface ICommand {
    public function execute(IRequest $request, IResponse $response);
}

?>
```

### 7.2 ICommandResolver

```
<?php

interface ICommandResolver {
    public function getCommand(IRequest $request);
}

?>
```

### 7.3 IDbConnection

```
<?php
interface IDbConnection {

    public function affected_rows();
```

```

public function query($sql);

public function real_escape_string($value);
}

?>

```

## 7.4 IRequest

```

<?php
//die Interfacebeschreibung für den Request soll bzw. muss so gehalten sein,
// dass sie nicht nur den HTTP-Request abdeckt, sondern auch z.B. einen XML-Request.

//Die Festlegung von Interfaces erfordert einen Weitblick bezüglich der möglichen
// Anwendungsbreite einer Anwendung, sowie technische Detailkenntnisse bezüglich notwendiger Methoden
// im Interface.

//Alle Kommentare im Interface-Abschnitt beziehen sich nur auf die Verwendung beim HTTP-Request.

interface IRequest {

    //Liefere alle Keys des $_POST Arrays zurück
    public function getParameterNames();

    //Prüfe, ob es diesen Key ($name) am $_POST Array gibt
    public function issetParameter($name);

    //Entferne einen Eintrag im $_POST Array. Der Eintrag hat den Key $name.
    public function unsetParameter($name);

    //holt den Wert zum Key ($name) aus dem $_POST Array
    public function getParameter($name);

    //Liefere einen Eintrag aus dem HTTP-Header.
    // $name ist ein Key aus $_SERVER
    public function getHeader($name);
}

?>

```

## 7.5 IResponse

```

<?php
//Warum wird eine Schnittstelle zum Senden von Daten an den Client verwendet?
//Ist nicht die Verwendung von print oder echo wesentlich einfacher?
// - Vorteil
//   - Die Daten können vor dem Versenden zentralisiert verändert werden
//     Beispiel: Anstelle von Antwort an den Browser wird eine XML-Datei an ein
//               Webservice geschickt.

interface IResponse {

    //Mit setStatus werden im Header-Teil des HTTP-Protokolls Statusmeldungen platziert
    //z.B. 200 ok, 404 not found
    public function setStatus($status);

    //Mit addHeader werden die Statusangaben tatsächlich in den Protokollheader eingefügt
    public function addHeader($name, $value);

    //Die an den Client zu sendenden Daten werden als Objekteigenschaftswert zwischengespeichert
    //die write-Methode beschifft diesen Eigenschaftswert.
    public function write($data);

    //die flush-Methode macht mit dem Abschicken ernst. Mit flush wird der über write() angesammelte Eigenschafts-
    //wert und alle Header abgeschickt.
}

```

```

    public function flush();
}

?>

```

## 7.6 ISecurityObject

```

<?php

interface ISecurityObject {

    public function checkCredentials(IDbConnection $db, IRequest $request);
}
?>

```

# 8 Klassenbeschreibungen

Die Programmierung dieser Software folgt dem Paradigma der „Objektorientierten Programmierung“.

## 8.1 DbConnection

```

<?php

include_once("Interfaces".DIRECTORY_SEPARATOR."IDbConnection.php");

class DbConnection implements IDbConnection {
    private $link=null;

    public function __construct() {
        //Verbindung zum Datenbank-Server herstellen (Connection String)
        $this->link = @ new mysqli('127.0.0.1', 'noejhw', 'noe21', '3306');
        //@ unterdrückt die Fehlermeldung, wenn Connection nicht erstellt werden kann.
        //{$this->link->connect_error erst ab PHP 5.2.9 unterstützt
        //daher hier die winzige Ausnahme:
        if(mysqli_connect_error()) {
            throw new DBConnectException('Konnte nicht zum Datenbankserver verbinden. Bitte prüfen Sie die Erreichbarkeit des
Selbigen.');
        }
        //Datenbank am Server auswählen
        $this->link->select_db('noejhw');
        if($this->link->errno) {
            throw new DBConnectException('Datenbank konnte nicht ausgewählt werden. Bitte prüfen Sie, ob die Datenbank
existiert.');
        }
    }

    public function affected_rows() {
        return $this->link->affected_rows;
    }

    public function query($sql) {
        $query_result = $this->link->query($sql);
        if($this->link->errno) {
            throw new DBSQLException('Der SQL-Ausdruck ist falsch. Bitte prüfen Sie die Schreibweise.');
        }
        return $query_result;
    }
}

```

```

public function real_escape_string($value) {
    $res_result = $this->link->real_escape_string($value);
    return $res_result;
}
?>

```

## 8.2 FileSystemCommandResolver

```

<?php
    include_once("Interfaces".DIRECTORY_SEPARATOR."ICommandResolver.php");

    class FileSystemCommandResolver implements ICommandResolver {

        //Ordner, in dem die Commands gefunden werden. In unserem Beispiel
        //der Commands-Ordner
        private $path;

        //falls der CommandResolver den gewünschten Command nicht findet,
        //soll er den defaultCommand nehmen.
        private $defaultCommand;

        //beim Instanziieren erfährt der FileSystemCommandResolver den Pfad der Commands,
        //und den Namen des Default Commands.
        public function __construct($path, $defaultCommand) {
            $this->path = $path;
            $this->defaultCommand = $defaultCommand;
        }

        public function getCommand(IRequest $request) {
            //Der gewünschte Command wird in Form eines Parameters im
            //Requestobjekt übergeben. Dieser Parameter hat den Namen cmd
            if($request->issetParameter('cmd')) {
                //Der Cmd-Parameter existiert, sein Wert ist der Name des gewünschten Commands
                $cmdName = $request->getParameter('cmd');
                $command = $this->loadCommand($cmdName);
                if ($command instanceof ICommand) {
                    return $command;
                }
            }
            $command = $this->loadCommand($this->defaultCommand);
            return $command;
        }

        protected function loadCommand($cmdName) {
            $class = 'cmd' . $cmdName;
            $file = $this->path . DIRECTORY_SEPARATOR . $class . '.php';

            //Wenn der Dateiname im Dateisystem nicht gefunden wird, liefert loadCommand
            //false zurück. False ist kein instanceof ICommand, daher wird die getCommand-Methode den
            //DefaultCommand zurückliefern.
            if(!file_exists($file)) {
                return false;
            }

            include_once $file;

            if(!class_exists($class, false)) {
                return false;
            }
        }
    }
}

```

```

        $command = new $class();
        return $command;
    }
}

?>
```

### 8.3 FrontController

```

<?php
class FrontController {
    private $resolver;
    public function __construct(ICommandResolver $resolver) {
        $this->resolver = $resolver;
    }

    public function handleRequest(IRequest $request, IResponse $response) {
        $command = $this->resolver->getCommand($request);
        $command->execute($request, $response);
        $response->flush();
    }
}
?>
```

### 8.4 HttpRequest

```

<?php
include_once("Interfaces".DIRECTORY_SEPARATOR."IRequest.php");

//Warum wird nicht direkt mit $_GET, $_POST oder $_REQUEST gearbeitet?
// - Grundlegende Regeln der Softwareentwicklung werden nicht eingehalten:
//   * Es wird nicht gegen eine Schnittstelle, sondern gegen eine konkrete Implementierung programmiert
//   * Die von PHP implementierte Schnittstelle zu PHP-Daten ($_GET, ...) ist nicht objektorientiert
//   Anzumerken ist, dass diese Schnittstelle zu Requestdaten als einfach empfunden wird,
//   und vermutlich zum Erfolg von PHP maßgeblich beigetragen hat.
//   * Die Zerlegung der Anwendung in Schichten wird vernachlässigt. Das führt vermutlich zur Duplizierung
//   von Code und einer enormen Steigerung von Komplexität, wenn es zur Erweiterung der Anwendung kommt.

class HTTPRequest implements IRequest {

    private $parameters = array();

    public function __construct($method="POST") {
        $method = strtoupper($method);
        //Falls über die Get-Methode der Parameter "JSON" gesetzt ist,
        //wird davon ausgegangen, dass die rohen Plaintext-Request-Daten
        //im JSON-Format ein Objekt mit allen Request-Daten enthalten.
        //Dieser Aufruf indiziert einen Ajax-Call.
        if (isset($_GET['json'])) {
            $oData = json_decode($_GLOBALS['HTTP_RAW_POST_DATA']);
            $this->setParameter('cmd',$oData->cmd);
            $this->setParameter('oData', $oData);
        }
        //Der else-Fall tritt ein, wenn der Request ganz normal über ein
        //Form-Submit oder ähnlich gemacht wird. Auch in diesem Fall kann es
        //sich um einen AJAX-Request handeln. Er ist aber nicht im JSON-Format übergeben.
        else {
            if($method == "GET") {$this->parameters=$_GET;}
            if($method == "POST") {$this->parameters=$_POST;}
            if($method == "REQUEST") {$this->parameters=$_REQUEST;}
        }
    }

    public function issetParameter($name) {
        return isset($this->parameters[$name]);
    }
}
```

```

    }

public function getParameter($name) {
    if($this->issetParameter($name)) {
        return $this->parameters[$name];
    }
    else return null;
}

public function getParameterNames() {
    //nur die Key-Spalte des Parameter-Arrays.
    return array_keys($this->parameters);
}

public function unsetParameter($name) {
    if($this->issetParameter($name)) {
        unset($this->parameters[$name]);
    }
}

//setParameter ist vielleicht etwas ungewohnt in einem HTTP-Request.
//wir brauchen diese Funktion, um den "Nicht-Ajax-Aufrufen" explizit ein Argument
//mit dem Namen nonajax mitzugeben. Das passiert in der cmdLogin.php
//Diese Markierung wird dem Request erst am Server verpasst, und zwar vom
//aufgerufenen Command
public function setParameter($name, $value) {
    if($this->issetParameter($name)) {
        return false;
    } else {
        $this->parameters[$name] = $value;
    }
}

public function getHeader($name) {
    $name = strtoupper($name);
    if (isset($_SERVER[$name])) {
        return $_SERVER[$name];
    } else return null;
}
}

?>
```

## 8.5 HttpResponse

```

<?php
include_once("Interfaces".DIRECTORY_SEPARATOR."IResponse.php");

class HttpResponse implements IResponse {
    private $status = "200 ok";
    private $headers = array();
    private $body = null;

    public function setStatus($status) {
        //Durch die Übergabe des Statuswertes in einer öffentlich zugänglichen Methode
        //an den Wert der privaten Variable "Status" wäre es möglich, aufwendige Prüfungen
        //durchzuführen, um das Speichern nicht erlaubter Stati auf dem privaten Variablenwert
        //zu verhindern.
        $this->status = $status;
    }

    public function addHeader($name, $value) {
        $this->headers[$name] = $value;
    }

    public function write($data) {
```

```

    $this->body .= $data;
}

public function flush() {
    header("HTTP/1.0 {$this->status}");
    foreach($this->headers as $name => $value) {
        header("{$name}: {$value}");
    }
    print $this->body;
    $this->headers = array();
    $this->body = null;
}

}
?>

```

## 8.6 SecurityObject

```

<?php

include_once("Interfaces".DIRECTORY_SEPARATOR."ISecurityObject.php");

class SecurityObject implements ISecurityObject {

    public function checkCredentials(IDbConnection $db, IRequest $request) {
        if ( session_is_registered( "valid_user" ) ) return true;

        $user_email = $request->getParameter("user_email");
        $password = $request->getParameter("password");

        if ( $user_email && $password ) {

            $sql = "select email, CONCAT(vorname, ',', nachname) as username, right_User, right_Anfrage, right_Buchung,
right_Leistung, right_Package, right_Partner, right_Auswertungen, right_Kunde, right_Jugendherbergen, ID_User
from tUser
where email='$user_email'
and password=md5('$password')
and aktiv = true";

            $ergebnis = $db->query($sql);

            $zeilen = $db->affected_rows();
            $userdata = $ergebnis->fetch_object();

            if ( $zeilen == 1 ) {
                //alle für die Session relevanten Daten in dieser Session speichern und somit für die restlichen
                //Files abrufbar machen...
                $_SESSION['valid_user'] = $userdata->email;
                $_SESSION['username'] = $userdata->username;
                $_SESSION['right_user'] = $userdata->right_User;
                $_SESSION['right_anfrage'] = $userdata->right_Anfrage;
                $_SESSION['right_buchung'] = $userdata->right_Buchung;
                $_SESSION['right_leistung'] = $userdata->right_Leistung;
                $_SESSION['right_package'] = $userdata->right_Package;
                $_SESSION['right_partner'] = $userdata->right_Partner;
                $_SESSION['right_auswertungen'] = $userdata->right_Auswertungen;
                $_SESSION['right_kunde'] = $userdata->right_Kunde;
                $_SESSION['right_jugendherbergen'] = $userdata->right_Jugendherbergen;
                $_SESSION['ID_User'] = $userdata->ID_User;
                //Rückgabewert der Funktion ist true, wenn der Datenbankcall eine valide Zeile ergeben hat.
                return true;
            } else return false;
            } else return false;
        }

    public function checkRight($right) {

```

```

        return $_SESSION[$right];
    }
}
?>

```

## 8.7 TemplateView

```

<?php
class TemplateView {
    //Das Model muss Templatelöcher mit Inhalt (Daten) beschicken.
    //Die Daten pro Loch werden mit der assign-Methode zur Verfügung gestellt.

    //Auf $template steht der Name der Datei im Templates-Ordner
    private $template;
    //Die Assign-Methode spricht nicht direkt "Löcher" im Template an, sondern
    //macht Einträge im $vars array.
    //Für die praktische Arbeit:
    // - Zwischen den Templateprogrammierern (HTML, CSS, JS) und den
    //   Modelprogrammierern (PHP) muss bezüglich der Keynamen am $vars-Array eine
    //   Abstimmung erfolgen.
    private $vars = array();

    //dem Konstruktor wird der Template-Name übergeben
    public function __construct($template) {
        $this->template = $template;
    }

    public function assign($name, $value) {
        $this->vars[$name] = $value;
    }

    //Wenn eine Datei mit include geladen wird, wird ein darin befindlicher
    //PHP-Code geparsed bzw. ausgeführt.
    //Wenn in den Löchern PHP-Code steht, dann wird dieser exekutiert.
    // - In unserem Modell müssen die Templateprogrammierer auch "bescheidene"
    //   PHP-Kenntnisse haben.

    //Hinter "Löcher beschicken" verbirgt sich eine Schreiboperation.
    // - Schreiben darf aber nur die Responsemethode "write".
    // - zu verhindern, dass die Schreiboperation bis zum Browser durchschlägt,
    //   ist eine wichtige Aufgabe. Lösung: Puffer einschalten.
    public function render(Response $response) {
        ob_start();
        $filename = "Templates".DIRECTORY_SEPARATOR.$this->template.DIRECTORY_SEPARATOR."vw".$this->template.".tpl";
        include_once $filename;
        $data = ob_get_clean();
        $response->write($data);
    }

    public function __get($property) {
        if(isset($this->vars[$property])) {
            return $this->vars[$property];
        }
        return null;
    }
}
?>

```

## 9 Commands

In unserer Model-View-Controller Logik entsprechen die Commands dem Bereich des Controllers. In den Commands steht der Source-Code, der die „echte“ Arbeit erledigt, also die Datenbankzugriffe, die Verwaltung der Daten, die Berechnungslogik und Ähnliches.

Die Commands sind als Klassen realisiert. Das bedeutet, die Applikation erzeugt auf Anfrage des Clients am Server ein Objekt der Command-Klasse und führt die „Execute“ Methode dieser Klasse aus (lt. Interface muss jedes Command eine Execute-Methode aufweisen). Der Name des benötigten Commands wird als POST-Parameter an den Server übergeben. Beim initialen Aufruf der index.php werden keine POST-Parameter übergeben und es wird der Standard-Command aufgerufen.

Es gibt in unserer Applikation Commands, die Daten liefern um Grids zu befüllen (cmdFill\_XY), sowie Commands, um andere Informationen zu erhalten (cmdGetXY, cmdHandleXY), um Mails zu versenden (cmdMail), Commands um Daten zu speichern (cmdInsert) und Commands, um Daten upzudaten (cmdUpdate).

### 9.1 Cmd\_Standard

```
<?php
include_once("Interfaces/ICommand.php");

class cmd_Standard implements ICommand {

    private $version="1.0 Stable";
    private $title="Smart Booking";

    public function execute(IRequest $request, IResponse $response) {
        //Der Standard-Command ist der Command, der als erstes vom FrontController aufgerufen wird
        //und entscheidet, welche weiteren Schritte passieren sollen.
        //Entweder es wird die Application gestartet, falls noch keine Server-Session besteht wird der Login für
        //diese Applikation, oder es wird ein Frontend für die Reservierung auf der Homepage generiert,
        //oder es wird ein Fehler ausgegeben.

        //Wir definieren den Seitenaufruf als "nicht-ajax" Aufruf, damit wir in der Fehlerbehandlung
        //mitbekommen, ob es ein Ajax oder Nicht-Ajax Seitenaufruf war.
        //so ist es viel einfacher, als jedem Ajax-Aufruf ein Attribut ajax zu schenken.
        $request->setParameter('nonajax', true);

        //make db-connection and user-security
        $db = new DbConnection();
        $user = new SecurityObject();

        if(isset($_GET['frontend'])) {
            $this->Frontend($response);
        } else{
            if ($user->checkCredentials($db, $request)==true) {
                $this->Application($response);
            } else {
                if($request->issetParameter('user_email') && $request->issetParameter('passwort')){
                    $text = "Leider stimmen Ihre Anmeldeinformationen nicht mit unserer Datenbank überein. Bitte versuchen Sie es
erneut!";
                } else {
                    $text = 'Willkommen beim Admin-Backend von Smart-Booking, eine Anwendung der NÖ-Jugendherbergswerke!
Bitte loggen Sie sich ein um fortzufahren...';
                }
                $this->LoginForm($text, $response, $request);
            }
        }
    }
}
```

```

    }

private function Application($response) {
    $view = new TemplateView('Application');
    $view->assign('version',$this->version);
    $view->assign('title',$this->title);
    $view->render($response);
}

private function Frontend($response) {
    $view = new TemplateView('Frontend');
    $view->assign('version',$this->version);
    $view->assign('title',$this->title);
    $view->render($response);
}

private function LoginForm($message, $response, $request) {
    $view = new TemplateView('LoginForm');
    $view->assign('message',$message);
    $view->assign('version',$this->version);
    $view->assign('title',$this->title);
    $view->render($response);
}
}

?>

```

## 9.2 cmdFill\_XY

```

<?php
include_once("Interfaces/ICommand.php");

class cmdFillAnfrageGrid implements ICommand {

    public function execute(IRequest $request, IResponse $response) {

        //make db-connection
        $db = new DbConnection();

        //check if user has permissions to proceed
        $user = new SecurityObject();
        if ($user->checkCredentials($db, $request)== true) {
            //Werte für Paging
            $start = (integer) $request->getParameter('start');
            $limit = (integer) $request->getParameter('limit');

            //Variable Suchen brauchen wir, weil wir diese Funktion auch zum Suchen in der Grid benützen wollen!
            //dieser Wert wird als zusätzliches Kriterium in den SQL-String eingebunden.
            $suchen = " (nachname LIKE '%".$request->getParameter('suchen')."%'
                or organisation LIKE '%".$request->getParameter('suchen')."%'
                or vorname LIKE '%".$request->getParameter('suchen')."%'
                or packagename LIKE '%".$request->getParameter('suchen')."%')";

            //Unterscheidung, ob ALLE (Archiv) oder nur die offenen Anfragen geladen werden
            $archiv = $request->getParameter('archiv');

            //den SQL-String bauen - bob the sql-builder :)
            if($archiv == 'true') {
                $sql = "select * from vAnfrage where".$suchen." ORDER BY ID_Anfrage DESC limit $start,$limit";
                $sql_anzahl = "select count(*) as anzahl from tAnfrage where tAnfrage.aktiv = true;";
            } else {
                $sql = "select * from vAnfrage where uebernommen = false and abgelehnt = false and".$suchen." ORDER BY
ID_Anfrage DESC limit $start,$limit";
            }
        }
    }
}

```

```

$sql_anzahl = "select count(*) as anzahl from tAnfrage where tAnfrage.aktiv = true and tAnfrage.uebernommen =
false and tAnfrage.abgelehnt = false;";
}

//Abschicken an die Datenbank
$ergebnis = $db->query($sql);
$zeilen = $db->affected_rows();

//alle Zeilen der SQL-SELECT Anweisung nacheinander in ein Array fetchen
while($zeile = $ergebnis->fetch_object()){
    $arr[] = $zeile;
}
//das $arr-Array in das JSON-Format konvertieren
$data = json_encode($arr);

//select-count ist möglicherweise langsam... Alternative?
//Link zum Thema: http://www.sqlhacks.com/index.php/Optimize/Fast_Count
$ergebnis_anzahl = $db->query($sql_anzahl);
$anzahl = $ergebnis_anzahl->fetch_object()->anzahl;

//Aufbereiten des Responses
if($zeilen == 0) $response->write('{"total":"0","results":[]}');
else $response->write('{"total":"' . $anzahl . '","results":"' . $data . '"}');

//4. Alles was geöffnet bzw. instanziert wurde, wird auch wieder
//geschlossen bzw. zerstört.
$ergebnis->free();
$ergebnis_anzahl->free();
} else $response->write("{success:false}");
}
?>

```

```

<?php
include_once("Interfaces/ICommand.php");

class cmdFillAnsprechpersonenGrid implements ICommand {

public function execute(IRequest $request, IResponse $response) {

//make db-connection
$db = new DbConnection();

//check if user has permissions to proceed
$user = new SecurityObject();
if ($user->checkCredentials($db, $request)== true) {

    $ID_Kunde= (integer) $request->getParameter('ID_Kunde');

    $sql = "select ID_Ansprechperson,ID_Kunde,nachname,vorname,telefon,email,aktiv,
        bemerkung, letzteBearbeitung from tAnsprechperson where (ID_Kunde = $ID_Kunde AND aktiv=true)";
    $sql_anzahl = "select count(*) as anzahl from tAnsprechperson where (ID_Kunde = $ID_Kunde AND tAnsprechperson.aktiv
    = true)";

    //Abschicken an die Datenbank
    $ergebnis = $db->query($sql);
    $zeilen = $db->affected_rows();

    //alle Zeilen der SQL-SELECT Anweisung nacheinander in ein Array fetchen
    while($zeile = $ergebnis->fetch_object()){
        $arr[] = $zeile;
    }
    //das $arr-Array in das JSON-Format konvertieren
    $data = json_encode($arr);
}
}

```

```

//select-count ist möglicherweise langsam... Alternative?
//Link zum Thema: http://www.sqlhacks.com/index.php/Optimize/Fast_Count
$ergebnis_anzahl = $db->query($sql_anzahl);
$anzahl = $ergebnis_anzahl->fetch_object()->anzahl;

        //Aufbereiten des Responses
if($zeilen == 0) $response->write('{"total":"0","results":[]}');
else $response->write('{"total":"' . $anzahl . '","results":"' . $data . '"}');

//4. Alles was geöffnet bzw. instanziert wurde, wird auch wieder
//geschlossen bzw. zerstört.
$ergebnis->free();
$ergebnis_anzahl->free();

} else $response->write("{success:false}");

}
?>

```

```

<?php
include_once("Interfaces/ICommand.php");

class cmdFillBuchungGrid implements ICommand {

    public function execute(IRequest $request, IResponse $response) {

        //make db-connection
        $db = new DbConnection();

        //check if user has permissions to proceed
        $user = new SecurityObject();
        if ($user->checkCredentials($db, $request)== true) {
            if($user->checkRight('right_buchung')) {

                //Werte für Paging
                $start = (integer) $request->getParameter('start');
                $limit = (integer) $request->getParameter('limit');

                $filter = $request->getParameter('filter');
                $comboFilter = $request->getParameter('comboFilter');

                if($filter=='true') {
                    //Variable Suchen brauchen wir, weil wir diese Funktion auch zum Suchen in der Grid benützen wollen!
                    //dieser Wert wird als zusätzliches Kriterium in den SQL-String eingebunden.
                    $suchen = " (resyBuchungsNr LIKE '%" . $request->getParameter('suchen') . "%'
                                or tAnsprechperson.vorname LIKE '%" . $request->getParameter('suchen') . "%'
                                or tAnsprechperson.nachname LIKE '%" . $request->getParameter('suchen') . "%'
                                or tKunde.organisation LIKE '%" . $request->getParameter('suchen') . "%'
                                or tPackage.packagename LIKE '%" . $request->getParameter('suchen') . "%'
                                AND tBuchung.aktiv=true) AND tBuchung.buchungsStatus = '$comboFilter' ";

                    $sql = "select tBuchung.ID_Buchung,tBuchung.resyBuchungsNr,tKunde.organisation as
Organisation,CONCAT(tAnsprechperson.vorname,',tAnsprechperson.nachname) as Ansprechperson,
tPackage.packagename,
tBuchung.terminAnreise,tBuchung.ersatzTerminAnreise,tBuchung.kinder,tBuchung.erwachsene,tBuchung.female,tBuchung.male,tBuchung.vegetarier,
tBuchung.relVorschriften,tBuchung.allergien,CONCAT(tUser.vorname,' ',tUser.nachname) as
username,tBuchung.letzteBearbeitung,tBuchung.erstelltAm,tBuchung.buchungsStatus,tBuchung.ID_Package
from tPackage right join ((tAnsprechperson left join tKunde ON tAnsprechperson.ID_Kunde =
tKunde.ID_Kunde) right join
(tBuchung left join tUser ON tBuchung.ID_User = tUser.ID_User) ON
tAnsprechperson.ID_Ansprechperson = tBuchung.ID_Ansprechperson)
ON tPackage.ID_Package = tBuchung.ID_Package where ".$suchen." ORDER BY tBuchung.ID_Buchung
DESC limit $start,$limit";
                }
            }
        }
    }
}

```

```

$ssql_anzahl = "select count(*) as anzahl from tBuchung where tBuchung.aktiv = true AND
tBuchung.buchungsStatus = '$comboFilter'";
} else {
    //Variable Suchen brauchen wir, weil wir diese Funktion auch zum Suchen in der Grid benützen wollen!
    //dieser Wert wird als zusätzliches Kriterium in den SQL-String eingebunden.
    $suchen = "(resyBuchungsNr LIKE '%" . $request->getParameter('suchen') . "%'
    or tAnsprechperson.vorname LIKE '%" . $request->getParameter('suchen') . "%'
    or tAnsprechperson.nachname LIKE '%" . $request->getParameter('suchen') . "%'
    or tKunde.organisation LIKE '%" . $request->getParameter('suchen') . "%'
    or tPackage.packagename LIKE '%" . $request->getParameter('suchen') . "%'
    AND tBuchung.aktiv=true)";

    $sql = "select tBuchung.ID_Buchung,tBuchung.resyBuchungsNr,tKunde.organisation as
Organisation,CONCAT(tAnsprechperson.vorname,',',tAnsprechperson.nachname) as Ansprechperson,
tPackage.packagename,
tBuchung.terminAnreise,tBuchung.ersatzTerminAnreise,tBuchung.kinder,tBuchung.erwachsene,tBuchung.female,tBuchung.male,tBuc
hung.vegetarier,
tBuchung.relVorschriften,tBuchung.allergien,CONCAT(tUser.vorname,' ',tUser.nachname) as
username,tBuchung.letzteBearbeitung,tBuchung.erstelltAm,tBuchung.buchungsStatus, tBuchung.ID_Package
from tPackage right join ((tAnsprechperson left join tKunde ON tAnsprechperson.ID_Kunde =
tKunde.ID_Kunde) right join
(tBuchung left join tUser ON tBuchung.ID_User = tUser.ID_User) ON
tAnsprechperson.ID_Ansprechperson = tBuchung.ID_Ansprechperson)
ON tPackage.ID_Package = tBuchung.ID_Package where ".$suchen." ORDER BY tBuchung.ID_Buchung
DESC limit $start,$limit";
    $sql_anzahl = "select count(*) as anzahl from tBuchung where tBuchung.aktiv = true";
}

//Abschicken an die Datenbank
$ergebnis = $db->query($sql);
$zeilen = $db->affected_rows();

//alle Zeilen der SQL-SELECT Anweisung nacheinander in ein Array fetchen
while($zeile = $ergebnis->fetch_object()){
    $arr[] = $zeile;
}
//das $arr-Array in das JSON-Format konvertieren
$data = json_encode($arr);

//select-count ist möglicherweise langsam... Alternative?
//Link zum Thema: http://www.sqlhacks.com/index.php/Optimize/Fast_Count
$ergebnis_anzahl = $db->query($sql_anzahl);
$anzahl = $ergebnis_anzahl->fetch_object()->anzahl;

//Aufbereiten des Responses
if($zeilen == 0) $response->write('{"total":"0","results":[]}');
else $response->write('{"total":"' . $anzahl . '","results":"' . $data . '}');

//4. Alles was geöffnet bzw. instanziert wurde, wird auch wieder
//geschlossen bzw. zerstört.
$ergebnis->free();
$ergebnis_anzahl->free();
} else $response->write("{success:false}");
} else $response->write("{success:false}");
}

}
?>

```

```

<?php
include_once("Interfaces/ICommand.php");

class cmdFillBuchungshinweisGrid implements ICommand {

    public function execute(IRequest $request, IResponse $response) {
        //make db-connection
    }
}

```

```

$db = new DbConnection();

//check if user has permissions to proceed
$user = new SecurityObject();
if ($user->checkCredentials($db, $request)== true) {

    $ID_Buchung = (integer)$request->getParameter('ID_Buchung');

    $sql = "select tBuchungsinfo.ID_Buchungsinfo, tBuchungsinfo.infoDatum, tBuchungsinfo.infotext, CONCAT(tUser.vorname,
',tUser.nachname) as user from tBuchungsinfo inner join tUser on (tBuchungsinfo.ID_User = tUser.ID_User) where
tBuchungsinfo.ID_Buchung = $ID_Buchung";

    //Abschicken an die Datenbank
    $ergebnis = $db->query($sql);
    $zeilen = $db->affected_rows();

    //alle Zeilen der SQL-SELECT Anweisung nacheinander in ein Array fetchen
    while($zeile = $ergebnis->fetch_object()){
        $arr[] = $zeile;
    }

    //das $arr-Array in das JSON-Format konvertieren
    $data = json_encode($arr);

    //Aufbereiten des Responses
    if($zeilen == 0) $response->write('{"total":"0","results":[]}');
    else $response->write('{"total":"' . $zeilen . '","results":"' . $data . '}');

    //4. Alles was geöffnet bzw. instanziiert wurde, wird auch wieder
    //geschlossen bzw. zerstört.
    $ergebnis->free();

    } else $response->write("{success:false}");
}
?>
```

```

<?php
include_once("Interfaces/ICommand.php");

class cmdFillJHBAuswertungGrid implements ICommand {

    public function execute(IRequest $request, IResponse $response) {

        //make db-connection
        $db = new DbConnection();

        //check if user has permissions to proceed
        $user = new SecurityObject();
        if ($user->checkCredentials($db, $request)== true) {

            //Werte für Paging
            // $start = (integer) $request->getParameter('start');
            // $limit = (integer) $request->getParameter('limit');
            $JHB = (integer) $request->getParameter('jhb');
            $AnfangsDatum = $request->getParameter('Vontermin');
            $EndDatum = $request->getParameter('Bistermin');

            //Variable Suchen brauchen wir, weil wir diese Funktion auch zum Suchen in der Grid benützen wollen!
            //dieser Wert wird als zusätzliches Kriterium in den SQL-String eingebunden.
            $suchen = " tLeistung.Leistung LIKE '%" . $request->getParameter('suchen') . "%' AND tLeistung.aktiv=true";

            $sql = "SELECT
```

```

tJhb.jhb,
tBuchung.resyBuchungsNr,
tKunde.organisation,
CONCAT(tAnsprechperson.nachname, ',', tAnsprechperson.vorname) AS Ansprechperson,
tPackage.packagename,
tBuchung.terminAnreise,
tBuchung.buchungsStatus,
tBuchung.kinder,
tBuchung.erwachsene,
tBuchung.kinder + tBuchung.erwachsene AS Gesamt
FROM
tJhb INNER JOIN tJhb_Package ON
tJhb.ID_Jhb = tJhb_Package.ID_Jhb
INNER JOIN tPackage ON
tJhb_Package.ID_Package = tPackage.ID_Package
INNER JOIN tBuchung ON
tPackage.ID_Package = tBuchung.ID_Package
INNER JOIN tAnsprechperson ON
tBuchung.ID_Ansprechperson = tAnsprechperson.ID_Ansprechperson
INNER JOIN tKunde ON
tAnsprechperson.ID_Kunde = tKunde.ID_Kunde
WHERE
tJhb.ID_Jhb = "'.$JHB.'" AND tBuchung.terminAnreise BETWEEN "'.$AnfangsDatum.'" AND "'.$EndDatum.'"
ORDER BY tBuchung.terminAnreise ASC,";

//$sql_anzahl = "select count(*) as anzahl from tLeistung where tLeistung.aktiv = true;";
//Abschicken an die Datenbank

//echo $sql;
$ergebnis = $db->query($sql);
$zeilen = $db->affected_rows();

//alle Zeilen der SQL-SELECT Anweisung nacheinander in ein Array fatchen
while($zeile = $ergebnis->fetch_object()){
    $arr[] = $zeile;
}
//das $arr-Array in das JSON-Format konvertieren
$data = json_encode($arr);

//select-count ist möglicherweise langsam... Alternative?
//Link zum Thema: http://www.sqlhacks.com/index.php/Optimize/Fast_Count
//$/ergebnis_anzahl = $db->query($sql_anzahl);
//$/anzahl = $ergebnis_anzahl->fetch_object()->anzahl;

//Aufbereiten des Responses
if($zeilen == 0) $response->write('{"total":0,"results":[]}');
else $response->write('{"total":"' . $zeilen . '","results":"' . $data . '}');

//4. Alles was geöffnet bzw. instanziert wurde, wird auch wieder
//geschlossen bzw. zerstört.
$ergebnis->free();
//$/ergebnis_anzahl->free();

} else $response->write("{success:false}");

}

?>
```

```

<?php
include_once("Interfaces/ICommand.php");

class cmdFillJhbGrid implements ICommand {

    public function execute(IRequest $request, IResponse $response) {
```

```

//make db-connection
$db = new DbConnection();

//check if user has permissions to proceed
$user = new SecurityObject();
if ($user->checkCredentials($db, $request)== true) {

    //Werte für Paging
    $start = (integer) $request->getParameter('start');
    $limit = (integer) $request->getParameter('limit');

    //Variable Suchen brauchen wir, weil wir diese Funktion auch zum Suchen in der Grid benützen wollen!
    //dieser Wert wird als zusätzliches Kriterium in den SQL-String eingebunden.
    $suchen = " jhb LIKE '%".$request->getParameter('suchen')."'% AND aktiv=true";

    $sql = "select * FROM tJhb WHERE" . $suchen . " ORDER BY jhb ASC limit $start,$limit";
    $sql_anzahl = "select count(*) as anzahl from tJhb where tJhb.aktiv = true;";

    //Abschicken an die Datenbank
    $ergebnis = $db->query($sql);
    $zeilen = $db->affected_rows();

    //alle Zeilen der SQL-SELECT Anweisung nacheinander in ein Array fetchen
    while($zeile = $ergebnis->fetch_object()){
        $arr[] = $zeile;
    }

    //das $arr-Array in das JSON-Format konvertieren
    $data = json_encode($arr);

    //select-count ist möglicherweise langsam... Alternative?
    //Link zum Thema: http://www.sqlhacks.com/index.php/Optimize/Fast_Count
    $ergebnis_anzahl = $db->query($sql_anzahl);
    $anzahl = $ergebnis_anzahl->fetch_object()->anzahl;

    //Aufbereiten des Responses
    if($zeilen == 0) $response->write('{"total":0,"results":[]}');
    else $response->write('{"total":"' . $anzahl . '","results":"' . $data . '"}');

    //4. Alles was geöffnet bzw. instanziert wurde, wird auch wieder
    //geschlossen bzw. zerstört.
    $ergebnis->free();
    $ergebnis_anzahl->free();

} else $response->write("{success:false}");
}
?>

```

```

<?php
include_once("Interfaces/ICommand.php");

class cmdFillJhbPackGrid implements ICommand {

    public function execute(IRequest $request, IResponse $response) {

        $JHB_ID = $request->getParameter('ID_Jhb');

        //make db-connection
        $db = new DbConnection();

        //check if user has permissions to proceed
        $user = new SecurityObject();
        if ($user->checkCredentials($db, $request)== true) {

```

```

$sql = "select tPackage.ID_Package, tPackage.packagename FROM
tPackage inner join tJhb_Package on tPackage.ID_Package = tJhb_Package.ID_Package WHERE ID_Jhb = $JHB_ID ORDER
BY packagename ASC";

//Abschicken an die Datenbank
$ergebnis = $db->query($sql);
$zeilen = $db->affected_rows();

//alle Zeilen der SQL-SELECT Anweisung nacheinander in ein Array fetchen
while($zeile = $ergebnis->fetch_object()){
    $arr[] = $zeile;
}

//das $arr-Array in das JSON-Format konvertieren
$data = json_encode($arr);

//Aufbereiten des Responses
if($zeilen == 0) $response->write('{"total":"0","results":[]}');
else $response->write('{"total":"' . $anzahl . '","results":' . $data . '}');

//4. Alles was geöffnet bzw. instanziert wurde, wird auch wieder
//geschlossen bzw. zerstört.
$ergebnis->free();

} else $response->write("{success:false}");
}
?

```

```

<?php
include_once("Interfaces/ICommand.php");

class cmdFillKundenGrid implements ICommand {

    public function execute(IRequest $request, IResponse $response) {

        //make db-connection
        $db = new DbConnection();

        //check if user has permissions to proceed
        $user = new SecurityObject();
        if ($user->checkCredentials($db, $request)== true) {

            //Werte für Paging
            $start = (integer)$request->getParameter('start');
            $limit = (integer) $request->getParameter('limit');

            //Variable Suchen brauchen wir, weil wir diese Funktion auch zum Suchen in der Grid benützen wollen!
            //dieser Wert wird als zusätzliches Kriterium in den SQL-String eingebunden.
            $suchen = "((nachname LIKE '%" . $request->getParameter('suchen') . "%' OR organisation LIKE '%" . $request-
>getParameter('suchen') . "%') AND tKunde.aktiv=true)";

            $sql = "select
tKunde.ID_Kunde,tKunde.kategoriek,tKunde.organisation,tKunde.adresse,tKunde.plz,tKunde.ort,tKunde.telefon,
tKunde.fax,tKunde.email,CONCAT(tUser.vorname, ' ',tUser.nachname) as username, tKunde.letzteBearbeitung,
tKunde.resykd,tKunde.bemerkung,tKunde.erstelltAm from tKunde left join tUser
ON tKunde.ID_User = tUser.ID_User where ".$suchen." limit $start,$limit";
            $sql_anzahl = "select count(*) as anzahl from tKunde where tKunde.aktiv = true";

            //Abschicken an die Datenbank
            $ergebnis = $db->query($sql);
            $zeilen = $db->affected_rows();

            //alle Zeilen der SQL-SELECT Anweisung nacheinander in ein Array fetchen
            while($zeile = $ergebnis->fetch_object()){


```

```

        $arr[] = $zeile;
    }

    //das $arr-Array in das JSON-Format konvertieren
    $data = json_encode($arr);

    //select-count ist möglicherweise langsam... Alternative?
    //Link zum Thema: http://www.sqlhacks.com/index.php/Optimize/Fast_Count
    $ergebnis_anzahl = $db->query($sql_anzahl);
    $anzahl = $ergebnis_anzahl->fetch_object()->anzahl;

    //Aufbereiten des Responses
    if($zeilen == 0) $response->write('{"total": "0", "results": []}');
    else $response->write('{"total": "' . $anzahl . '", "results": ' . $data . '}');

    //4. Alles was geöffnet bzw. instanziert wurde, wird auch wieder
    //geschlossen bzw. zerstört.
    $ergebnis->free();
    $ergebnis_anzahl->free();

    } else $response->write("{success:false}");
}
?>
```

```

<?php
include_once("Interfaces/ICommand.php");

class cmdFillLeistungGrid implements ICommand {

    public function execute(IRequest $request, IResponse $response) {

        //make db-connection
        $db = new DbConnection();

        //check if user has permissions to proceed
        $user = new SecurityObject();
        if ($user->checkCredentials($db, $request)== true) {

            //Werte für Paging
            $start = (integer) $request->getParameter('start');
            $limit = (integer) $request->getParameter('limit');

            //Variable Suchen brauchen wir, weil wir diese Funktion auch zum Suchen in der Grid benützen wollen!
            //dieser Wert wird als zusätzliches Kriterium in den SQL-String eingebunden.
            $suchen = " tLeistung.Leistung LIKE '%" . $request->getParameter('suchen') . "%' AND tLeistung.aktiv=true";

            $sql = "select tLeistung.ID_Leistung, tLeistung.Leistung, tLeistung.StandardUhrzeit, tLeistung.LeistungsBemerkung,
tLeistung.ID_Partner, tPartner.firmenname from tLeistung inner join tPartner on tLeistung.ID_Partner = tPartner.ID_Partner WHERE".
$suchen . " limit $start,$limit";
            $sql_anzahl = "select count(*) as anzahl from tLeistung where tLeistung.aktiv = true";

            //Abschicken an die Datenbank
            $ergebnis = $db->query($sql);
            $zeilen = $db->affected_rows();

            //alle Zeilen der SQL-SELECT Anweisung nacheinander in ein Array fetchen
            while($zeile = $ergebnis->fetch_object()){
                $arr[] = $zeile;
            }

            //das $arr-Array in das JSON-Format konvertieren
            $data = json_encode($arr);

            //select-count ist möglicherweise langsam... Alternative?
```

```
//Link zum Thema: http://www.sqlhacks.com/index.php/Optimize/Fast_Count
$ergebnis_anzahl = $db->query($sql_anzahl);
$anzahl = $ergebnis_anzahl->fetch_object()->anzahl;

//Aufbereiten des Responses
if($zeilen == 0) $response->write('({"total":"0","results":[]})');
else $response->write('({"total":"' . $anzahl . '","results":"' . $data . '"})');

//4. Alles was geöffnet bzw. instanziert wurde, wird auch wieder
//geschlossen bzw. zerstört.
$ergebnis->free();
$ergebnis_anzahl->free();

} else $response->write("{success:false}");

}
?>
```

```
<?php
include_once("Interfaces/ICommand.php");

class cmdFillPackageGrid implements ICommand {

    public function execute(IRequest $request, IResponse $response) {

        $User_ID = $request->getParameter('ID_User');

        //make db-connection
        $db = new DbConnection();

        //check if user has permissions to proceed
        $user = new SecurityObject();
        if ($user->checkCredentials($db, $request)== true) {

            //Werte für Paging
            $start = (integer) $request->getParameter('start');
            $limit = (integer) $request->getParameter('limit');

            //Variable Suchen brauchen wir, weil wir diese Funktion auch zum Suchen in der Grid benützen wollen!
            //dieser Wert wird als zusätzliches Kriterium in den SQL-String eingebunden.
            $suchen = " tPackage.packagename LIKE '%" . $request->getParameter('suchen') . "%' AND tPackage.aktiv=true";

            $sql = "select tPackage.ID_Package,
                    tPackage.packagename,
                    tPackage.pdfPfad,
                    concat(tUser.nachname, ' ', tUser.vorname) as username,
                    tPackage.letzteBearbeitung
                from tPackage inner join tUser
                on tPackage.ID_User = tUser.ID_User
                where " . $suchen . " ORDER BY tPackage.packagename ASC limit $start,$limit";

            $sql_anzahl = "select count(*) as anzahl from tPackage where tPackage.aktiv = true;";

            //Abschicken an die Datenbank
            $ergebnis = $db->query($sql);
            $zeilen = $db->affected_rows();

            //alle Zeilen der SQL-SELECT Anweisung nacheinander in ein Array fetchen
            while($zeile = $ergebnis->fetch_object()){
                $arr[] = $zeile;
            }

            //das $arr-Array in das JSON-Format konvertieren
            $data = json_encode($arr);
```

```

//select-count ist möglicherweise langsam... Alternative?
//Link zum Thema: http://www.sqlhacks.com/index.php/Optimize/Fast_Count
$ergebnis_anzahl = $db->query($sql_anzahl);
$anzahl = $ergebnis_anzahl->fetch_object()->anzahl;

//Aufbereiten des Responses
if($zeilen == 0) $response->write('{"total":"0","results":[]}');
else $response->write('{"total":"' . $anzahl . '","results":"' . $data . '}');

//4. Alles was geöffnet bzw. instanziert wurde, wird auch wieder
//geschlossen bzw. zerstört.
$ergebnis->free();
$ergebnis_anzahl->free();

} else $response->write("{success:false}");
}
?>
<?php
include_once("Interfaces/ICommand.php");

class cmdFillPackLeistungGrid implements ICommand {

public function execute(IRequest $request, IResponse $response) {

//make db-connection
$db = new DbConnection();

//check if user has permissions to proceed
$user = new SecurityObject();
if ($user->checkCredentials($db, $request)== true) {

$PACK_ID = $request->getParameter('ID_Package');

$sql = "SELECT tPackageleistung.ID_Leistung,
           tLeistung.Leistung,
           tPackageleistung.leistungstag,
           tLeistung.StandardUhrzeit
      FROM tPackageleistung inner join tLeistung
        on tPackageleistung.ID_Leistung = tLeistung.ID_Leistung
 WHERE tPackageleistung.ID_Package = $PACK_ID ORDER BY leistungstag ASC",

//Abschicken an die Datenbank
$ergebnis = $db->query($sql);
$zeilen = $db->affected_rows();

//alle Zeilen der SQL-SELECT Anweisung nacheinander in ein Array fetchen
while($zeile = $ergebnis->fetch_object()){
    $arr[] = $zeile;
}

//das $arr-Array in das JSON-Format konvertieren
$data = json_encode($arr);

//Aufbereiten des Responses
if($zeilen == 0) $response->write('{"total":"0","results":[]}');
else $response->write('{"total":"' . $zeilen . '","results":"' . $data . '}');

//4. Alles was geöffnet bzw. instanziert wurde, wird auch wieder
//geschlossen bzw. zerstört.
$ergebnis->free();

} else $response->write("{success:false}");
}

```

```
}
```

```
<?php
include_once("Interfaces/ICommand.php");

class cmdFillPartnerAuswertungGrid implements ICommand {

    public function execute(IRequest $request, IResponse $response) {

        //make db-connection
        $db = new DbConnection();

        //check if user has permissions to proceed
        $user = new SecurityObject();
        if ($user->checkCredentials($db, $request)== true) {

            //Werte für Paging
            // $start = (integer) $request->getParameter('start');
            // $limit = (integer) $request->getParameter('limit');
            $Partner = (integer) $request->getParameter('jhb');
            $AnfangsDatum = $request->getParameter('Vontermin');
            $EndDatum = $request->getParameter('Bistermin');

            //Variable Suchen brauchen wir, weil wir diese Funktion auch zum Suchen in der Grid benützen wollen!
            //dieser Wert wird als zusätzliches Kriterium in den SQL-String eingebunden.
            // $suchen = " tleistung.Leistung LIKE '%" . $request->getParameter('suchen') . "%' AND tleistung.aktiv=true";

            $sql = "SELECT
                    tPartner.firmenname,
                    tKunde.organisation,
                    CONCAT(tAnsprechperson.nachname, ' ', tAnsprechperson.vorname) AS Ansprechperson,
                    ADDDATE(tBuchung.terminAnreise, tPackageleistung.leistungstag) AS Termin,
                    tLeistung.StandardUhrzeit,
                    tBuchung.erwachsene,
                    tBuchung.kinder,
                    CONCAT(tBuchung.kinder+tBuchung.erwachsene) AS Gesamt_Personen,
                    tBuchung.buchungsStatus

                FROM
                    tPartner INNER JOIN tLeistung ON
                    tPartner.ID_Partner = tLeistung.ID_Partner
                    INNER JOIN tPackageleistung ON
                    tLeistung.ID_Leistung = tPackageleistung.ID_Leistung
                    INNER JOIN tPackage ON
                    tPackageleistung.ID_Package = tPackage.ID_Package
                    INNER JOIN tBuchung ON
                    tPackage.ID_Package = tBuchung.ID_Package
                    INNER JOIN tAnsprechperson ON
                    tBuchung.ID_Ansprechperson = tAnsprechperson.ID_Ansprechperson
                    INNER JOIN tKunde ON
                    tAnsprechperson.ID_Kunde = tKunde.ID_Kunde
                WHERE
                    tPartner.ID_Partner = '$Partner' AND tBuchung.terminAnreise BETWEEN '$AnfangsDatum' AND
                    '$EndDatum' ORDER BY Termin ASC;
                    ";
                    // $sql_anzahl = "select count(*) as anzahl from tLeistung where tLeistung.aktiv = true;";

                    //Abschicken an die Datenbank
                    $ergebnis = $db->query($sql);
                    $zeilen = $db->affected_rows();
```

```

//alle Zeilen der SQL-SELECT Anweisung nacheinander in ein Array fetchen
while($zeile = $ergebnis->fetch_object()){
    $arr[] = $zeile;
}
//das $arr-Array in das JSON-Format konvertieren
$data = json_encode($arr);

//select-count ist möglicherweise langsam... Alternative?
//Link zum Thema: http://www.sqlhacks.com/index.php/Optimize/Fast_Count
//$ergebnis_anzahl = $db->query($sql_anzahl);
//$anzahl = $ergebnis_anzahl->fetch_object()->anzahl;

        //Aufbereiten des Responses
if($zeilen == 0) $response->write('{"total":"0","results":[]}');
else $response->write('{"total":"' . $zeilen . '","results":"' . $data . '}');

//4. Alles was geöffnet bzw. instanziert wurde, wird auch wieder
//geschlossen bzw. zerstört.
$ergebnis->free();
//$ergebnis_anzahl->free();

} else $response->write("{success:false}");

}
?>

```

```

<?php
include_once("Interfaces/ICommand.php");

class cmdFillPartnerGrid implements ICommand {

    public function execute(IRequest $request, IResponse $response) {

        //make db-connection
        $db = new DbConnection();

        //check if user has permissions to proceed
        $user = new SecurityObject();
        if ($user->checkCredentials($db, $request)== true) {

            //Werte für Paging
            $start = (integer)$request->getParameter('start');
            $limit = (integer) $request->getParameter('limit');

            //Variable Suchen brauchen wir, weil wir diese Funktion auch zum Suchen in der Grid benützen wollen!
            //dieser Wert wird als zusätzliches Kriterium in den SQL-String eingebunden.
            $suchen = " (tPartner.nachname LIKE '%" . $request->getParameter('suchen') . "%' or tPartner.firmenname LIKE
            '%" . $request->getParameter('suchen') . "%' or tPartner.plz LIKE '%" . $request->getParameter('suchen') . "%') AND tPartner.aktiv=true";

            $sql = "select
tPartner.ID_Partner,tPartner.firmenname,tPartner.vorname,tPartner.nachname,tPartner.adresse,tPartner.plz,tPartner.ort,
tPartner.tel,tPartner.email,CONCAT(tUser.vorname, ',tUser.nachname) as username, tPartner.letzteBearbeitung
from tPartner left join tUser
        ON tPartner.ID_User = tUser.ID_User where" . $suchen . " limit $start,$limit";
            $sql_anzahl = "select count(*) as anzahl from tPartner where tPartner.aktiv = true;";

            //Abschicken an die Datenbank
            $ergebnis = $db->query($sql);
            $zeilen = $db->affected_rows();

            //alle Zeilen der SQL-SELECT Anweisung nacheinander in ein Array fetchen
            while($zeile = $ergebnis->fetch_object()){
                $arr[] = $zeile;
}
//das $arr-Array in das JSON-Format konvertieren
$data = json_encode($arr);

```

```

//select-count ist möglicherweise langsam... Alternative?
//Link zum Thema: http://www.sqlhacks.com/index.php/Optimize/Fast_Count
$ergebnis_anzahl = $db->query($sql_anzahl);
$anzahl = $ergebnis_anzahl->fetch_object()->anzahl;

//Aufbereiten des Responses
if($zeilen == 0) $response->write('{"total":"0","results":[]}');
else $response->write('{"total":"' . $anzahl . '","results":"' . $data . '"}');

//Alles was geöffnet bzw. instanziert wurde, wird auch wieder
//geschlossen bzw. zerstört.
$ergebnis->free();
$ergebnis_anzahl->free();

} else $response->write("{success:false}");
}
?>

```

```

<?php
include_once("Interfaces/ICommand.php");

class cmdFillTimetableGrid implements ICommand {

public function execute(IRequest $request, IResponse $response) {

//make db-connection
$db = new DbConnection();

//check if user has permissions to proceed
$user = new SecurityObject();
if ($user->checkCredentials($db, $request)== true) {

$id_buchung = $request->getParameter('ID_Buchung');
$sql = "SELECT tLeistungszeitpunkt.ID_LeistungsZeitpunkt, tLeistungszeitpunkt.ID_Buchung, tLeistung.Leistung,
tLeistungszeitpunkt.EchtUhrzeit, tLeistungszeitpunkt.EchtDatum FROM tLeistungszeitpunkt INNER JOIN tLeistung
ON (tLeistungszeitpunkt.ID_Leistung = tLeistung.ID_Leistung) WHERE ID_Buchung = $id_buchung";

//Abschicken an die Datenbank
$ergebnis = $db->query($sql);
$zeilen = $db->affected_rows();

//alle Zeilen der SQL-SELECT Anweisung nacheinander in ein Array fetchen
while($zeile = $ergebnis->fetch_object()) {
$arr[] = $zeile;
}

//das $arr-Array in das JSON-Format konvertieren
$data = json_encode($arr);

//Aufbereiten des Responses
if($zeilen == 0) $response->write('{"total":"0","results":[]}');
else $response->write('{"total":"' . $anzahl . '","results":"' . $data . '"}');

//. Alles was geöffnet bzw. instanziert wurde, wird auch wieder
//geschlossen bzw. zerstört.
$ergebnis->free();

} else $response->write("{success:false}");
}
?>

```

```

<?php
include_once("Interfaces/ICommand.php");

class cmdFillUserGrid implements ICommand {

    public function execute(IRequest $request, IResponse $response) {

        //make db-connection
        $db = new DbConnection();

        //check if user has permissions to proceed
        $user = new SecurityObject();
        if ($user->checkCredentials($db, $request)== true) {

            //Werte für Paging
            $start = (integer) $request->getParameter('start');
            $limit = (integer) $request->getParameter('limit');

            //Variable Suchen brauchen wir, weil wir diese Funktion auch zum Suchen in der Grid benützen wollen!
            //dieser Wert wird als zusätzliches Kriterium in den SQL-String eingebunden.
            $suchen = " ((nachname LIKE '%" . $request->getParameter('suchen') . "%' or email LIKE '%" . $request-
>getParameter('suchen') . "%') AND tUser.aktiv=true)";

            $sql = "select ID_User,vorname,nachname,email,adresse,plz,ort,
                    right_User,right_Anfrage,right_Buchung,right_Leistung, right_Package,
                    right_Partner,right_Auswertungen,right_Kunde,right_Jugendherbergen from tUser
                    where". $suchen." limit $start,$limit";
            $sql_anzahl = "select count(*) as anzahl from tUser where tUser.aktiv = true";

            //Abschicken an die Datenbank
            $ergebnis = $db->query($sql);
            $zeilen = $db->affected_rows();

            //alle Zeilen der SQL-SELECT Anweisung nacheinander in ein Array fetchen
            while($zeile = $ergebnis->fetch_object()){
                $arr[] = $zeile;
            }

            //das $arr-Array in das JSON-Format konvertieren
            $data = json_encode($arr);

            //select-count ist möglicherweise langsam... Alternative?
            //Link zum Thema: http://www.sqlhacks.com/index.php/Optimize/Fast_Count
            $ergebnis_anzahl = $db->query($sql_anzahl);
            $anzahl = $ergebnis_anzahl->fetch_object()->anzahl;

            //Aufbereiten des Responses
            if($zeilen == 0) $response->write('{"total":0,"results":[]}');
            else $response->write('{"total":"' . $anzahl . '","results":"' . $data . '"}');

            //Alles was geöffnet bzw. instanziert wurde, wird auch wieder
            //geschlossen bzw. zerstört.
            $ergebnis->free();
            $ergebnis_anzahl->free();

            } else $response->write("{success:false}");
        }
    ?>
}

```

### 9.3 cmdGet\_XY

```

<?php
include_once("Interfaces/ICommand.php");

```

```

class cmdGetJhb implements ICommand {
    public function execute(IRequest $request, IResponse $response) {
        //make db-connection
        $db = new DbConnection();

        $sql = "select ID_Jhb, jhb from tJhb WHERE aktiv = true ORDER BY jhb ASC";

        $ergebnis = $db->query($sql);
        $zeilen = $db->affected_rows();

        while($zeile = $ergebnis->fetch_object()){
            $arr[] = $zeile;
        }
        $data = json_encode($arr);

        $response->write('{"total":"' . $zeilen . '", "results":"' . $data . '}');
    }
}
?>

```

```

<?php
include_once("Interfaces/ICommand.php");

class cmdGetLeistung implements ICommand {
    public function execute(IRequest $request, IResponse $response) {
        //make db-connection
        $db = new DbConnection();

        //check if user has permissions to proceed
        $user = new SecurityObject();
        if ($user->checkCredentials($db, $request)== true) {

            $sql = "select
                    tLeistung.ID_Leistung,
                    tLeistung.Leistung
                from tLeistung where tLeistung.aktiv = true";

            $ergebnis = $db->query($sql);
            $zeilen = $db->affected_rows();

            while($zeile = $ergebnis->fetch_object()){
                $arr[] = $zeile;
            }

            $data = json_encode($arr);

            $response->write('{"total":"' . $zeilen . '", "results":"' . $data . '}');

        } else $response->write("{success:false}");
    }
}
?>

```

```

<?php
include_once("Interfaces/ICommand.php");

class cmdGetMergeSuggestions implements ICommand {
    public function execute(IRequest $request, IResponse $response) {

```

```

//make db-connection
$db = new DbConnection();

//Werte für Paging
$start = (integer) $request->getParameter('start');
$limit = (integer) $request->getParameter('limit');
$alle = $request->getParameter('alle');

$plz = $request->getParameter('plz');
if ($plz == "") $plz = 'null';

$name = $request->getParameter('name');
if ($name == "") $name = 'null';
$name = substr($name,0,4);

$resykd = $request->getParameter('resykd');
if($resykd=="") $resykd = 'null';

//check if user has permissions to proceed
$user = new SecurityObject();
if ($user->checkCredentials($db, $request)== true) {
    if($alle == 'true') {
        $sql = "select ID_Kunde,kategoriek,organisation,adresse,plz from tKunde WHERE aktiv = true limit $start,$limit";
    }
    else {
        $sql = "select ID_Kunde,kategoriek,organisation,adresse,plz from tKunde WHERE (resykd = '$resykd' OR plz = '$plz' OR organisation LIKE '%$name%') AND aktiv = true limit $start,$limit";
    }
    $ergebnis = $db->query($sql);
    $zeilen = $db->affected_rows();

    if($alle == 'true') {
        $sql_anzahl = "select count(*) as anzahl from tKunde WHERE aktiv = true";
    } else {
        $sql_anzahl = "select count(*) as anzahl from tKunde WHERE (resykd = '$resykd' OR plz = '$plz' OR organisation LIKE '%$name%') AND aktiv = true";
    }
    //select-count ist möglicherweise langsam... Alternative?
    //Link zum Thema: http://www.sqlhacks.com/index.php/Optimize/Fast_Count

    $ergebnis_anzahl = $db->query($sql_anzahl);
    $anzahl = $ergebnis_anzahl->fetch_object()->anzahl;

    while($zeile = $ergebnis->fetch_object()){
        $arr[] = $zeile;
    }
    $data = json_encode($arr);
    $response->write("{\"total\":\"" . $anzahl . "\",\"results\":\"" . $data . '\"}');

    } else $response->write("{success:false}");
}
}
?>

```

```

<?php
include_once("Interfaces/ICommand.php");

class cmdGetPackage implements ICommand {

    public function execute(IRequest $request, IResponse $response) {

        //make db-connection
        $db = new DbConnection();

```

```

if($request->issetParameter('ID_Jhb')) {
    $ID_JHB = $request->getParameter('ID_Jhb');
    $sql = "SELECT tPackage.ID_Package, packagename FROM tPackage INNER JOIN tJhb_Package ";
    $sql = $sql."ON (tPackage.ID_Package = tJhb_Package.ID_Package) WHERE (tPackage.aktiv = TRUE AND
    tJhb_Package.ID_Jhb = '$ID_JHB')";
} else {
    $sql = "select ID_Package, packagename from tPackage WHERE aktiv = true";
}

$ergebnis = $db->query($sql);
$zeilen = $db->affected_rows();

while($zeile = $ergebnis->fetch_object()){
    $arr[] = $zeile;
}
$data = json_encode($arr);
$response->write('{"total":"' . $zeilen . '","results":"' . $data . '"}');
}
?>

```

```

<?php
include_once("Interfaces/ICommand.php");

class cmdGetPartner implements ICommand {

public function execute(IRequest $request, IResponse $response) {

//make db-connection
$db = new DbConnection();

//check if user has permissions to proceed
$user = new SecurityObject();
if ($user->checkCredentials($db, $request)== true) {

$sql = "select ID_Partner, firmenname from tPartner WHERE aktiv = true";
$ergebnis = $db->query($sql);
$zeilen = $db->affected_rows();

while($zeile = $ergebnis->fetch_object()){
    $arr[] = $zeile;
}
$data = json_encode($arr);
$response->write('{"total":"' . $zeilen . '","results":"' . $data . '"');

} else $response->write("{success:false}");

}
?>

```

```

<?php
include_once("Interfaces/ICommand.php");

class cmdGetRights implements ICommand {

public function execute(IRequest $request, IResponse $response) {

//make db-connection
$db = new DbConnection();

//check if user has permissions to proceed
$user = new SecurityObject();

```

```

if ($user->checkCredentials($db, $request)== true) {
    $response->write('({{"total": "1", "results": [{"ID_User": "'.$_SESSION['ID_User'].'", "email": "'.$_SESSION['valid_user'].'", "username": "'.$_SESSION['username'].'", "right_user": "'.$_SESSION['right_user'].'", "right_anfrage": "'.$_SESSION['right_anfrage'].'", "right_buchung": "'.$_SESSION['right_buchung'].'", "right_leistung": "'.$_SESSION['right_leistung'].'", "right_package": "'.$_SESSION['right_package'].'", "right_partner": "'.$_SESSION['right_partner'].'", "right_auswertungen": "'.$_SESSION['right_auswertungen'].'", "right_kunde": "'.$_SESSION['right_kunde'].'", "right_jugendherbergen": "'.$_SESSION['right_jugendherbergen'].'"}]})');
}
?>

```

## 9.4 cmdHandle\_XY

```

<?php
include_once("Interfaces/ICommand.php");

class cmdHandleJhbPack implements ICommand {

    public function execute(IRequest $request, IResponse $response) {

        //make db-connection
        $db = new DbConnection();

        //check if user has permissions to proceed
        $user = new SecurityObject();
        if ($user->checkCredentials($db, $request)== true) {

            $id_jhb = $db->real_escape_string($request->getParameter('ID_Jhb'));
            $id_package = $db->real_escape_string($request->getParameter('ID_Package'));
            $oldText = $db->real_escape_string($request->getParameter('oldText'));
            $oldValue = $db->real_escape_string($request->getParameter('oldValue'));

            if($request->isSetParameter('remove')) {

                $SQL = "DELETE FROM tJhb_Package WHERE (ID_Package = $id_package AND ID_Jhb = $id_jhb)";
                $db->query($SQL);
                $response->write("{success:true}");

            } else {

                if($oldText == 'neues Package') {
                    //Wenn es ein neues Package ist, muss es erst in der Tabelle angelegt werden!!
                    $SQL = "CALL procInsertJHBPack('$id_jhb', '$id_package')";
                } else {
                    //Wenn es aber KEIN neues Package ist, muss der betreffende Datensatz nur upgedatet werden!
                    $SQL = "UPDATE tJhb_Package SET ID_Package = $id_package WHERE (ID_Package = $oldValue AND ID_Jhb = $id_jhb)";
                }

                //Einfügen in die Tabelle
                $db->query($SQL);

                //die beeinflussten Zeilen checken um festzustellen ob Erfolg oder Misserfolg.
                $zeilen = $db->affected_rows();
                if($zeilen > 0) $response->write("{success:true}");
                else $response->write("{success:false}");
            }
        }
    }
}

```

```

        } else $response->write("{success:false}");
    }
?>

```

```

<?php
include_once("Interfaces/ICommand.php");

class cmdHandlePackLeistung implements ICommand {

    public function execute(IRequest $request, IResponse $response) {

        //make db-connection
        $db = new DbConnection();

        //check if user has permissions to proceed
        $user = new SecurityObject();
        if ($user->checkCredentials($db, $request)== true) {

            $id_package = $db->real_escape_string($request->getParameter('ID_Package'));
            $id_leistung = $db->real_escape_string($request->getParameter('ID_Leistung'));
            $leistungsTag = $db->real_escape_string($request->getParameter('leistungstag'));
            $oldText = $db->real_escape_string($request->getParameter('oldText'));
            $oldValue = $db->real_escape_string($request->getParameter('oldValue'));

            if($request->issetParameter('remove')) {

                $SQL = "DELETE FROM tPackageleistung WHERE (ID_Package = $id_package AND ID_Leistung = $id_leistung)";
                $db->query($SQL);
                $response->write("{success:true}");

            } else {

                if($oldText == 'neue Leistung') {
                    //Wenn es eine neue Leistung ist, muss es erst in der Tabelle angelegt werden!
                    $SQL = "CALL procInsertPackLeistung('$id_package', '$id_leistung')";
                } else {
                    //Wenn es aber KEINE neue Leistung ist, muss der betreffende Datensatz nur upgedated werden!
                    $SQL = "UPDATE tPackageleistung SET ID_Leistung = $id_leistung WHERE (ID_Leistung = $oldValue AND
ID_Package = $id_package)";

                }

                //Einfügen in die Tabelle
                $db->query($SQL);

                //die beeinflussten Zeilen checken um festzustellen ob Erfolg oder Misserfolg.
                $zeilen = $db->affected_rows();
                if($zeilen > 0) $response->write("{success:true}");
                else $response->write("{success:false}");
            }
        } else $response->write("{success:false}");
    }
?>

```

## 9.5 cmdInsert\_XY

```

<?php
include_once("Interfaces/ICommand.php");

class cmdInsertAnfrage implements ICommand {

    public function execute(IRequest $request, IResponse $response) {

```

```

//make db-connection
$db = new DbConnection();

//Befüllen der benötigten Variablen für den Insert
$katgorie = $request->getParameter('katgorie');
$organisation = $db->real_escape_string($request->getParameter('organisation'));
$ID_Package = (integer) $request->getParameter('ID_Package');
$stermin = $request->getParameter('termin');
$ersatztermin = $request->getParameter('ersatztermin');
$kinder = (integer) $db->real_escape_string($request->getParameter('kinder'));
$erwachsene = (integer) $db->real_escape_string($request->getParameter('erwachsene'));
$female = (integer) $db->real_escape_string($request->getParameter('female'));
$male = (integer) $db->real_escape_string($request->getParameter('male'));
$vegetarier = (integer) $db->real_escape_string($request->getParameter('vegetarier'));
$relVorschriften = $db->real_escape_string($request->getParameter('relVorschriften'));
$allergien = $db->real_escape_string($request->getParameter('allergien'));
$abgefrKnr = $db->real_escape_string($request->getParameter('abgefrKnr'));
$adresse = $db->real_escape_string($request->getParameter('adresse'));
$plz = $db->real_escape_string($request->getParameter('plz'));
$ort = $db->real_escape_string($request->getParameter('ort'));
$tel = $db->real_escape_string($request->getParameter('tel'));
$fax = $db->real_escape_string($request->getParameter('fax'));
$email = $db->real_escape_string($request->getParameter('email'));
$vorname = $db->real_escape_string($request->getParameter('vorname'));
$nachname = $db->real_escape_string($request->getParameter('nachname'));

//die IP-Adresse des anlegenden Hosts wird durch eine PHP-Funktion ermittelt...
$ipadr = $_SERVER['REMOTE_ADDR'];
$telAP = $db->real_escape_string($request->getParameter('telAP'));
$emailAP = $db->real_escape_string($request->getParameter('emailAP'));
//der User wird aus der Session ermittelt und die zugehörige User-ID wird aus der Datenbank rückgefragt...
$ID_User = $_SESSION['ID_User'];
$letzteBearbeitung = date("Y-m-d");
$bemerkung = $db->real_escape_string($request->getParameter('bemerkung'));
$erstelltAm = date("Y-m-d");

//Zusammenbauen des SQL-Statements (Stored Procedure)
$SQL = "CALL proclInsertAnfrage('$katgorie', '$organisation', '$ID_Package', '$stermin', '$ersatztermin', ";
$SQL .= "'$kinder', '$erwachsene', $female, $male, '$vegetarier', '$relVorschriften', '$allergien', '$abgefrKnr', '$adresse', '$plz',";
'$ort', '$tel', ";
$SQL .= "'$fax', '$email', '$vorname', '$nachname', '$ipadr', '$telAP', '$emailAP', '".$ID_User[0]."', '$letzteBearbeitung',";
'$bemerkung', '$erstelltAm')";

//abfeuern des SQL-Strings mittels mysql_query
$db->query($SQL);
//die beeinflussten Zeilen checken um festzustellen ob Erfolg oder Misserfolg.
$zeilen = $db->affected_rows();

if($zeilen > 0)
    $response->write("{success:true}");
else
    throw new DBSQLException("Der Datensatz konnte nicht eingefügt werden, bitte prüfen Sie den SQL-Ausdruck.");
}

?>

```

```

<?php
include_once("Interfaces/ICommand.php");

class cmdInsertAnsprechperson implements ICommand {

    public function execute(IRequest $request, IResponse $response) {

```

```

//make db-connection
$db = new DbConnection();

//check if user has permissions to proceed
$user = new SecurityObject();
if ($user->checkCredentials($db, $request)== true) {

    $ID_Kunde = $request->getParameter('ID_Kunde');
    $Nachname = $db->real_escape_string($request->getParameter('nachname'));
    $Vorname = $db->real_escape_string($request->getParameter('vorname'));
    $Telefon = $db->real_escape_string($request->getParameter('telefon'));
    $Email = $db->real_escape_string($request->getParameter('email'));
    $Bemerkung = $db->real_escape_string($request->getParameter('bemerkung'));

    $ID_User = $_SESSION['ID_User'];
    $LetzteBearbeitung = date("Y-m-d");

    //Zusammenbauen des SQL-Statements (Stored Procedure)
    $SQL = "CALL proclnsertAnsprechperson('$ID_Kunde', '$Nachname', '$Vorname', '$Telefon', ";
    $SQL .= "'$Email', '$ID_User', '$LetzteBearbeitung', '$Bemerkung')";

    //abfeuern des SQL-Strings mittels mysql_query
    $db->query($SQL);
    //die beeinflussten Zeilen checken um festzustellen ob Erfolg oder Misserfolg.
    $zeilen = $db->affected_rows();

    if($zeilen > 0) {
        //Auf diesem Wege können wir die ID des neu eingefügten Datensatzes ermitteln und an die Client-Applikation zur weiteren Verarbeitung übermitteln.
        $neueAnsprechperson = $db->query("SELECT LAST_INSERT_ID() as neueAnsprechperson")->fetch_object()->neueAnsprechperson;
        $response->write("{success:true, neueAnsprechperson:$neueAnsprechperson}");
    }
    else
        $response->write("{success:false}");
    } else $response->write("{success:false}");
}
}
?>

```

```

<?php
include_once("Interfaces/ICommand.php");

class cmdInsertBuchung implements ICommand {

    public function execute(IRequest $request, IResponse $response) {

        //make db-connection
        $db = new DbConnection();

        //check if user has permissions to proceed
        $user = new SecurityObject();
        if ($user->checkCredentials($db, $request)== true) {
            $resyBuchungsNr = $request->getParameter('resyBuchungsNr');
            $sterminAnreise = $db->real_escape_string($request->getParameter('terminAnreise'));
            $ersatzTerminAnreise = $db->real_escape_string($request->getParameter('ersatzTerminAnreise'));
            $kinder = $request->getParameter('kinder');
            $erwachsene = $request->getParameter('erwachsene');
            $female = $request->getParameter('female');
            $male = $request->getParameter('male');
            $vegetarier = $request->getParameter('vegetarier');
            $relVorschriften = $db->real_escape_string($request->getParameter('relVorschriften'));
            $allergien = $db->real_escape_string($request->getParameter('allergien'));
            $erstellitAm = $db->real_escape_string($request->getParameter('erstellitAm'));
            $buchungsStatus = $db->real_escape_string($request->getParameter('buchungsStatus'));
        }
    }
}

```

```

$ID_Anprechperson = $request->getParameter('ID_Anprechperson');

//der User wird aus der Session ermittelt
$ID_User = $_SESSION['ID_User'];
$letzteBearbeitung = date("Y-m-d");

if($request->isSetParameter('ID_Package')) {
    $ID_Package = $request->getParameter('ID_Package');
}
else {
    $ID_Anfrage = $request->getParameter('ID_Anfrage');
    $PackSQL = "SELECT ID_Package from tAnfrage where ID_Anfrage = $ID_Anfrage";
    $ID_Package = $db->query($PackSQL)->fetch_object()->ID_Package;
}

***** STARTEN DER TRANSAKTION: *****/
$db->query('begin');

//Zusammenbauen des SQL-Statements (Stored Procedure)
$SQL = "CALL procInsertBuchung('$resyBuchungsNr','$ID_Package', '$terminAnreise', '$ersatzTerminAnreise', '$kinder',
'$erwachsene', '$female', '$male', '$vegetarier', ";
$SQL .= "'$relVorschriften', '$allergien', '$ID_User', '$letzteBearbeitung', '$erstelltAm',
'$buchungsStatus', '$ID_Anprechperson')";

//abfeuern des SQL-Strings mittels mysql_query
$db->query($SQL);
//die beeinflussten Zeilen checken um festzustellen ob Erfolg oder Misserfolg.
$zeilen = $db->affected_rows();

//In die Tabelle tLeistungsZeitpunkt müssen jetzt noch die Leistungen des gebuchten Packages gespeichert werden:
$SQL = "SELECT tLeistung.ID_Leistung as IDL, tLeistung.StandardUhrzeit as ZEIT, tPackageleistung.leistungstag as TAG
FROM tPackageleistung ";
$SQL .= "INNER JOIN tLeistung ON (tPackageleistung.ID_Leistung = tLeistung.ID_Leistung) WHERE
tPackageleistung.ID_Package = $ID_Package";
$ergebnis = $db->query($SQL);

//die ID der neuen Buchung über die Datenbank ermitteln
$ID_Buchung = $db->query("SELECT LAST_INSERT_ID() as neueBuchung")->fetch_object()->neueBuchung;

//für jeden Datensatz der SQL-SELECT Anweisung wird ein eigenes Insert-StoreProc aufgerufen
while($zeile = $ergebnis->fetch_object()) {

    $ScheitDatum = date("Y-m-d", strtotime("+." . ($zeile->TAG-1) . " day", strtotime("$terminAnreise")));
    $db->query("CALL procInsertLeistungsZeitpunkt('$ID_Buchung', '$zeile->IDL', '$zeile->ZEIT', '$ScheitDatum')");
}

$db->query('commit');
***** ENDE DER TRANSAKTION *****

if($zeilen > 0) {
    $response->write("{success:true}");
}
else
    $response->write("{success:false}");
} else $response->write("{success:false}");
}
?>

```

```

<?php
include_once("Interfaces/ICommand.php");

class cmdInsertBuchungsHinweis implements ICommand {

```

```

public function execute(IRequest $request, IResponse $response) {

    //make db-connection
    $db = new DbConnection();

    //check if user has permissions to proceed
    $user = new SecurityObject();
    if ($user->checkCredentials($db, $request)== true) {

        //Befüllen der benötigten Variablen für den Insert
        $ID_Buchung = (integer) $request->getParameter('ID_Buchung');
        $infodatum = date("Y-m-d");
        $infotext = $db->real_escape_string($request->getParameter('infotext'));
        $ID_User = $_SESSION['ID_User'];

        //Zusammenbauen des SQL-Statements (Stored Procedure)
        $SQL = "CALL procInsertBuchungsHinweis('$ID_Buchung', '$infodatum', '$infotext', '$ID_User')";

        //abfeuern des SQL-Strings mittels mysql_query
        $db->query($SQL);

        //die beeinflussten Zeilen checken um festzustellen ob Erfolg oder Misserfolg.
        $zeilen = $db->affected_rows();

        if($zeilen > 0) $response->write("{success:true}");
        else $response->write("{success:false}");

    } else $response->write("{success:false}");
}
?>

```

```

<?php
include_once("Interfaces/ICommand.php");

class cmdInsertJhb implements ICommand {

    public function execute(IRequest $request, IResponse $response) {

        //make db-connection
        $db = new DbConnection();

        //check if user has permissions to proceed
        $user = new SecurityObject();
        if ($user->checkCredentials($db, $request)== true) {

            $jhb = $db->real_escape_string($request->getParameter('jhb'));
            $Adresse = $db->real_escape_string($request->getParameter('Adresse'));
            $plz = $db->real_escape_string($request->getParameter('plz'));
            $ort = $db->real_escape_string($request->getParameter('ort'));

            //Zusammenbau für das Stored Prcedures
            $SQL = "CALL procInsertJHB('$jhb', '$Adresse', '$plz', '$ort')";

            //Einfügen in die Tabelle
            $db->query($SQL);

            //die beeinflussten Zeilen checken um festzustellen ob Erfolg oder Misserfolg.
            $zeilen = $db->affected_rows();

            if($zeilen > 0) $response->write("{success:true}");
            else $response->write("{success:false}");

        } else $response->write("{success:false}");
}

```

```
}
```

```
<?php
include_once("Interfaces/ICommand.php");

class cmdInsertJhbPack implements ICommand {

public function execute(IRequest $request, IResponse $response) {

    //make db-connection
    $db = new DbConnection();

    //check if user has permissions to proceed
    $user = new SecurityObject();
    if ($user->checkCredentials($db, $request)== true) {

        $id_jhb = $db->real_escape_string($request->getParameter('ID_Jhb'));
        $id_package = $db->real_escape_string($request->getParameter('ID_Package'));

        //Zusammenbau für das Stored Procedures
        $SQL = "CALL procInsertJHPack('$id_jhb', '$id_package')";

        //Einfügen in die Tabelle
        $db->query($SQL);

        //die beeinflussten Zeilen checken um festzustellen ob Erfolg oder Misserfolg.
        $zeilen = $db->affected_rows();

        if($zeilen > 0) $response->write("{success:true}");
        else $response->write("{success:false}");

    } else $response->write("{success:false}");
}
}

?>
```

```
<?php
include_once("Interfaces/ICommand.php");

class cmdInsertKunden implements ICommand {

public function execute(IRequest $request, IResponse $response) {

    //make db-connection
    $db = new DbConnection();

    //check if user has permissions to proceed
    $user = new SecurityObject();
    if ($user->checkCredentials($db, $request)== true) {

        $kategorie = $request->getParameter('kategorie');
        $organisation = $request->getParameter('organisation');
        $Adresse = $db->real_escape_string($request->getParameter('adresse'));
        $plz = $db->real_escape_string($request->getParameter('plz'));
        $ort = $db->real_escape_string($request->getParameter('ort'));
        $telefon = $db->real_escape_string($request->getParameter('telefon'));
        $fax = $db->real_escape_string($request->getParameter('fax'));
        $email = $db->real_escape_string($request->getParameter('email'));
        $resykd = $db->real_escape_string($request->getParameter('resykd'));
        $Bemerkung = $db->real_escape_string($request->getParameter('bemerkung'));

    }
}
```

```
//die IP-Adresse des anlegenden Hosts wird durch eine PHP-Funktion ermittelt...
$ipadr = $_SERVER['REMOTE_ADDR'];

$ID_User = $_SESSION['ID_User'];
$letzteBearbeitung = date("Y-m-d");
$bemerkung = $db->real_escape_string($request->getParameter('bemerkung'));
$erstelltAm = date("Y-m-d");

//Zusammenbauen des SQL-Statements (Stored Procedure)
$SQL = "CALL procInsertKunde('$kategorie', '$organisation', '$adresse', '$plz', ";
$SQL .= "'$ort', '$telefon', '$fax', '$email', '$resykd', ";
$SQL .= "'$bemerkung', '$ID_User[0]', '$letzteBearbeitung', '$erstelltAm')";

//abfeuern des SQL-Strings mittels mysql_query
$db->query($SQL);
//die beeinflussten Zeilen checken um festzustellen ob Erfolg oder Misserfolg.
$zeilen = $db->affected_rows();

if($zeilen > 0) {
    $neueID = $db->query("SELECT LAST_INSERT_ID() as neueID")->fetch_object()->neueID;
    $response->write("{success:true, neueID:$neueID}");
}
else
    $response->write("{success:false}");
} else
    $response->write("{success:false}");
}
}

?>
```

```
<?php
include_once("Interfaces/ICommand.php");

class cmdInsertLeistung implements ICommand {

    public function execute(IRequest $request, IResponse $response) {

        //make db-connection
        $db = new DbConnection();

        //check if user has permissions to proceed
        $user = new SecurityObject();
        if ($user->checkCredentials($db, $request)== true) {

            $Leistung = $request->getParameter('Leistung');
            $StandardUhrzeit = $request->getParameter('StandardUhrzeit');
            $LeistungsBemerkung = $db->real_escape_string($request->getParameter('LeistungsBemerkung'));
            $ID_Partner = (integer) $request->getParameter('ID_Partner');

            //Zusammenbauen des SQL-Statements (Stored Procedure)
            $SQL = "CALL procInsertLeistung('$Leistung', '$StandardUhrzeit', '$LeistungsBemerkung', ";
            $SQL .= "'$ID_Partner')";

            //abfeuern des SQL-Strings mittels mysql_query
            $db->query($SQL);

            //die beeinflussten Zeilen checken um festzustellen ob Erfolg oder Misserfolg.
            $zeilen = $db->affected_rows();

            if($zeilen > 0)
                $response->write("{success:true}");
            else
                $response->write("{success:false}");
        } else
            $response->write("{success:false}");
    }
}
```

```
}
```

```
?>
```

```
<?php
include_once("Interfaces/ICommand.php");

class cmdInsertPackage implements ICommand {

    public function execute(IRequest $request, IResponse $response) {

        //make db-connection
        $db = new DbConnection();

        //check if user has permissions to proceed
        $user = new SecurityObject();
        if ($user->checkCredentials($db, $request)== true) {

            $packagename = $db->real_escape_string($request->getParameter('packagename'));
            $pdfPfad = $db->real_escape_string($request->getParameter('pdfPfad'));
            $ID_User = $_SESSION['ID_User'];
            $letzteBearbeitung = date("Y-m-d");

            //Zusammenbau für das Stored Procedures
            $SQL = "CALL procInsertPackage('$packagename', '$pdfPfad', '".$ID_User[0]."', '$letzteBearbeitung')";

            //Einfügen in die Tabelle
            $db->query($SQL);

            //die beeinflussten Zeilen checken um festzustellen ob Erfolg oder Misserfolg.
            $zeilen = $db->affected_rows();

            if($zeilen > 0) $response->write("{success:true}");
            else $response->write("{success:false}");

        } else $response->write("{success:false}");
    }
}
?>
```

```
<?php
include_once("Interfaces/ICommand.php");

class cmdInsertPartner implements ICommand {

    public function execute(IRequest $request, IResponse $response) {

        //make db-connection
        $db = new DbConnection();

        //check if user has permissions to proceed
        $user = new SecurityObject();
        if ($user->checkCredentials($db, $request)== true) {

            //Befüllen der benötigten Variablen für den Insert
            $ID_Partner = (integer) $request->getParameter('ID_Partner');
            $firmenname = $db->real_escape_string($request->getParameter('firmenname'));
            $vorname = $db->real_escape_string($request->getParameter('vorname'));
            $nachname = $db->real_escape_string($request->getParameter('nachname'));
            $adresse = $db->real_escape_string($request->getParameter('adresse'));
            $plz = $db->real_escape_string($request->getParameter('plz'));
            $ort = $db->real_escape_string($request->getParameter('ort'));
            $tel = $db->real_escape_string($request->getParameter('tel'));
            $email = $db->real_escape_string($request->getParameter('email'));

        }
    }
}
```

```

$ID_User = $_SESSION['ID_User'];
$letzteBearbeitung = date("Y-m-d");

//Zusammenbauen des SQL-Statements (Stored Procedure)
$SQL = "CALL procInsertPartner('$ID_Partner', '$firmenname', '$vorname', '$nachname', ";
$SQL .= "'$adresse', '$plz', '$ort', '$tel', '$email', '".$ID_User[0]."', '$letzteBearbeitung')";

//abfeuern des SQL-Strings mittels mysql_query
$db->query($SQL);

//die beeinflussten Zeilen checken um festzustellen ob Erfolg oder Misserfolg.
$zeilen = $db->affected_rows();

if($zeilen > 0) $response->write("{success:true}");
else $response->write("{success:false}");

} else $response->write("{success:false}");
}

?>

```

```

<?php
include_once("Interfaces/ICommand.php");

class cmdInsertUser implements ICommand {

public function execute(IRequest $request, IResponse $response) {

//make db-connection
$db = new DbConnection();

//check if user has permissions to proceed
$user = new SecurityObject();
if ($user->checkCredentials($db, $request)== true) {

$adresse = $db->real_escape_string($request->getParameter('adresse'));
$email = $db->real_escape_string($request->getParameter('email'));
$nachname = $db->real_escape_string($request->getParameter('nachname'));
$vorname = $db->real_escape_string($request->getParameter('vorname'));
$ort = $db->real_escape_string($request->getParameter('ort'));
$plz = $db->real_escape_string($request->getParameter('plz'));

//Zusammenbauen des SQL-Statements (Stored Procedere)
$SQL = "CALL procInsertUser('$adresse', '$email', '$nachname', '$vorname', '$ort', '$plz', md5('init'))";

//abfeuern des SQL-Strings mittels mysql_query
$db->query($SQL);

//die beeinflussten Zeilen checken um festzustellen ob Erfolg oder Misserfolg.
$zeilen = $db->affected_rows();

if($zeilen > 0) $response->write("{success:true}");
else $response->write("{success:false}");

} else $response->write("{success:false}");
}

}
?>

```

## 9.6 cmdLogout

```
<?php
```

```

include_once("Interfaces/ICommand.php");

class cmdLogout implements ICommand {
    //Vernichtet die Session und setzt alle angelegten Session-Variablen (valid_user, rights) auf null.
    public function execute(IRequest $request, IResponse $response) {
        //Zerstören aller Session-Variablen:
        $_SESSION = array();
        //Zersöten der Session-Cookies:
        if(ini_get("session.use_cookies")) {
            $params = session_get_cookie_params();
            setcookie(session_name(), " ", time() - 42000, $params["path"], $params["domain"], $params["secure"],
            $params["httponly"]);
        }
        //Zerstören der Session:
        session_destroy();
        //Rückmeldung ob alles erledigt:
        $response->write("{success:true}");
    }
}
?>

```

## 9.7 cmdMail

```

<?php
include_once("Interfaces/ICommand.php");
include("Classes/PHPMailer/class.phpmailer.php");

class cmdMail implements ICommand {

    public function execute(IRequest $request, IResponse $response) {

        //make db-connection
        $db = new DbConnection();

        //check if user has permissions to proceed
        // $user = new SecurityObject();

        //MAILVERSAND WIRD NICHT ÜBER BENUTZERSICHERHEIT GESCHÜTZT, DA AUCH VOM FRONTEND BENÜTZBAR SEIN SOLL
        //if ($user->checkCredentials($db, $request)== true) {

            if ( session_is_registered( "valid_user" ) ) $from = $_SESSION['valid_user'];
            else $from = "smartbooking@noejhw.at";

            $to = $request->getParameter('to');

            $mail = new PHPMailer();

            $mail->IsSMTP(); // set mailer to use SMTP
            $mail->Host = "zeus.axelsoft.at"; // specify main and backup server
            $mail->SMTPSecure = "SSL";
            $mail->Port = 587;
            $mail->SMTPAuth = true; // turn on SMTP authentication
            $mail->Username = "smartbooking@noejhw.at"; // SMTP username
            $mail->Password = "her@cules"; // SMTP password

            $mail->From = $from; //do NOT fake header.
            $mail->FromName = "Sachbearbeiter NOEJHW";
            $mail->AddAddress($to);
            $mail->AddReplyTo($from, "Support and Help"); //optional

            $mail->Subject = $request->getParameter('subject');
            $mail->Body = $request->getParameter('body');

            if(!$mail->Send())
            {
                throw new Exception($mail->ErrorInfo);
            }
        }
    }
}
?>

```

```

        }else{
            $response->write("{success:true}");
        }
    //}
}
?>

```

## 9.8 cmdUpdate

```

<?php
include_once("Interfaces/ICommand.php");

class cmdUpdate implements ICommand {

    public function execute(IRequest $request, IResponse $response) {

        //make db-connection
        $db = new DbConnection();

        //check if user has permissions to proceed
        $user = new SecurityObject();
        if ($user->checkCredentials($db, $request)== true) {

            $table = $request->getParameter('table');
            $key = $request->getParameter('key');
            $id = (integer)$db->real_escape_string($request->getParameter('keyID'));
            $field = $request->getParameter('field');
            $value = $db->real_escape_string($request->getParameter('value'));
            $now = date("Y-m-d");

            //welcher User ist momentan als 'valider User angemeldet...
            $userid = $_SESSION['ID_User'];
            //Der tatsächliche Update der DB mit den neuen Werten.
            if(is_string($request->getParameter('value'))AND $value != 'true' AND $value != 'false') {
                $db->query("UPDATE ".$table.' SET '.$field.'='.$value.' WHERE '.$key.'='.$id);
            }
            else {
                $db->query("UPDATE ".$table.' SET '.$field.'='.$value.' WHERE '.$key.'='.$id);
            }

            $rows = $db->affected_rows();

            if($rows > 0) {
                //ID des angemeldeten Users und das aktuelle Datum zu diesem Datensatz dazuspeichern.
                if ($table == "tAnsprechperson" || $table == "tKunde" || $table == "tAnfrage" || $table == "tPackage" || $table == "tPartner" || $table == "tBuchung") {
                    $db->query("UPDATE ".$table." SET ID_User = $userid WHERE ".$key.'='.$id);
                    $db->query("UPDATE ".$table.' SET letzteBearbeitung = \''.$now.'\' WHERE '.$key.'='.$id);
                    $response->write("{success:true}");
                } else {
                    $response->write("{success:true}");
                }
            } else throw new DBSQLException("Kein Datensatz wurde upgedated. Bitte prüfen Sie den SQL-Ausdruck!");
        } else $response->write("{success:false}");
    }
}
?>

```

```

<?php
include_once("Interfaces/ICommand.php");

class cmdUpdatePackLeistung implements ICommand {

```

```

public function execute(IRequest $request, IResponse $response) {

    //make db-connection
    $db = new DbConnection();

    //check if user has permissions to proceed
    $user = new SecurityObject();
    if ($user->checkCredentials($db, $request)== true) {

        $table = $request->getParameter('table');
        $key1 = $request->getParameter('key1');
        $key2 = $request->getParameter('key2');
        $id1 = (integer)$db->real_escape_string($request->getParameter('keyID1'));
        $id2 = (integer)$db->real_escape_string($request->getParameter('keyID2'));
        $field = $request->getParameter('field');
        $value = $db->real_escape_string($request->getParameter('value'));

        //Der tatsächliche Update der DB mit den neuen Werten.
        $db->query ('UPDATE '.$table.' SET '.$field.'= \''. $value. '\' WHERE '.$key1.'='.$id1. ' AND '.$key2.'='.$id2);

        $rows = $db->affected_rows();

        if($rows > 0) $response->write("{success:true}");
        else $response->write("{success:false}");

        } else $response->write("{success:false}");
    }
}
?>

```

```

<?php
include_once("Interfaces/ICommand.php");

class cmdUpdateUser implements ICommand {

    public function execute(IRequest $request, IResponse $response) {

        //make db-connection
        $db = new DbConnection();

        //check if user has permissions to proceed
        $user = new SecurityObject();
        if ($user->checkCredentials($db, $request)== true) {

            //DIESES COMMAND STELLT DIE ÄNDERUNG DER MAILADRESSE SOWIE DIE ÄNDERUNG DES PASSWORTES
            //FÜR DEN USER ZUR VERFÜGUNG. ZUERST WIRD ABER MIT PASSWORTABFRAGE GEPRÜFT, OB DER USER AUCH
            //BERECHTIGT IST, ÄNDERUNGEN VORZUNEHMEN

            //welcher User ist momentan als valider User angemeldet...
            $userid = $_SESSION['ID_User'];
            //das Passwort aus der DB besorgen und mit dem eingegebenen Passwort abgleichen
            $pwdcheck = $db->query('SELECT passwort from tUser WHERE ID_User =' . $userid) ->fetch_object();
            if(md5($request->getParameter('pwd')) == $pwdcheck->passwort)
            {
                $field = $request->getParameter('field');
                $value = $db->real_escape_string($request->getParameter('value'));

                if ($request->issetParameter('newPwd')) {
                    $value = md5($value);
                } else $mailinsession = true;

                //Der tatsächliche Update der DB mit den neuen Werten.
                //echo 'UPDATE tUser SET '.$field.'= \''. $value. '\' WHERE ID_User =' . $userid;
                $db->query('UPDATE tUser SET '.$field.'= \''. $value. '\' WHERE ID_User =' . $userid);
                $rows = $db->affected_rows();
            }
        }
    }
}
?>

```

```

if($mailinsession == true) {
    $_SESSION['valid_user'] = $value;
}

if($rows > 0) $response->write("{success:true}");
else $response->write("{success:false, pwd:true}");
} else $response->write("{success:false,pwd:false}");

} else $response->write("{success:false}");
}
?>

```

## 10 Applikations-Register (Tabs)

Nachfolgend steht der Code für die Registerkarten, die per Druck auf einen Navigationsbutton erzeugt werden und in das Main-Tabpanel eingehängt werden.

### 10.1 Anfragen

```

*****-----Funktion, um einen Tab zum Mainbereich zu adden.-----*****
Ich habe mich für diese Variante entschieden, da somit das Initiieren des Programmes flotter geht.
Beim Instanziieren eines neuen Tabs für das Center-Panel wird der jeweilige Programmteil nachgeladen.
Ich bin der Meinung das diese Herangehensweise den Workflow des Users am wenigsten beeinträchtigt.
Technisch wäre es auch möglich, alle Tabs beim Ext.onReady() Event zu laden. Man hätte dadurch den Vorteil
einer verkürzten Zeit während des Betriebes. Dafür dauert das initiiierende Laden des Programmes um ein viel-
faches länger und nicht benötigte Programmteile werden gleich von Beginn an immer mitgeladen.
*****-----/*****
```

```

//addAnfrageTab() erstellt ein Ticketsystem für die eingehenden Anfragen der potentiellen Kunden
//technisch wird ein FormPanel erstellt, indem sich (grob umrissen) ein GridPanel im Center und ein Fielset im Osten
//des Layouts (Border-Layout) befinden. Dieses FormPanel wird in das TabPanel "tabcenter" als neuer Tab eingehängt.
//Den größten Zeilaufwand nehmen hier die FieldSets mit den vielen Steuerelementen (Textfields, Combo-Boxes, Dateiels...)
//ein.
function addAnfrageTab(){
    //zuerst gehört gecheckt, ob es den Anfragentab nicht vielleicht schon gibt.
    //wenn ja, wird er nur mehr focusiert, wenn nein werden die Grids, Forms und Stores erstellt
    //und alles läuft wie geschmiert :)
    if(document.getElementById('anfrageGridForm') == null) {

        // create the data store
        var ticketstore = new Ext.data.Store({
            proxy: new Ext.data.HttpProxy({
                url: 'index.php',
                method: 'POST'
            }),
            baseParams:{
                cmd: "FillAnfrageGrid",
                archiv: false
            },
            remoteSort:false,
            sortInfo: {field:'ID_Anfrage', direction:'DESC'},
            reader: new Ext.data.JsonReader({
                root: 'results',
                totalProperty: 'total'
            },Ext.data.Record.create([
                {name: 'ID_Anfrage', type:'int'},
                {name: 'kategorie', type:'string'},

```

```

        {name: 'organisation', type:'string'},
        {name: 'packagename', type:'string'},
        {name: 'termin', type: 'date', dateFormat: 'Y-m-d'},
        {name: 'ersatztermin', type: 'date', dateFormat: 'Y-m-d'},
        {name: 'kinder', type:'int'},
        {name: 'erwachsene', type:'int'},
        {name: 'female', type:'int'},
        {name: 'male', type:'int'},
        {name: 'vegetarier', type:'int'},
        {name: 'relVorschriften', type:'string'},
        {name: 'allergien', type:'string'},
        {name: 'abgefrKnr', type:'string'},
        {name: 'adresse', type:'string'},
        {name: 'plz', type:'string'},
        {name: 'ort', type:'string'},
        {name: 'tel', type:'string'},
        {name: 'fax', type:'string'},
        {name: 'email', type:'string'},
        {name: 'vorname', type:'string'},
        {name: 'nachname', type:'string'},
        {name: 'ipadr', type:'string'},
        {name: 'telAP', type:'string'},
        {name: 'emailAP', type:'string'},
        {name: 'uebernommen', type:'boolean'},
        {name: 'abgelehnt', type:'boolean'},
        {name: 'username', type:'string'},
        {name: 'letzteBearbeitung', type:'date', dateFormat: 'Y-m-d'},
        {name: 'bemerkung', type:'string'},
        {name: 'erstelltAm', type: 'date', dateFormat: 'Y-m-d'}
    })
}
});
//das Polling nach neuen Datensätzen soll so lange true sein, bis ein Datensatz in Bearbeitung steht. Während dieser
Bearbeitung
//wird diese Eigenschaft auf false gesetzt und somit ein reload des stores verhindert.
ticketstore._pollEnabled = true;
ticketstore.addListener('update',function(st,rec,op) {
    //Hier brauchen wir einen kleinen Trick den ich (Gott sei Dank)
    //aus dem Netz der Netze geklaut habe! Wir müssen ja für jedes upgedatede Feld
    //einen Ajax-Update-Call machen. Ich hatte aber von rec.getChanges() nur ein Objekt
    //indem ALLE geänderten Werte gespeichert waren. Diese kurze aber echt geile Schleife
    //löst mein Problem und setzt für jede Property des Objects einen AJAX-Update-Call ab
    //mit den geänderten Werten! Genial einfach, einfach genial!!
    var obj = eval(rec.getChanges());
    if (typeof(obj) == "object") {
        for (var j in obj) { //das ist die Schleife aus dem I-Netz! TOLLE SACHE!
            var fieldwert = j;
            if(j == "packagename") { //In der Grid ist das Package ein String,
                fieldwert = "ID_Package"; //Darstellen tun wir einen String, updaten eine ID.
            }
            Ext.Ajax.request({ //hier wird der AJAX-Call abgesetzt.
                waitTitle:'Connecting',
                waitMsg:'Sending data...',
                url: 'index.php',
                params: {
                    cmd:"Update",
                    key: 'ID_Anfrage',
                    keyID: rec.data.ID_Anfrage,
                    table:"tAnfrage",
                    field: fieldwert,
                    value: obj[j]
                },
                failure:function(response,options){
                    Ext.Msg.alert('Fehler', 'Server wurde nicht erreicht! Bitte versuchen Sie es später nochmals.');
                },
                success:function(response,options){
                    var responseJSON = Ext.decode(response.responseText);
                }
            });
        }
    }
});

```

```

        if(responseJSON.error==true) { //FEHLERBEHANDLUNG
            handleException(responseJSON);
        } else {
            if(responseJSON.success == true) {
                ticketstore.reload();
                ticketstore.commitChanges();
                Ext.example.msg('Status', 'Updated successfully!');
            } else Ext.example.msg('Fehler!', 'Leider kein Zugriff');
        }
    }
});

//custom renderer für Erstellt am: Soll das Datum rot angeben, wenn sei dem Erstellen min. 2 Tage vergangen sind und noch
//nicht bearbeitet wurde.
function urgent(value, metaData, record, rowIndex, colIndex, store){
    var ONE_DAY = 1000 * 60 * 60 * 24; //ein Tag in Millisek.
    //Beide Werte in Millisek. umrechnen
    var date1_ms = new Date().getTime();
    var date2_ms = value.getTime();
    // Differenz in Millisekunden
    var difference_ms = date1_ms - date2_ms
    // Auf Tage zurückrechnen
    var difference = Math.round(difference_ms/ONE_DAY)

    if(difference > 2 && record.data.abgelehnt==false && record.data.uebernommen==false)
        {metaData.css = 'redboldtext';}
    return Ext.util.Format.date(value,'d.m.Y');
}

// create the Grid
var anfrageGrid = new Ext.grid.GridPanel({
    store: ticketstore,
    columns: [
        {header: "Nr", width: 65, sortable: true, dataIndex: 'ID_Anfrage'},
        {header: "Eingelangt am", renderer: urgent, width: 95, sortable: true, dataIndex: 'erstelltAm'},
        {header: "Organisation", width: 135, sortable: true, dataIndex: 'organisation'},
        {header: "Nachname", width: 115, sortable: true, dataIndex: 'nachname'},
        {header: "Package", width: 115, sortable: true, dataIndex: 'packagename'},
        {header: "Termin", id:"Termin", width: 95, sortable: true, xtype: 'datecolumn', format: 'd.m.Y', dataIndex: 'termin'}
    ],
    stripeRows: true,
    viewConfig: {
        forceFit:true
    },
    layout:'fit',
    region:'center',
    sm: new Ext.grid.RowSelectionModel({
        singleSelect:true,
        listeners: {
            rowselect: function(sm, row, rec) {
                Ext.getCmp("anfrageGridForm").getForm().loadRecord(rec);
            }
        }
    }),
    //Mit diesem Listener auf den "render-Event" könnten wir den ersten Datensatz
    //während des Renderns der Grid auswählen lassen.
    listeners: {
        render: function(g) {
            g.getSelectionModel().selectRow(0);
        },
        delay: 10
    },
    bbar: new Ext.PagingToolbar({

```

```

    pageSize: 10,
    store: ticketstore,
    displayInfo: true,
    plugins: new Ext.ux.ProgressBarPager()
},
//die Top-Toolbar der Grid, hier sind die Buttons "neue Anfrage",
//"/"Anfrage übernehmen", "abgeschlossene Anfragen anzeigen" und die Suchleiste zuhause.
tbar: new Ext.Toolbar({
    items: [{
        tooltip:'Datensatz anzeigen',
        iconCls:'lesen',
        id:'AnfrageShowButton',
        handler:function() {
            var rec = anfrageGrid.getSelectionModel().getSelected();
            if(rec)
            {
                //Datensatz Anzeigen - 1. Fieldset (Buchungsdaten)
                var showAnfrFS1 = new Ext.form.FieldSet({
                    labelAlign: 'left',
                    labelWidth: 150,
                    layout:'form',
                    defaults: {width: 200},
                    bodyStyle:'padding:10px',
                    defaultType: 'textfield',
                    title:'Buchungsdaten',
                    height: 'auto',
                    border: false,
                    items:[{
                        fieldLabel: 'Anfrage-ID',
                        disabled:true,
                        readOnly:true,
                        name: 'ID_Anfrage'
                    },{
                        fieldLabel: 'IP Adresse',
                        readOnly:true,
                        name: 'ipadr'
                    },{
                        fieldLabel: 'Kategorie',
                        readOnly:true,
                        name: 'kategorie'
                    },{
                        fieldLabel: 'Package',
                        readOnly:true,
                        name: 'packagename'
                    },{
                        fieldLabel: 'Termin',
                        readOnly:true,
                        hideTrigger:true,
                        format: 'd.m.Y',
                        xtype:'datefield',
                        name: 'termin'
                    },{
                        fieldLabel: 'Ersatz-Termin',
                        readOnly:true,
                        hideTrigger:true,
                        format: 'd.m.Y',
                        xtype:'datefield',
                        name: 'ersatztermin'
                    },{
                        fieldLabel: 'Kinder',
                        readOnly:true,
                        name: 'kinder'
                    },{
                        fieldLabel: 'Erwachsene',
                        readOnly:true,
                        name: 'erwachsene'
                    }]
                })
            }
        }
    }]
})
}

```

```

        fieldLabel: 'Weiblich',
        readOnly:true,
        name: 'female'
    },
    fieldLabel: 'Männlich',
    readOnly:true,
    name: 'male'
},
fieldLabel: 'Vegetarier',
readOnly:true,
name: 'vegetarier'
},
fieldLabel: 'Kundennr.',
readOnly:true,
name: 'abgefrKnr'
}
});
};

//Datensatz Anzeigen - 2. Fieldset (Personendaten)
var showAnfrFS2 = new Ext.form.FieldSet({
    labelAlign: 'left',
    labelWidth: 150,
    layout:'form',
    defaults: {width: 200},
    defaultType: 'textfield',
    bodyStyle:'padding:10px',
    title:'Personendaten',
    height: 'auto',
    border: false,
    items:[{ fieldLabel: 'Schulen- /Firmenname',
        readOnly:true,
        name: 'organisation'
    },
    fieldLabel: 'Vorname',
    readOnly:true,
    name: 'vorname'
},
    fieldLabel: 'Nachname',
    readOnly:true,
    name: 'nachname'
},
    fieldLabel: 'Adresse',
    readOnly:true,
    name: 'adresse'
},
    fieldLabel: 'PLZ',
    readOnly:true,
    name: 'plz'
},
    fieldLabel: 'Ort',
    readOnly:true,
    name: 'ort'
},
    fieldLabel: 'Telefon',
    readOnly:true,
    name: 'tel'
},
    fieldLabel: 'Fax',
    readOnly:true,
    name: 'fax'
},
    fieldLabel: 'eMail',
    readOnly:true,
    name: 'email'
},
    fieldLabel: 'Telefon pers.',
    readOnly:true,
    name: 'telefonPers'
}]
});
```

```

        name: 'telAP'
    },{  

        fieldLabel: 'eMail pers.',  

        readOnly:true,  

        name: 'emailAP'  

    }]  

});  
  

//Datensatz Anzeigen - 3. Fieldset (Sonstige Daten)
var showAnfrFS3 = new Ext.form.FieldSet({  

    labelAlign: 'left',  

    labelWidth: 150,  

    layout:'form',  

    title:'Sonstiges',  

    bodyStyle:'padding:10px',  

    defaults: {width: 200},  

    defaultType: 'textfield',  

    height: 'auto',  

    border: false,  

    items:[{  

        fieldLabel: 'Allergien',  

        xtype: 'textarea',  

        readOnly:true,  

        name: 'allergien'  

    },{  

        fieldLabel: 'religiöse Vorschriften',  

        xtype: 'textarea',  

        readOnly:true,  

        name: 'relVorschriften'  

    },{  

        fieldLabel: 'Übernommen',  

        readOnly:true,  

        disabled:true,  

        xtype: 'checkbox',  

        name: 'uebernommen'  

    },{  

        fieldLabel: 'Abgelehnt',  

        readOnly:true,  

        disabled:true,  

        xtype: 'checkbox',  

        name: 'abgelehnt'  

    },{  

        fieldLabel: 'Erstellt am',  

        readOnly:true,  

        hideTrigger:true,  

        disabled:true,  

        format: 'd.m.Y',  

        xtype:'datefield',  

        name: 'erstelltAm'  

    },{  

        fieldLabel: 'Zuletzt bearbeitet von',  

        disabled:true,  

        readOnly:true,  

        name: 'username'  

    },{  

        fieldLabel: 'Zuletzt bearbeitet am',  

        readOnly:true,  

        hideTrigger:true,  

        disabled:true,  

        format: 'd.m.Y',  

        xtype:'datefield',  

        name: 'letzteBearbeitung'  

    },{  

        fieldLabel: 'Bemerkung',  

        xtype: 'textarea',  

        readOnly:true,  

        name: 'bemerkung'  

    }]
});

```

```

        });
    });

    var showAnfrFP = new Ext.form.FormPanel({
        labelAlign: 'left',
        id:'ticketForm',
        items: [{
            xtype:'tabpanel',
            activeTab: 0,
            defaults:{autoHeight:true},
            items:[showAnfrFS1, showAnfrFS2, showAnfrFS3]
        }],
        buttons: [{text:'Ok',
            handler:function() {win.close();}}]
    });

    var win = new Ext.Window({
        title:'Anfrage-Datensatz anzeigen',
        closable:true,
        width:420,
        border:true,
        resizable:false,
        modal:true,
        items: [showAnfrFP]
    });
    showAnfrFP.getForm().loadRecord(anfrageGrid.getSelectionModel().getSelected());
    win.show(this);
} else {Ext.example.msg('Fehler', 'Bitte wählen Sie einen Datensatz aus!');}
},
width:50
},{
    tooltip:'Datensatz ändern',
    iconCls:'schreiben',
    handler:function() {
        //das Polling des Stores wird ausgeschalten, solange bis die Bearbeitung des Datensatzes abgeschlossen ist.
        ticketstore._pollEnabled = false;
        //der in der Grid selektierte Record aus dem Store wird vollständig in der Variable rec gespeichert, und in das FormPanel zum Bearbeiten geladen.
        var rec = anfrageGrid.getSelectionModel().getSelected();
        //wenn diese if-Abfrage false ergibt ist kein Datensatz in der Grid selektiert und es wird ein entsprechender Hinweis ausgegeben.
        if(rec)
        {
            //Datensatz Ändern - 1. Fieldset (Buchungsdaten)
            var writeAnfrFS1 = new Ext.form.FieldSet({
                labelAlign: 'left',
                labelWidth: 150,
                layout:'form',
                defaults: {width: 200},
                bodyStyle:'padding:10px',
                defaultType: 'textfield',
                title:'Buchungsdaten',
                height: 'auto',
                border: false,
                items:[{
                    fieldLabel: 'Anfrage-ID',
                    readOnly:true,
                    name: 'ID_Anfrage'
                },{
                    fieldLabel: 'IP Adresse',
                    readOnly:true,
                    name: 'ipadr'
                },{
                    fieldLabel: 'Kategorie*',
                    xtype:'combo',
                    allowBlank:false,
                    editable:false,

```

```

        forceSelection:true,
        triggerAction:'all',
        mode: 'local',
        store: new Ext.data.ArrayStore({
            fields: ['KategorieText'],
            data: [['Privat'],['Schule'],['Firma']]
        }),
        valueField: 'KategorieText',
        displayField: 'KategorieText',
        name: 'kategorie'
    },
    fieldLabel: 'Package*',
    xtype:'combo',
    allowBlank:false,
    editable:false,
    forceSelection:true,
    triggerAction:'all',
    mode: 'local',
    store: new Ext.data.Store({
        proxy: new Ext.data.HttpProxy({
            url: 'index.php',
            method: 'POST'
        }),
        reader: new Ext.data.JsonReader({
            root: 'results',
            totalProperty: 'total',
            id: 'ID_Package'
        },Ext.data.Record.create([
            {name: 'ID_Package', type: 'int'},
            {name: 'packagename', type: 'string'}
        ])),
        baseParams:{},
        start: 0,
        cmd: "GetPackage"
    },
    autoLoad: true
}),
valueField: 'ID_Package',
displayField: 'packagename',
name: 'packagename'
},
fieldLabel: 'Termin*',
allowBlank:false,
format: 'd.m.Y',
// minValue: new Date(),
xtype:'datefield',
name: 'termin'
},
fieldLabel: 'Ersatz-Termin',
format: 'd.m.Y',
// minValue: new Date(),
xtype:'datefield',
name: 'ersatztermin'
},
fieldLabel: 'Kinder',
name: 'kinder'
},
fieldLabel: 'Erwachsene',
name: 'erwachsene'
},
fieldLabel: 'Weiblich',
name: 'female'
},
fieldLabel: 'Männlich',
name: 'male'
},
fieldLabel: 'Vegetarier',

```

```

        name: 'vegetarier'
    },{  

        fieldLabel: 'Kundennr.',  

        name: 'abgefrKnr'  

    }  

});  
  

//Datensatz Ändern - 2. Fieldset (Personendaten)  

var writeAnfrFS2 = new Ext.form.FieldSet({  

    labelAlign: 'left',  

    labelWidth: 150,  

    layout:'form',  

    defaults: {width: 200},  

    defaultType: 'textfield',  

    bodyStyle:'padding:10px',  

    title:'Personendaten',  

    height: 'auto',  

    border: false,  

    items:[{ fieldLabel:'Schulen- /Firmenname',  

        name: 'organisation'  

    },{  

        fieldLabel: 'Vorname*',  

        allowBlank:false,  

        name: 'vorname'  

    },{  

        fieldLabel: 'Nachname*',  

        allowBlank:false,  

        name: 'nachname'  

    },{  

        fieldLabel: 'Adresse*',  

        allowBlank:false,  

        name: 'adresse'  

    },{  

        fieldLabel: 'PLZ*',  

        allowBlank:false,  

        name: 'plz'  

    },{  

        fieldLabel: 'Ort*',  

        allowBlank:false,  

        name: 'ort'  

    },{  

        fieldLabel: 'Telefon',  

        name: 'tel'  

    },{  

        fieldLabel: 'Fax',  

        name: 'fax'  

    },{  

        fieldLabel: 'eMail',  

        vtype:'email',  

        name: 'email'  

    },{  

        fieldLabel: 'Tel. Ansprechpartner*',  

        allowBlank:false,  

        name: 'telAP'  

    },{  

        fieldLabel: 'eMail Ansprechpartner*',  

        allowBlank:false,  

        vtype:'email',  

        name: 'emailAP'  

    }]  

});  
  

//Datensatz Ändern - 3. Fieldset (Sonstige Daten)  

var writeAnfrFS3 = new Ext.form.FieldSet({  

    labelAlign: 'left',  

    labelWidth: 150,  

    layout:'form',  


```

```

        title:'Sonstiges',
        bodyStyle:'padding:10px',
        defaults: {width: 200},
        defaultType: 'textfield',
        height: 'auto',
        border: false,
        items:[{
            fieldLabel: 'Allergien',
            xtype: 'textarea',
            name: 'allergien'
        },{
            fieldLabel: 'religiöse Vorschriften',
            xtype: 'textarea',
            name: 'relVorschriften'
        },{
            fieldLabel: 'Übernommen',
            xtype: 'checkbox',
            name: 'uebernommen'
        },{
            fieldLabel: 'Abgelehnt',
            xtype: 'checkbox',
            name: 'abgelehnt'
        },{
            fieldLabel: 'Zuletzt bearbeitet von',
            readOnly:true,
            name: 'username'
        },{
            fieldLabel: 'Zuletzt bearbeitet am',
            readOnly:true,
            hideTrigger:true,
            format: 'd.m.Y',
            xtype:'datefield',
            name: 'letzteBearbeitung'
        },{
            fieldLabel: 'Bemerkung',
            xtype: 'textarea',
            name: 'bemerkung'
        }]
    });
}

var writeAnfrFP = new Ext.form.FormPanel({
    labelAlign: 'left',
    id:'ticketForm',
    monitorValid:true,
    items: [{  

        xtype:'tabpanel',
        activeTab: 0,
        defaults:{autoHeight:true},
        items:[writeAnfrFS1, writeAnfrFS2, writeAnfrFS3]
    }],
    buttons: [{ text:'save',
        iconCls:'save',
        formBind: true,
        handler: function() {
            writeAnfrFP.getForm().updateRecord(anfrageGrid.getSelectionModel().getSelected());
            //das Polling für den Store wieder aktivieren...
            ticketstore._pollEnabled = true;
            win.close();
        }
    },
    {  

        text: 'cancel',
        handler: function() {
            //das Polling für den Store wieder aktivieren...
            ticketstore._pollEnabled = true;
            win.close();
        }
    }]
});

```

```

        }
    ];
});

var win = new Ext.Window({
    title:'Anfrage-Datensatz ändern',
    closable:true,
    width:420,
    border:true,
    resizable:false,
    modal:true,
    items: [writeAnfrFP]
});
writeAnfrFP.getForm().loadRecord(anfrageGrid.getSelectionModel().getSelected());
win.show(this);
} else {Ext.example.msg('Fehler', 'Bitte wählen Sie einen Datensatz aus!');}
},
width:50
},{
tooltip:'Neuen Datensatz anlegen',
iconCls:'add',
handler: function() {
//Datensatz NEU - 1. Fieldset (Buchungsdaten)
var neuAnfrFS1 = new Ext.form.FieldSet({
    labelAlign: 'left',
    labelWidth: 150,
    layout:'form',
    defaults: {width: 200},
    bodyStyle:'padding:10px',
    defaultType: 'textfield',
    title:'Buchungsdaten',
    height: 'auto',
    border: false,
    items:[{
        fieldLabel: 'Kategorie*',
        xtype:'combo',
        allowBlank:false,
        editable:false,
        forceSelection:true,
        triggerAction:'all',
        mode: 'local',
        store: new Ext.data.ArrayStore({
            fields: ['KategorieText'],
            data: [['Privat'],['Schule'],['Firma']]
        }),
        valueField: 'KategorieText',
        displayField: 'KategorieText',
        name: 'kategorie'
    },{
        fieldLabel: 'Package*',
        xtype:'combo',
        allowBlank:false,
        editable:false,
        id:'PackageldCombo',
        forceSelection:true,
        triggerAction:'all',
        mode: 'local',
        store: new Ext.data.Store({
            proxy: new Ext.data.HttpProxy({
                url: 'index.php',
                method: 'POST'
            }),
            reader: new Ext.data.JsonReader({
                root: 'results',
                totalProperty: 'total',
                id: 'ID_Package'
            },Ext.data.Record.create([

```

```

        {name: 'ID_Package', type: 'int'},
        {name: 'packagename'}
    ]}),
    baseParams:{
        start: 0,
        cmd: "GetPackage"
    },
    autoLoad: true
}),
valueField: 'ID_Package',
displayField: 'packagename',
hiddenName: 'ID_Package',
hiddenId:'hiddenPackageIdCombo',
name:'ID_Package'
},
fieldLabel: 'Termin*',
allowBlank:false,
format: 'd.m.Y',
minValue: new Date(),
xtype:'xdatefield',
name: 'termin'
},
fieldLabel: 'Ersatz-Termin',
format: 'd.m.Y',
minValue: new Date(),
xtype:'xdatefield',
name: 'ersatztermin'
},
fieldLabel: 'Kinder',
name: 'kinder'
},
fieldLabel: 'Erwachsene',
name: 'erwachsene'
},
fieldLabel: 'Weiblich',
name: 'female'
},
fieldLabel: 'Männlich',
name: 'male'
},
fieldLabel: 'Vegetarier',
name: 'vegetarier'
},
fieldLabel: 'Kundennr.',
name: 'abgefrKnr'
}
});
};

//Datensatz NEU - 2. Fieldset (Personendaten)
var neuAnfrFS2 = new Ext.form.FieldSet({
labelAlign: 'left',
labelWidth: 150,
layout:'form',
defaults: {width: 200},
defaultType: 'textfield',
bodyStyle:'padding:10px',
title:'Personendaten',
height: 'auto',
border: false,
items:[{
    fieldLabel:'Schulen- /Firmenname',
    name: 'organisation'
},
{
    fieldLabel: 'Vorname*',
    allowBlank:false,
    name: 'vorname'
},
{
    fieldLabel: 'Nachname*',
    allowBlank:false,
    name: 'nachname'
}
]
});

```

```

        allowBlank:false,
        name: 'nachname'
    },
    fieldLabel: 'Adresse*',
    allowBlank:false,
    name: 'adresse'
},
fieldLabel: 'PLZ*',
allowBlank:false,
name: 'plz'
},
fieldLabel: 'Ort*',
allowBlank:false,
name: 'ort'
},
fieldLabel: 'Telefon',
name: 'tel'
},
fieldLabel: 'Fax',
name: 'fax'
},
fieldLabel: 'eMail',
vtype:'email',
name: 'email'
},
fieldLabel: 'Tel. Ansprechperson*',
allowBlank:false,
name: 'telAP'
},
fieldLabel: 'eMail Ansprechperson*',
allowBlank:false,
vtype:'email',
name: 'emailAP'
}
});
}

//Datensatz NEU - 3. Fieldset (Sonstige Daten)
var neuAnfrFS3 = new Ext.form.FieldSet({
    labelAlign: 'left',
    labelWidth: 150,
    layout:'form',
    title:'Sonstiges',
    bodyStyle:'padding:10px',
    defaults: {width: 200},
    defaultType: 'textfield',
    height: 'auto',
    border: false,
    items:[{
        fieldLabel: 'Allergien',
        xtype: 'textarea',
        name: 'allergien'
    },
    fieldLabel: 'religiöse Vorschriften',
        xtype: 'textarea',
        name: 'relVorschriften'
    },
    fieldLabel: 'Übernommen',
        xtype: 'checkbox',
        name: 'uebernommen'
    },
    fieldLabel: 'Abgelehnt',
        xtype: 'checkbox',
        name: 'abgelehnt'
    },
    fieldLabel: 'Bemerkung',
        xtype: 'textarea',
        name: 'bemerkung'
    }
});

```

```

        name: 'bemerkung'
    });
});

var neuAnfrFP = new Ext.form.FormPanel({
    labelAlign: 'left',
    id:'ticketForm',
    monitorValid:true,
    url:'index.php',
    items: [
        {
            xtype:'tabpanel',
            activeTab: 0,
            defaults:{autoHeight:true},
            items:[neuAnfrFS1, neuAnfrFS2, neuAnfrFS3]
        },
        buttons: [
            {
                text: 'Save',
                formBind: true,
                iconCls:'save',
                handler: function(){
                    neuAnfrFP.getForm().submit({
                        method:'POST',
                        waitTitle:'Connecting',
                        waitMsg:'Sending data...',
                        params: {
                            cmd:'InsertAnfrage'
                        },
                        success:function(form, action){
                            if(action.result.success == true) {
                                Ext.example.msg('Status', 'Saved successfully!');
                                ticketstore.reload();
                            } else {
                                Ext.example.msg('Fehler', 'Nicht gespeichert!!');
                            }
                            win.close();
                        },
                        failure:function(form, action){
                            if(action.failureType == 'server'){
                                Ext.example.msg('Not Saved. Server-Error Occured.');
                            }else{
                                Ext.example.msg('Warning!', 'Server is unreachable at the Moment: ' +
action.response.responseText);
                            }
                            win.close();
                        }
                    });
                }
            },
            {
                text: 'Cancel',
                handler: function() {win.close();}
            }
        ]);
};

var win = new Ext.Window({
    title:'Anfrage-Datensatz hinzufügen',
    closable:true,
    width:420,
    border:true,
    resizable:false,
    modal:true,
    items: [neuAnfrFP]
});
win.show(this);
},
width:50
},{
tooltip:'Datensatz löschen',

```

```

handler: function() {
    var rec = anfrageGrid.getSelectionModel().getSelected();
    if(rec)
    {
        Ext.Msg.show({
            title:'wirklich löschen?',
            msg: 'Sie sind dabei, eine Anfrage zu löschen. Wollen Sie diese Anfrage wirklich löschen?',
            buttons: Ext.Msg.YESNO,
            fn: function(buttonID) {
                if(buttonID=='yes') {
                    Ext.Ajax.request({ //hier wird der AJAX-Call abgesetzt.
                        waitTitle:'Connecting',
                        waitMsg:'updating data...',
                        url: 'index.php',
                        params: {
                            cmd:"Update",
                            key: 'ID_Anfrage',
                            keyID: anfrageGrid.getSelectionModel().getSelected().data.ID_Anfrage,
                            table:'tAnfrage',
                            field: 'aktiv',
                            value: 0
                        },
                        failure:function(response,options){
                            Ext.example.msg('Status', 'Not deleted! Error Occured!');
                        },
                        success:function(response,options){
                            var responseData = Ext.util.JSON.decode(response.responseText);
                            if(responseData.success == true) {
                                ticketstore.reload();
                                ticketstore.commitChanges();
                                Ext.example.msg('Status', 'Deleted successfully!');
                            } else {
                                Ext.example.msg('Fehler', 'Keine Berechtigung');
                            }
                        }
                    });
                    ticketstore.reload();
                }
            },
            icon: Ext.MessageBox.QUESTION
        });
    } else {Ext.example.msg('Fehler', 'Bitte wählen Sie einen Datensatz aus!');}
},
iconCls:'delete',
width:50
},
tooltip:'Kunde und Buchung übernehmen',
iconCls:'grant',
handler: function() {

    var rec = anfrageGrid.getSelectionModel().getSelected(); //in diese Variable kommt im Success-Callback des Ansprechpersonen-Ajax-Requests die gewählte
    ID_Ansprechperson
    var AP_ID = null;

    if (rec)
    {
        //STORES
        var SuggestionStore = new Ext.data.Store({
            proxy: new Ext.data.HttpProxy({
                url: 'index.php',
                method: 'POST'
            }),
            remoteSort:true,
            baseParams:{
                cmd: "GetMergeSuggestions",
                alle: false,
            }
        });
    }
}
}

```

```

plz:rec.data.plz,
name:rec.data.organisation,
resykd:rec.data.abgefrKnr
},
sortInfo: {field:'ID_Kunde', direction:'ASC'},
reader: new Ext.data.JsonReader({
    root: 'results',
    totalProperty: 'total'
},Ext.data.Record.create([
    {name: 'ID_Kunde', type:'int'},
    {name: 'kategoriek', type:'string'},
    {name: 'organisation', type:'string'},
    {name: 'adresse', type:'string'},
    {name: 'plz', type:'string'}
])
)
});
SuggestionStore.load({params:{start:0, limit:8}});

var AnsprechpersonenStore = new Ext.data.Store{
    proxy: new Ext.data.HttpProxy({
        url: 'index.php',
        method: 'POST'
}),
    remoteSort:true,
    baseParams:{
        cmd: "FillAnsprechpersonenGrid"
},
    sortInfo: {field:'ID_Ansprechperson', direction:'ASC'},
    reader: new Ext.data.JsonReader({
        root: 'results',
        totalProperty: 'total'
},Ext.data.Record.create([
    {name: 'ID_Ansprechperson', type:'int'},
    {name: 'nachname', type:'string'},
    {name: 'vorname', type:'string'},
    {name: 'email', type:'string'}
])
)
};
//GRIDS
var SuggestionGrid = new Ext.grid.GridPanel({
    title:'Vorschläge für bereits bestehende Kunden',
    id:'SuggestionGrid',
    store: SuggestionStore,
    columns: [
        {header: "Kategorie", width: 95, sortable: false, dataIndex: 'kategoriek'},
        {header: "Organisation", width: 125, sortable: false, dataIndex: 'organisation'},
        {header: "PLZ", width: 75, sortable: false, dataIndex: 'plz'},
        {header: "Adresse", width: 135, sortable: false, dataIndex: 'adresse'}
    ],
    stripeRows: true,
    columnWidth: .50,
    viewConfig: {
        forceFit:true
    },
    tbar: new Ext.Toolbar({
        items:[{xtype:'button',
            id:'btnAlle',
            handler: function(b) {
                SuggestionStore.setBaseParam('alle', Ext.getCmp('btnAlle').pressed);
                SuggestionStore.load({params:{start:0, limit:8}});
            },
            enableToggle: true,
            width: 50,
        }]
    })
}
);

```

```

        tooltip:'alle',
        text:'alle Kunden'
    }]
}),
bbar: new Ext.PagingToolbar({
    pageSize: 8,
    store: SuggestionStore,
    displayInfo: true
}),
height:260,
sm: new Ext.grid.RowSelectionModel({
    singleSelect:true
})
});

var AnsprechpersonenGrid = new Ext.grid.GridPanel({
    title:'bestehende Ansprechpersonen',
    id:'AnsprechpersonenGrid',
    store: AnsprechpersonenStore,
    columns: [
        {header: "Nachname", width: 110, sortable: false, dataIndex: 'nachname'},
        {header: "Vorname", width: 135, sortable: false, dataIndex: 'vorname'},
        {header: "eMail", width: 90, sortable: false, dataIndex: 'email'}
    ],
    stripeRows: true,
    columnWidth: .50,
    height:300,
    viewConfig: {
        forceFit:true
    },
    sm: new Ext.grid.RowSelectionModel({
        singleSelect:true
    })
});

//FORMS
var kundenForm = new Ext.form.FormPanel({
    id:'uebernehmenFormKunde',
    defaultType: 'textfield',
    title:'neuer Kunde',
    columnWidth: .50,
    bodyStyle:'padding:10px',
    defaults: {
        width:150
    },
    labelAlign: 'left',
    items:[{
        fieldLabel: 'Kategorie',
        readOnly:true,
        name: 'kategorie'
    },{
        fieldLabel: 'Firma',
        readOnly:true,
        name: 'organisation'
    },{
        fieldLabel: 'Nachname',
        readOnly:true,
        name: 'nachname'
    },{
        fieldLabel: 'Adresse',
        readOnly:true,
        name: 'adresse'
    },{
        fieldLabel: 'PLZ',
        readOnly:true,
        name: 'plz'
    }]
});

```

```

        fieldLabel: 'Ort',
        readOnly:true,
        name: 'ort'
    },{
        fieldLabel: 'Telefon',
        readOnly:true,
        name: 'tel'
    },{
        fieldLabel: 'Fax',
        readOnly:true,
        name: 'fax'
    },{
        fieldLabel: 'eMail',
        readOnly:true,
        name: 'email'
    }],
    buttons:[{
        text:'Neuer Kunde',
        handler: function() {
            Ext.Ajax.request({           //hier wird der AJAX-Call für den Kunden abgesetzt.
                waitTitle:'Connecting',
                waitMsg:'updating data...',
                url: 'index.php',
                params: {
                    cmd:"InsertKunden",
                    kategorie:rec.data.kategorie,
                    organisation:rec.data.organisation,
                    adresse:rec.data.adresse,
                    plz:rec.data.plz,
                    ort:rec.data.ort,
                    telefon:rec.data.tel,
                    fax:rec.data.fax,
                    email:rec.data.email,
                    resykD:rec.data.resykD,
                    bemerkung:rec.data.bemerkung
                },
                failure:function(response,options){
                    Ext.example.msg('Neuer Kunde', 'Nicht übernommen! Error Occured!');
                },
                success:function(response,options){
                    var responseData = Ext.util.JSON.decode(response.responseText);
                    if(responseData.success == true) {
                        Ext.example.msg('Neuer Kunde', 'Kunde erfolgreich angelegt!');
                    }
                }
            });
        }
    }];
    Ext.Ajax.request({           //hier wird der AJAX-Call für die Ansprechperson
        waitTitle:'Connecting',
        waitMsg:'updating data...',
        url: 'index.php',
        params: {
            cmd:"InsertAnsprechperson",
            ID_Kunde:responseData.neuID,
            nachname:rec.data.nachname,
            vorname:rec.data.vorname,
            telefon:rec.data.telAP,
            email:rec.data.emailAP
        },
        failure:function(response,options){
            Ext.example.msg('Neuer Kunde', 'Nicht übernommen! Error
        },
        success:function(response,options){
            var responseData = Ext.util.JSON.decode(response.responseText);
            if(responseData.success == true) {
                Ext.example.msg('Neue Ansprechperson', 'Ansprechperson
            }
        }
    });
    BuchungsForm.enable();
}

```

abgesetzt.

Occured!');

erfolgreich angelegt!');

```

AP_ID = responseData.neueAnsprechperson;
Ext.getCmp("BuchungsForm").getForm().loadRecord(rec);
UebernehmenTabpanel.setActiveTab(2);
kundenCombination.disable();
} else {
    Ext.example.msg('Ansprechperson', 'Keine Berechtigung für
Ansprechpersonen!');
}
}
});
} else {
    Ext.example.msg('Kunden', 'Keine Berechtigung für Kunden!');
}
}
});
}

}, {
text:'Bestehender Kunde',
handler: function() {
if(SuggestionGrid.getSelectionModel().hasSelection() == false) {
    Ext.example.msg('Kein Vorschlag ausgewählt', 'Bitte wählen Sie einen der Vorschläge
aus!');
}
} else {
    Ext.example.msg('Mergen', 'Kunde erfolgreich zugewiesen!');
    AnsprechpersonenCombination.enable();
}

AnsprechpersonenStore.load({params:{ID_Kunde:SuggestionGrid.getSelectionModel().getSelected().data.ID_Kunde}});
AnsprechpersonenForm.getForm().loadRecord(rec);
UebernehmenTabpanel.setActiveTab(1);
kundenCombination.disable();
}
}
});
}

var AnsprechpersonenForm = new Ext.form.FormPanel({
id:'AnsprechpersonForm',
defaultType: 'textfield',
title:'neue Ansprechperson',
columnWidth: .50,
bodyStyle:'padding:10px',
defaults: {
width:150
},
labelAlign: 'left',
items:[{
fieldLabel: 'Nachname',
readOnly:true,
name: 'nachname'
},{

fieldLabel: 'Vorname',
readOnly:true,
name: 'vorname'
},{

fieldLabel: 'eMail',
readOnly:true,
name: 'email'
},{

fieldLabel: 'Telefon',
readOnly:true,
name: 'telefon'
}],
buttons:[{
text:'Neue Ansprechperson',
handler: function() {
}
}]
});

```

```

Ext.Ajax.request({
    //hier wird der AJAX-Call abgesetzt.
    waitTitle:'Connecting',
    waitMsg:'updating data...!',
    url: 'index.php',
    params: {
        cmd:"InsertAnsprechperson",
        ID_Kunde:SuggestionGrid.getSelectionModel().getSelected().data.ID_Kunde,
        nachname:rec.data.nachname,
        vorname:rec.data.vorname,
        telefon:rec.data.telAP,
        email:rec.data.emailAP
    },
    failure:function(response,options){
        Ext.example.msg('Neue Ansprechperson', 'Nicht übernommen! Error Occured!');
    },
    success:function(response,options){
        var responseData = Ext.util.JSON.decode(response.responseText);
        if(responseData.success == true) {
            Ext.example.msg('Neue Ansprechperson', 'Ansprechperson erfolgreich angelegt!');
            Ext.getCmp('BuchungsForm').enable();
            Ext.getCmp('BuchungsForm').getForm().loadRecord(rec);
            AP_ID = responseData.neueAnsprechperson;
            UebernehmenTabpanel.setActiveTab(2);
            AnsprechpersonenCombination.disable();
        } else {
            Ext.example.msg('Ansprechperson', 'Keine Berechtigung für Ansprechperson!');
        }
    });
});
};

text:'Bestehende Ansprechperson',
handler: function() {
    if(AnsprechpersonenGrid.getSelectionModel().hasSelection() == false) {
        Ext.example.msg('Keine Person ausgewählt', 'Bitte wählen Sie eine der Ansprechpersonen aus!');
    } else {
        Ext.example.msg('Ansprechperson', 'Ansprechperson erfolgreich gewählt');
        BuchungsForm.enable();
        Ext.getCmp('BuchungsForm').getForm().loadRecord(rec);
        AP_ID =
        AnsprechpersonenGrid.getSelectionModel().getSelected().data.ID_Ansprechperson;
        UebernehmenTabpanel.setActiveTab(2);
        AnsprechpersonenCombination.disable();
    }
}
});

var BuchungsFieldset1 = new Ext.form.FieldSet({
    labelAlign: 'left',
    layout:'form',
    columnWidth: .50,
    defaults: {width: 150},
    bodyStyle:'padding:10px',
    defaultType: 'textfield',
    border: false,
    items:[{
        fieldLabel: 'Package',
        readOnly:true,
        name: 'packagename'
    },{
        fieldLabel: 'Termin',
        hideTrigger:true,
        format: 'd.m.Y',
    }]
});

```

```

xtype:'datefield',
readOnly:true,
name: 'termin'
},
fieldLabel: 'Ersatz-Termin',
hideTrigger:true,
format: 'd.m.Y',
xtype:'datefield',
readOnly:true,
name: 'ersatztermin'
},
fieldLabel: 'Anzahl Kinder',
readOnly:true,
name: 'kinder'
},
fieldLabel: 'Erwachsene',
readOnly:true,
name: 'erwachsene'
},
fieldLabel: 'Weiblich',
readOnly:true,
name: 'female'
},
fieldLabel: 'Männlich',
readOnly:true,
name: 'male'
},
fieldLabel: 'Vegetarier',
readOnly:true,
name: 'vegetarier'
}],
});
};

var BuchungsFieldset2 = new Ext.form.FieldSet({
labelAlign: 'left',
layout:'form',
columnWidth: .50,
defaults: {width: 125},
bodyStyle:'padding:10px',
defaultType: 'textfield',
border: false,
items:[{
fieldLabel: 'Rel.Vorschriften',
readOnly:true,
xtype:'textarea',
name: 'relVorschriften'
},
fieldLabel: 'Allergien',
readOnly:true,
xtype:'textarea',
name: 'allergien'
},
fieldLabel: 'Bemerkung',
readOnly:true,
xtype:'textarea',
name: 'bemerkung'
}],
});

var BuchungsForm = new Ext.form.FormPanel({
disabled:true,
id:'BuchungsForm',
title:'Buchung',
layout:'column',
items:[BuchungsFieldset1,BuchungsFieldset2],
buttons:[{

```

```

text:'Buchung übernehmen',
handler: function() {
    Ext.Ajax.request({
        waitTitle:'Connecting',
        waitMsg:'updating data...',
        url: 'index.php',
        params: {
            ID_Anfrage:rec.data.ID_Anfrage,
            terminAnreise:rec.data.termin,
            ersatzTerminAnreise:rec.data.ersatztermin,
            kinder:rec.data.kinder,
            erwachsene:rec.data.erwachsene,
            female:rec.data.female,
            male:rec.data.male,
            vegetarier:rec.data.vegetarier,
            relVorschriften:rec.data.relVorschriften,
            allergien:rec.data.allergien,
            erstelltAm:rec.data.erstelltAm,
            buchungsStatus:'offen',
            ID_Ansprechperson:AP_ID,
            cmd:"InsertBuchung"
        },
        failure:function(response,options){
            Ext.example.msg('Buchung', 'Nicht übernommen! Error Occured!');
        },
        success:function(response,options){
            var responseData = Ext.util.JSON.decode(response.responseText);
            if(responseData.success == true) {
                Ext.example.msg('Buchung', 'Buchung erfolgreich gespeichert');
                //Falls die Buchung richtig übernommen wurde, müssen wir auch das Ticket
                kennzeichnen. 2. Ajax-Call
                Ext.Ajax.request{
                    waitTitle:'Connecting',
                    waitMsg:'updating data...',
                    url: 'index.php',
                    params: {
                        cmd:"Update",
                        key: 'ID_Anfrage',
                        keyID: rec.data.ID_Anfrage,
                        table:"Anfrage",
                        field: 'uebernommen',
                        value: 1
                    },
                    failure:function(response,options){
                        Ext.example.msg('Buchung', 'Nicht übernommen! Error Occured!');
                    },
                    success:function(response,options){
                        var responseData = Ext.util.JSON.decode(response.responseText);
                        if(responseData.success == true) {
                            ticketstore.reload();
                            ticketstore.commitChanges();
                            Ext.example.msg('Buchung', 'Anfrage erfolgreich
                            übernommen!');
                            win.close();
                        } else {
                            Ext.example.msg('Buchung', 'Keine Berechtigung für
                            Anfragen!');
                        }
                    }
                };
            } else {
                Ext.example.msg('Buchung', 'Keine Berechtigung für Buchung!');
            }
        }
    });
}
}

```

```

    });

//COMBINATIONS
var kundenCombination = new Ext.Panel({
    title:'Kunde',
    id:'panelKundeUebernehmen',
    layout:'column',
    items:[kundenForm, SuggestionGrid]
});

var AnsprechpersonenCombination = new Ext.Panel({
    title:'Ansprechperson',
    layout: 'column',
    disabled: true,
    items:[AnsprechpersonenForm, AnsprechpersonenGrid]
});

//TABPANEL
var UebernehmenTabpanel = new Ext.TabPanel({
    activeTab: 0,
    items:[kundenCombination, AnsprechpersonenCombination, BuchungsForm]
});

//WINDOW
var win = new Ext.Window({
    title:'Kunde / Ansprechperson / Buchung übernehmen',
    closable:true,
    width:600,
    layout:'fit',
    height: 380,
    frame:true,
    resizable:false,
    modal:true,
    items: [UebernehmenTabpanel]
});
Ext.getCmp('uebernehmenFormKunde').getForm().loadRecord(rec);
win.show(this);
} else {Ext.example.msg("Fehler", 'Bitte wählen Sie einen Datensatz aus!');}
},
width:50
},{
tooltip:'Anfrage ablehnen',
iconCls:'deny',
handler: function() {
    var rec = anfrageGrid.getSelectionModel().getSelected();
    if(rec)
    {
        Ext.Ajax.request({           //hier wird der AJAX-Call abgesetzt.
            waitTitle:'Connecting',
            waitMsg:'updating data...',
            url: 'index.php',
            params: {
                cmd:"Update",
                key: 'ID_Anfrage',
                keyID: anfrageGrid.getSelectionModel().getSelected().data.ID_Anfrage,
                table:"tAnfrage",
                field: 'abgelehnt',
                value: 1
            },
            failure:function(response,options){
                Ext.example.msg('Status', 'Error Occured!');
            },
            success:function(response,options){
                var responseData = Ext.util.JSON.decode(response.responseText);
                if(responseData.success == true) {
                    ticketstore.reload();
                    ticketstore.commitChanges();
                }
            }
        });
    }
}
}

```

```

        Ext.example.msg('Status', 'Anfrage abgelehnt!');
    } else {
        Ext.example.msg('Fehler', 'Keine Berechtigung');
    }
}
});
ticketstore.reload();
} else {Ext.example.msg('Fehler', 'Bitte wählen Sie einen Datensatz aus!');}
},
width:50
},{

tooltip:'Kunde Mail',
iconCls:'email',
id:'KundenMailButton',
width:50,
handler:function() {

if(anfrageGrid.getSelectionModel().hasSelection()) {

var MailFS = new Ext.form.FieldSet({
    labelAlign: 'left',
    labelWidth: 150,
    layout:'form',
    title:'Mail',
    bodyStyle:'padding:10px',
    defaults: {width: 200},
    defaultType: 'textfield',
    height: 'auto',
    border: false,
    items:[{
        fieldLabel: 'Betreff',
        xtype: 'textfield',
        name: 'subject'
    },{
        fieldLabel: 'Text',
        xtype: 'textarea',
        name: 'body'
    }]
});

var neuMailFP = new Ext.form.FormPanel({
    labelAlign: 'left',
    monitorValid:true,
    url:'index.php',
    items: [MailFS],
    buttons: [{
        text: 'Send',
        formBind: true,
        iconCls:'email',
        handler: function(){
            neuMailFP.getForm().submit({
                method:'POST',
                waitTitle:'Connecting',
                waitMsg:'Sending data...',
                params: {
                    cmd:'Mail',
                    to:anfrageGrid.getSelectionModel().getSelected().data.email
                },
                success:function(form, action) {
                    Ext.example.msg('Success', 'Mail erfolgreich gesendet!');
                    win.close();
                },
                failure:function(form, action) {
                    var responseJSON = Ext.decode(action.response.responseText);
                    handleException(responseJSON);
                }
            });
        }
    }]
});

```

```

        }
    },
    text: 'Cancel',
    handler: function() {win.close();}
}
});

var win = new Ext.Window({
    title:'Kunden-Mail',
    closable:true,
    width:420,
    border:true,
    resizable:false,
    modal:true,
    items: [neuMailFP]
});
win.show(this);
} else {
    Ext.example.msg('Fehler', 'Kein Datensatz ausgewählt.');
}
}

},'->','-',{
    xtype:'button',
    id:'barchiv',
    handler: function(b) {
        //Beim ersten Aufruf des Archiv-Buttons wird der Basisparameter "archiv" auf den Toogle-Status des
        //Buttons gesetzt. Beim initialen Laden des Stores ist dieser Button noch nicht erstellt, deswegen
        //diese Vorgehensweise.
        ticketstore.setBaseParam('archiv', Ext.getCmp('barchiv').pressed);
        ticketstore.load({params:{start:0, limit:10}});
    },
    enableToggle: true,
    width: 50,
    tooltip:'Archiv',
    iconCls:'archiv'
},
{
    xtype:'textfield',
    name:'searchfield',
    id:'anfragesuchfeld',
    enableKeyEvents:true,
    emptyText:'Anfrage suchen...'
}
});
});

//dieser Event wird bei jedem Keyup-Event des Suchfeldes ausgelöst. Wir machen nach jedem Buchstaben einen AJAX-
Request und
//schießen das Ergebnis dieses neuen Requests an den Store. Hoffentlich ist der Datenbankzugriff schnell genug damit sich
dieser geile
//moderne Effekt den ich mir vorstelle auch zu 100% entfaltet... --> wenn dieser Kommentar in der Endversion noch
enthalten ist, dann gehts :)
Ext.getCmp('anfragesuchfeld').addListerner('keyup',function(tf,e) {
    ticketstore.reload({params:{suchen:tf.getValue(),start:0, limit:10}});
});

anfrageGrid.getView().getRowClass = function(record, index){
    if (record.data.uebernommen == true) {
        return 'lightgreen';
    }
    if (record.data.abgelehnt == true) {
        return 'lightred';
    }
};

//Erstellen des Fieldsets, in dem die Daten angezeigt werden, die in der Grid
//ausgewählt sind.

```

```

var anfrageMainFieldset = new Ext.form.FieldSet({
    labelWidth: 90,
    region:'east',
    width: 230,
    margins: '15 0 0 0',
    defaults: {width: 110},
    defaultType: 'textfield',
    height: 'auto',
    border: false,
    items: [{
        fieldLabel: 'IP Adresse',
        readOnly:true,
        name: 'ipadr'
    },{
        fieldLabel: 'Kategorie',
        readOnly:true,
        name: 'kategorie'
    },{
        fieldLabel: 'Kinder',
        readOnly:true,
        name: 'kinder'
    },{
        fieldLabel: 'Erwachsene',
        readOnly:true,
        name: 'erwachsene'
    },{
        fieldLabel: 'Adresse',
        readOnly:true,
        name: 'adresse'
    },{
        fieldLabel: 'PLZ',
        readOnly:true,
        name: 'plz'
    },{
        fieldLabel: 'Ort',
        readOnly:true,
        name: 'ort'
    },{
        fieldLabel: 'e-Mail',
        readOnly:true,
        name: 'emailAP'
    },{
        fieldLabel: 'Vorname AP',
        readOnly:true,
        name: 'vorname'
    },{
        fieldLabel: 'Nachname AP',
        readOnly:true,
        name: 'nachname'
    },{
        fieldLabel: 'letzter Bearbeiter',
        readOnly:true,
        name: 'username'
    },{
        fieldLabel: 'zuletzt bearbeitet',
        readOnly:true,
        hideTrigger:true,
        format: 'd.m.Y',
        xtype:'datefield',
        name: 'letzteBearbeitung'
    }]
});
//In diesem FormPanel werden die erstellte Grid und das Fieldset gemerged.
var anfrageGridForm = new Ext.FormPanel({
    id: 'anfrageGridForm',
    frame: false,
    labelAlign: 'left',

```

```

iconCls: 'anfrage',
closable:true,
title: 'Anfragen',
layout: 'border',
items: [anfrageGrid, anfrageMainFieldset]
});

//beim Schließen des Anfrage-Tabs müssen auch sämtliche erzeugte Variablen wieder zerstört werden, damit der Referenz-
Count auf 0 gestellt wird
//damit der Garbage-Collector seine Arbeit erledigen kann.
anfrageGridForm.addListerner('beforedestroy',function() {
    ticketstore.destroy();
    anfrageGrid.destroy();
    anfrageMainFieldset.destroy();
});

//die ersten 10 Datensätze in den Store laden (Event an die Grid)
pollForChanges(anfrageGrid,true);
//hier wird die gerade erstellte GridForm-Kombination an das TabPanel im Center-Bereich geadded
Ext.getCmp('tabcenter').add(anfrageGridForm).show();
//doLayout ist notwendig, wenn ein Item in einen bereits gerenderten Container nachträglich
//eingepflegt wird. So wie die grid in das Tabpanel.
Ext.getCmp('bl2_center').doLayout();
} else Ext.getCmp('tabcenter').setActiveTab('anfrageGridForm');
}
}

```

## 10.2 Auswertungen

```

*****-----Funktion, um einen Tab zum Mainbereich zu adden.-----
Ich habe mich für diese Variante entschieden, da somit das Initieren des Programmes flotter geht.
Beim Instanziieren eines neuen Tabs für das Center-Panel wird der jeweilige Programmteil nachgeladen.
Ich bin der Meinung das diese Herangehensweise den Workflow des Users am wenigsten beeinträchtigt.
Technisch wäre es auch möglich, alle Tabs beim Ext.onReady() Event zu laden. Man hätte dadurch den Vorteil
einer verkürzten Zeit während des Betriebes. Dafür dauert das initiiierende Laden des Programmes um ein viel-
faches länger und nicht benötigte Programmteile werden gleich von Beginn an immer mitgeladen.
*****-----/
```

```

//Pfad für Druck-Layout-Einstellungs-CSS:
Ext.ux.GridPrinter.stylesheetPath = 'Templates/Application/css/grid_print.css';

function addAuswertungenTab(){
    //zuerst gehört gecheckt, ob es den Tab nicht vielleicht schon gibt.
    //wenn ja, wird er nur mehr focusiert, wenn nein werden die Grids, Forms und Stores erstellt
    //und alles läuft wie geschmiert :)
    if(document.getElementById('auswertungsTab') == null) {

        var CenterAuswertungen = new Ext.Panel({
            region:'center',
            layout:'fit',
            frame:false,
            border:false
        });

        var WestAuswertungen = new Ext.Panel({
            region:'west',
            border:true,
            title:'Buchungen auswerten:',
            frame:true,
            layout: {
                type:'vbox',
                padding:'30',
                align:'stretch'
            },
            defaults:{margins:'0 0 30 0'},
            width:200,

```

```

items:[
    new Ext.Button({
        text: 'je JHB',
        id:'btnAuswertung1',
        iconCls: 'herbergen',
        scale:'medium',
        flex:1,
        handler: function() {

            var Buchungjh = new Ext.form.FieldSet({
                labelAlign: 'left',
                labelWidth: 150,
                layout:'form',
                defaults: {width: 200},
                bodyStyle:'padding:10px',
                defaultType: 'textfield',
                title:'Auswahl',
                height: 'auto',
                border: false,
                items:[{
                    fieldLabel: 'JHB',
                    xtype:'combo',
                    allowBlank:false,
                    editable:false,
                    forceSelection:true,
                    id:'jhCombolD',
                    triggerAction:'all',
                    mode: 'local',
                    store: new Ext.data.Store({
                        proxy: new Ext.data.HttpProxy({
                            url: 'index.php',
                            method: 'POST'
                        }),
                        reader: new Ext.data.JsonReader({
                            root: 'results',
                            totalProperty: 'total',
                            id: 'ID_Jhb'
                        },Ext.data.Record.create([
                            {name: 'ID_Jhb', type: 'int'},
                            {name: 'jhb', type: 'string'}
                        ])
                    },
                    baseParams:{
                        cmd: "GetJhb"
                    },
                    autoLoad: true
                }),
                valueField: 'ID_Jhb',
                displayField: 'jhb',
                name: 'jhb'
            },{
                fieldLabel: 'Datum Von',
                allowBlank:false,
                format: 'Y-m-d',
                id:'txtJhbVonTermin',
                xtype:'xdatefield',
                name: 'Vontermin'
            },{
                fieldLabel: 'Datum Bis',
                allowBlank:false,
                id:'txtJhbBisTermin',
                format: 'Y-m-d',
                xtype:'xdatefield',
                name: 'Bistermin'
            })
        });
    }
];

```

```

var showBuchungjh = new Ext.form.FormPanel({
    labelAlign: 'left',
    id:'Buchungjh',
    items: [
        {
            activeTab: 0,
            defaults:{autoHeight:true},
            items:[Buchungjh]
        }
    ],
    buttons: [
        {
            text:'Berechnung',
            formBind: true,
            handler: function() {

                //Daten Store JHB
                var BuchungjhStore = new Ext.data.Store({
                    proxy: new Ext.data.HttpProxy({
                        url: 'index.php',
                        method: 'POST'
                    }),
                    remoteSort:true,
                    baseParams:{
                        cmd: "FillJHBAuswertungGrid"
                    },
                    reader: new Ext.data.JsonReader({
                        root: 'results',
                        totalProperty: 'total'
                    },Ext.data.Record.create([
                        {name: 'jhb', type:'string'},
                        {name: 'resyBuchungsNr', type:'string'},
                        {name: 'organisation', type:'string'},
                        {name: 'Ansprechperson', type:'string'},
                        {name: 'packagename', type:'string'},
                        {name: 'terminAnreise', type:'date', dateFormat: 'Y-m-d'},
                        {name: 'buchungsStatus', type:'string'},
                        {name: 'kinder', type:'int'},
                        {name: 'erwachsene', type:'int'},
                        {name: 'Gesamt', type:'int'}
                    ])
                })
            }
        });
    });

    // Grid JHB
    var BerechnungjhGrid = new Ext.grid.GridPanel({
        title:'Buchung',
        frame: false,
        border:true,
        iconCls: 'herbergen',
        id:'JHBAuswertungGrid',
        closable:true,
        store: BuchungjhStore,
        columns: [
            {header: "Jugendherberge", width: 115, sortable: false, dataIndex: 'jhb'},
            {header: "Resy Buchungsnummer", width: 115, sortable: false, dataIndex: 'resyBuchungsNr'},

            {header: "Schule / Firma", width: 130, sortable: false, dataIndex: 'organisation'},
            {header: "Ansprechperson", width: 115, sortable: false, dataIndex: 'Ansprechperson'},

            {header: "Packagename", width:115, sortable: false, dataIndex: 'packagename'},
            {header: "Anreise Termin", width:90, sortable: false, xtype: 'datecolumn', format: 'd.m.Y', dataIndex: 'terminAnreise'},

            {header: "Buchungs Status", width:90, sortable: false, dataIndex: 'buchungsStatus'},
            {header: "Kinder", width:40, sortable: false, dataIndex: 'kinder'},
            {header: "Erwachsene", width:40, sortable: false, dataIndex: 'erwachsene'},
            {header: "Gesamt", width:40, sortable: false, dataIndex: 'Gesamt'}
        ]
    });

```

```

        stripeRows: true,
        viewConfig: {
            forceFit:true
        },
        layout:'fit',
        region:'center',
        sm: new Ext.grid.RowSelectionModel({
            singleSelect:true
        }),
        tbar: new Ext.Toolbar({
            items: [{tooltip:'Tabelle drucken',
                iconCls:'print',
                id:'DruckenButton',
                width:50,
                handler:function() {
                    //nachträglich eininstalliertes extjs-Plugin
                    Ext.ux.GridPrinter.print(BerechnungjhGrid);
                }
            }]
        });
    });

    BuchungjhStore.load({params:{jhb:Ext.getCmp('jhbcComboid').getValue(),
Vontermin:Ext.getCmp('txtJhbVonTermin').getValue(),Bistermin:Ext.getCmp('txtJhbBisTermin').getValue()}});
    CenterAuswertungen.removeAll();
    CenterAuswertungen.add(BerechnungjhGrid);
    CenterAuswertungen.doLayout();
    win.close();
}
},{
    text: 'cancel',
    handler: function() {
        win.close();
    }
})
});

var win = new Ext.Window({
    title:'Buchungen JHB',
    closable:true,
    width:420,
    border:true,
    resizable:false,
    modal:true,
    items: [showBuchungjh]
}).show(this);
}

},new Ext.Button({
    text: "je Partner",
    id:'btnAuswertung2',
    flex:1,
    iconCls: 'partner',
    scale:'medium',
    handler: function() {

        var BuchungPartner = new Ext.form.FieldSet({
            labelAlign: 'left',
            labelWidth: 150,
            layout:'form',
            defaults: {width: 200},
            bodyStyle:'padding:10px',
            defaultType: 'textfield',
            title:'Auswahl',
            height: 'auto',
            border: false,
            items:[{
                fieldLabel: 'Partner',
                xtype:'combo',

```

```

        allowBlank:false,
        editable:false,
        id:'ComboPartner',
        forceSelection:true,
        triggerAction:'all',
        mode: 'local',
        store: new Ext.data.Store({
            proxy: new Ext.data.HttpProxy({
                url: 'index.php',
                method: 'POST'
            }),
            reader: new Ext.data.JsonReader({
                root: 'results',
                totalProperty: 'total',
                id: 'ID_Partner'
            },Ext.data.Record.create([
                {name: 'ID_Partner', type: 'int'},
                {name: 'firmenname', type: 'string'}
            ])
        },
        baseParams:{
            cmd: "GetPartner"
        },
        autoLoad: true
    }),
    valueField: 'ID_Partner',
    displayField: 'firmenname',
    name: 'firmenname'
},
{
    fieldLabel: 'Datum Von',
    allowBlank:false,
    format: 'Y-m-d',
    id:'txtPartnerVonTermin',
    xtype:'xdatefield',
    name: 'Vontermin'
},
{
    fieldLabel: 'Datum Bis',
    allowBlank:false,
    id:'txtPartnerBisTermin',
    format: 'Y-m-d',
    xtype:'xdatefield',
    name: 'Bistermin'
}
]);
});
```

```

var showBuchungPartner = new Ext.form.FormPanel({
    labelAlign: 'left',
    id:'BuchungPartner',
    items: [
        activeTab: 0,
        defaults:{autoHeight:true},
        items:[BuchungPartner]
    ],
    buttons: [
        text:'Berechnung',
        formBind: true,
        handler: function() {

            //Daten Store Partner
            var BuchungPartnerStore = new Ext.data.Store({
                proxy: new Ext.data.HttpProxy({
                    url: 'index.php',
                    method: 'POST'
                }),
                remoteSort:true,
                baseParams:{
```

```

        cmd: "FillPartnerAuswertungGrid"
    },
    reader: new Ext.data.JsonReader({
        root: 'results',
        totalProperty: 'total'
    },Ext.data.Record.create([
        {name: 'firmenname', type:'string'},
        {name: 'organisation', type:'string'},
        {name: 'Ansprechperson', type:'string'},
        {name: 'Termin', type:'date', dateFormat: 'Y-m-d'},
        {name: 'StandardUhrzeit', type:'string'},
        {name: 'erwachsene', type:'int'},
        {name: 'kinder', type:'int'},
        {name: 'Gesamt_Personen', type:'int'},
        {name: 'buchungsStatus', type:'string'}
    ])
});
});

// Grid Partner
var BerechnungPartnerGrid = new Ext.grid.GridPanel({
    title:'Buchung',
    frame: false,
    border:true,
    iconCls: 'herbergen',
    id:'PartnerAuswertungGrid',
    closable:true,
    store: BuchungPartnerStore,
    columns: [
        {header: "Partner Name", width: 110, sortable: false, dataIndex: 'firmenname'},
        {header: "Organisation", width: 115, sortable: false, dataIndex: 'organisation'},
        {header: "Ansprechperson", width: 110, sortable: false, dataIndex: 'Ansprechperson'},
        {header: "Termin", width:80, sortable: false, xtype: 'datecolumn', format: 'd.m.Y',
            dataIndex: 'Termin'},
        {header: "Standard Zeit", width:65, sortable: false, dataIndex: 'StandardUhrzeit'},
        {header: "Erwachsene", width:40, sortable: false, dataIndex: 'erwachsene'},
        {header: "Kinder", width:40, sortable: false, dataIndex: 'kinder'},
        {header: "Pers Ges.", width:40, sortable: false, dataIndex: 'Gesamt_Personen'},
        {header: "Status", width:50, sortable: false, dataIndex: 'buchungsStatus'}
    ],
    stripeRows: true,
    viewConfig: {
        forceFit:true
    },
    layout:"fit",
    region:'center',
    sm: new Ext.grid.RowSelectionModel({
        singleSelect:true
    }),
    tbar: new Ext.Toolbar({
        items: [{tooltip:'Tabelle drucken',
            iconCls:'print',
            id:'DruckenButton',
            width:50,
            handler:function() {
                //nachträglich eininstalliertes extjs-Plugin
                Ext.ux.GridPrinter.print(BerechnungPartnerGrid);
            }
        }]
    });
};

BuchungPartnerStore.load({params:{jhb:Ext.getCmp('ComboPartner').getValue(),
Vontermin:Ext.getCmp('txtPartnerVonTermin').getValue(),Bistermin:Ext.getCmp('txtPartnerBisTermin').getValue()}});
CenterAuswertungen.removeAll();
CenterAuswertungen.add(BerechnungPartnerGrid);

```

```

        CenterAuswertungen.doLayout();
        win.close();
    }
}{

    text: 'cancel',
    handler: function() {
        win.close();
    }
}
]);
});

var win = new Ext.Window({
    title:'Buchungen Partner',
    closable:true,
    width:420,
    border:true,
    resizable:false,
    modal:true,
    items: [showBuchungPartner]
}).show(this);

//alert("Hallo2");
}
}
]
});
};

var AuswertungPanel = new Ext.Panel({
    title:'Auswertungen',
    closable:true,
    id:'Auswertung',
    layout:'border',
    iconCls: 'auswertungen',
    items:[CenterAuswertungen,WestAuswertungen]
});

//Ext.getCmp('tabcenter').add(leistungsGrid).show();
Ext.getCmp('tabcenter').add(AuswertungPanel).show();
} else Ext.getCmp('tabcenter').setActiveTab('auswertungsTab');
}

//Diese Printklasse habe ich nachträglich installiert...
//Ext.ux.GridPrinter.stylesheetPath = '/some/other/path/gridPrint.css';
//Ext.ux.GridPrinter.print(grid);

```

## 10.3 Buchungen

```

*****
-----Funktion, um einen Tab zum Mainbereich zu adden.-----
Ich habe mich für diese Variante entschieden, da somit das Initiieren des Programmes flotter geht.
Beim Instanziieren eines neuen Tabs für das Center-Panel wird der jeweilige Programmteil nachgeladen.
Ich bin der Meinung das diese Herangehensweise den Workflow des Users am wenigsten beeinträchtigt.
Technisch wäre es auch möglich, alle Tabs beim Ext.onReady() Event zu laden. Man hätte dadurch den Vorteil
einer verkürzten Zeit während des Betriebes. Dafür dauert das initierende Laden des Programmes um ein viel-
faches länger und nicht benötigte Programmteile werden gleich von Beginn an immer mitgeladen.
*****
```

```

function addBuchungTab(){
    //zuerst gehört gecheckt, ob es den Tab nicht vielleicht schon gibt.
    //wenn ja, wird er nur mehr focusiert, wenn nein werden die Grids, Forms und Stores erstellt
    if(document.getElementById("buchungGrid") == null) {

```

```

var stati = new Ext.data.ArrayStore({
    fields: ['Buchungsstatus'],
    data: [['offen'],['reserviert'],['termine_gesendet'],['abgeschlossen'],['bezahlt']]
});

/* IN DIESEM STORE WERDEN DIE DATENSÄTZE DER BUCHUNGEN GESPEICHERT *****/
/*************************************************************/
var buchungStore = new Ext.data.Store({
    proxy: new Ext.data.HttpProxy({
        url: 'index.php',
        method: 'POST'
    }),
    baseParams:{
        cmd: "FillBuchungGrid",
        filter:true,
        comboFilter:'offen'
    },
    remoteSort:false,
    sortInfo: {field:'ID_Buchung', direction:'DESC'},
    reader: new Ext.data.JsonReader({
        root: 'results',
        totalProperty: 'total'
    },Ext.data.Record.create([
        {name: 'ID_Buchung', type:'int'},
        {name: 'resyBuchungsNr', type:'string'},
        {name: 'Organisation', type:'string'},
        {name: 'Ansprechperson', type:'string'},
        {name: 'packagename', type:'string'},
        {name: 'terminAnreise', type:'date', dateFormat: 'Y-m-d'},
        {name: 'ersatzTerminAnreise', type:'date', dateFormat: 'Y-m-d'},
        {name: 'kinder', type:'int'},
        {name: 'erwachsene', type:'int'},
        {name: 'female', type:'int'},
        {name: 'male', type:'int'},
        {name: 'vegetarier', type:'int'},
        {name: 'relVorschriften', type:'string'},
        {name: 'allergien', type:'string'},
        {name: 'username', type:'string'},
        {name: 'letzteBearbeitung', type:'date', dateFormat: 'Y-m-d'},
        {name: 'erstelltAm', type: 'date', dateFormat: 'Y-m-d'},
        {name: 'buchungsStatus', type:'string'}
    ])
});
//das Polling nach neuen Datensätzen soll so lange true sein, bis ein Datensatz in Bearbeitung steht. Während dieser
Bearbeitung
//wird diese Eigenschaft auf false gesetzt und somit ein reload des stores verhindert.
buchungStore._pollEnabled = true;

//DIESER LISTENER LAUSCHT AUF ÄNDERUNGEN, DIE MANUELL IM STORE VORGENOMMEN WURDEN
//UND UPDATED DIESE ÄNDERUNG AUCH INSTANTLY IN DER DATENBANK.
buchungStore.addListener('update',function(st,rec,op){
    //Im Objekt obj werden als Eigenschaften alle Änderungen gespeichert. In der Schleife wird für jede dieser Eigenschaften
    //ein eigener AJAX-Call abgesetzt, der den geänderten Wert in der Datenbank aktualisiert.
    var obj = eval(rec.getChanges());
    if (typeof(obj) == "object") {
        for (var j in obj) { //das ist die Schleife aus dem I-Netz! TOLLE SACHE!
            var fieldwert = j;
            if(j == "packagename"){ //In der Grid ist das Package ein String,
                fieldwert = "ID_Package"; //Darstellen tun wir einen String, updaten eine ID.
            }
            Ext.Ajax.request({ //hier wird der AJAX-Call abgesetzt.
                waitTitle:'Connecting',
                waitMsg:'Sending data...',
                url: 'index.php',
                params: {

```

```

        cmd: "Update",
        key: 'ID_Buchung',
        keyID: rec.data.ID_Buchung,
        table: "tBuchung",
        field: fieldwert,
        value: obj[j]
    },
    failure:function(response,options){
        Ext.example.msg('Status', 'not Updated, Errors occurred!');
    },
    success:function(response,options){
        var responseJSON = Ext.decode(response.responseText);

        if(responseJSON.error==true) { //FEHLERBEHANDLUNG
            handleException(responseJSON);
        } else {
            if(responseJSON.success == true) {
                buchungStore.reload();
                buchungStore.commitChanges();
                Ext.example.msg('Status', 'Updated successfully!');
            } else Ext.example.msg('Fehler!', 'Leider kein Zugriff');
        }
    }
});
};

/****************************************
****Die Funktionen, die in den Toolbarbuttons auf der Grid hinterlegt sind: ****/
/****************************************

/* DIESER BUTTON IST ZUM ANZEIGEN DER BUCHUNGSDATEN ZUSTÄNDIG ****/
/*************************************************************/
function BuchungAnzeigen() {

    //der in der Grid selektierte Record aus dem Store wird vollständig in der Variable rec gespeichert, und in das FormPanel
    //zum Bearbeiten geladen.
    var rec = buchungGrid.getSelectionModel().getSelected();

    //wenn diese if-Abfrage false ergibt ist kein Datensatz in der Grid selektiert und es wird ein entsprechender Hinweis
    //ausgegeben.
    if(rec)
    {

        /* BEREICH FÜR DIE BUCHUNGSDATEN ****/
       /*************************************************************/
        var BuchungShowFS1 = new Ext.form.FieldSet({
            labelAlign: 'left',
            labelWidth: 115,
            columnWidth: .50,
            layout:'form',
            defaults: {width: 130},
            bodyStyle:'padding:10px',
            defaultType: 'textfield',
            height: 'auto',
            border: false,
            items:[{
                fieldLabel: 'Buchungsnr.',
                readOnly:true,
                disabled:true,
                name: 'ID_Buchung'
            },
                fieldLabel: 'Termin*',
                allowBlank:false,
                readOnly:true,
                format: 'd.m.Y'
            ]
        });
    }
}

```

```

xtype:'datefield',
name: 'terminAnreise'
},{
    fieldLabel: 'Ersatztermin',
    format: 'd.m.Y',
    readOnly:true,
    hideTrigger:true,
    xtype:'datefield',
    name: 'ersatzTerminAnreise'
},{
    fieldLabel: 'Package*',
    xtype:'combo',
    disabled:true,
    allowBlank:false,
    editable:false,
    forceSelection:true,
    triggerAction:'all',
    mode: 'local',
    store: new Ext.data.Store({
        proxy: new Ext.data.HttpProxy({
            url: 'index.php',
            method: 'POST'
        }),
        reader: new Ext.data.JsonReader({
            root: 'results',
            totalProperty: 'total',
            id: 'ID_Package'
        },Ext.data.Record.create([
            {name: 'ID_Package', type: 'int'},
            {name: 'packagename', type: 'string'}
        ])),
        baseParams:{
            start: 0,
            cmd: "GetPackage"
        },
        autoLoad: true
    }),
    valueField: 'ID_Package',
    displayField: 'packagename',
    name: 'packagename'
},{
    fieldLabel: 'Organisation',
    readOnly:true,
    disabled:true,
    name: 'Organisation'
},{
    fieldLabel: 'Ansprechperson',
    readOnly:true,
    disabled:true,
    name: 'Ansprechperson'
},{
    fieldLabel: 'Kinder',
    readOnly:true,
    name: 'kinder'
},{
    fieldLabel: 'Erwachsene',
    readOnly:true,
    name: 'erwachsene'
},{
    fieldLabel: 'Weiblich',
    readOnly:true,
    name: 'female'
},{
    fieldLabel: 'Männlich',
    readOnly:true,
    name: 'male'
}
}

```

```

        ]
    });

var BuchungShowFS2 = new Ext.form.FieldSet({
    labelAlign: 'left',
    labelWidth: 115,
    layout:'form',
    columnWidth: .50,
    defaults: {width: 130},
    bodyStyle:'padding:10px',
    defaultType: 'textfield',
    height: 'auto',
    border: false,
    items:[{
        fieldLabel: 'Resy Buchungsnr.',
        allowBlank:false,
        readOnly:true,
        name: 'resyBuchungsNr'
    },{
        fieldLabel: 'Vegetarier',
        readOnly:true,
        name: 'vegetarier'
    },{
        fieldLabel: 'Rel.Vorschriften',
        readOnly:true,
        xtype:'textarea',
        name: 'relVorschriften'
    },{
        fieldLabel: 'Allergien',
        readOnly:true,
        xtype:'textarea',
        name: 'allergien'
    },{
        fieldLabel: 'Buchungsstatus*',
        xtype:'combo',
        allowBlank:false,
        readOnly:true,
        hideTrigger:true,
        editable:false,
        forceSelection:true,
        triggerAction:'all',
        mode: 'local',
        store: stati,
        valueField: 'Buchungsstatus',
        displayField: 'Buchungsstatus',
        name: 'buchungsStatus'
    }
]
});

var BuchungShowFP = new Ext.form.FormPanel({
    labelAlign: 'left',
    id:'PartnerForm',
    title:Daten,
    monitorValid:true,
    layout:'column',
    url:'index.php',
    items: [BuchungShowFS1,BuchungShowFS2],
    buttons: [{{
        text: 'ok',
        handler: function() {
            win.close();
        }
    }}]
});
BuchungShowFP.getForm().loadRecord(rec);

```

```

/* BEREICH FÜR DIE HINWEISE***** */
/*****
var HinweiseStore = new Ext.data.Store({
    proxy: new Ext.data.HttpProxy({
        url: 'index.php',
        method: 'POST'
    }),
    baseParams:{
        cmd: "FillBuchungshinweisGrid",
        ID_Buchung: rec.data.ID_Buchung
    },
    reader: new Ext.data.JsonReader({
        root: 'results',
        totalProperty: 'total'
    },Ext.data.Record.create([
        {name: 'ID_Buchungsinfo', type:'int'},
        {name: 'infoDatum', type:'date', dateFormat: 'Y-m-d'},
        {name: 'infotext', type:'string'},
        {name: 'user', type:'string'}
    ])
});
HinweiseStore.load();

var expander = new Ext.ux.grid.RowExpander({
    tpl : new Ext.Template(
        '<p><b>Hinweistext:</b><br />{infotext}</p>'
    )
});

var HinweiseGrid = new Ext.grid.GridPanel({
    title:'Hinweise',
    stripeRows: true,
    height:300,
    viewConfig: {
        forceFit:true
    },
    animCollapse: false,
    store: HinweiseStore,
    columns: [
        expander,
        {header: "Datum", width: 30, sortable: true, xtype: 'datecolumn', format: 'd.m.Y', dataIndex: 'infoDatum'},
        {header: "User / Hinweis", width: 135, sortable: false, dataIndex: 'user'}
    ],
    sm: new Ext.grid.RowSelectionModel({
        singleSelect:true
    }),
    plugins: expander,
    bbar: new Ext.Toolbar({
        items: ['->',{
            text: ' o.k.',
            width:100,
            iconCls:'grant',
            handler:function() {
                //das Polling für den Store wieder aktivieren...
                win.close();
            }
        }]
    })
});
/* BEREICH FÜR DIE TIMETABLE***** */
/*****
var TimetableStore = new Ext.data.Store({
    proxy: new Ext.data.HttpProxy({

```

```

        url: 'index.php',
        method: 'POST'
    },
    autoLoad:'true',
    baseParams:{  

        cmd: "FillTimetableGrid",
        ID_Buchung: rec.data.ID_Buchung
    },
    reader: new Ext.data.JsonReader({  

        root: 'results',
        totalProperty: 'total'
    },Ext.data.Record.create([
        {name: 'ID_LeistungsZeitpunkt', type:'int'},
        {name: 'Leistung', type:'string'},
        {name: 'EchtDatum', type:'date', dateFormat: 'Y-m-d'},
        {name: 'EchtUhrzeit', type:'string'}
    ])
);
};

var TimetableGrid = new Ext.grid.GridPanel({  

    title:'Zeiten',
    stripeRows: true,
    height:300,
    viewConfig: {
        forceFit:true
    },
    store: TimetableStore,
    columns: [  

        {header: "Leistung", width: 210, sortable: false, dataIndex: 'Leistung'},
        { header: "Datum",
            width: 90,
            sortable: true,
            xtype: 'datecolumn',
            format: 'd.m.Y',
            dataIndex: 'EchtDatum'
        },
        { header: "Uhrzeit",
            width: 90,
            sortable: false,
            dataIndex: 'EchtUhrzeit'
        }
    ],
    sm: new Ext.grid.RowSelectionModel({
        singleSelect:true
    }),
    bbar: new Ext.Toolbar({
        items: ['<-',{
            text: ' o.k.',
            width:100,
            iconCls:'grant',
            handler:function() {
                //das Polling für den Store wieder aktivieren...
                buchungStore._pollEnabled = true;
                win.close();
            }
        }]
    });
};

/* TABPANEL, DER DIE TIMETABLE, DIE HINWEISE und die BUCHUNGSDATEN BEINHALTET****/  

*****  

var BuchungShowTabPanel = new Ext.TabPanel({  

    activeTab: 0,
    items:[BuchungShowFP, TimetableGrid, HinweiseGrid]
});

```

```
/* TABPANEL, DER DIE TIMETABLE, DIE HINWEISE und die BUCHUNGSDATEN BEINHALTET****/
/*********************************************
```

```
var win = new Ext.Window({
```

```
title:'Buchung bearbeiten',
```

```
closable:true,
```

```
width:600,
```

```
height:390,
```

```
layout:'fit',
```

```
border:true,
```

```
resizable:false,
```

```
modal:true,
```

```
items: [BuchungShowTabPanel]
```

```
}).show(this);
```

```
} else {Ext.example.msg('Fehler', 'Bitte wählen Sie einen Datensatz aus!');}
```

```
}
```

```
/* DIESER BUTTON IST ZUM BEARBEITEN DER BUCHUNGSDATEN ZUSTÄNDIG ****/
/*********************************************
```

```
function BuchungBearbeiten() {
```

```
//das Polling des Stores wird ausgeschalten, solange bis die Bearbeitung des Datensatzes abgeschlossen ist.
```

```
buchungStore._pollEnabled = false;
```

```
//der in der Grid selektierte Record aus dem Store wird vollständig in der Variable rec gespeichert, und in das FormPanel zum Bearbeiten geladen.
```

```
var rec = buchungGrid.getSelectionModel().getSelected();
```

```
//wenn diese if-Abfrage false ergibt ist kein Datensatz in der Grid selektiert und es wird ein entsprechender Hinweis ausgegeben.
```

```
if(rec)
```

```
{
```

```
/* BEREICH FÜR DIE BUCHUNGSDATEN ****/
/*********************************************
```

```
var BuchungChangeFS1 = new Ext.form.FieldSet({
```

```
labelAlign: 'left',
```

```
labelWidth: 115,
```

```
columnWidth: .50,
```

```
layout:'form',
```

```
defaults: {width: 130},
```

```
bodyStyle:'padding:10px',
```

```
defaultType: 'textfield',
```

```
height: 'auto',
```

```
border: false,
```

```
items:[{
```

```
fieldLabel: 'Buchungsnr.',
```

```
readOnly:true,
```

```
disabled:true,
```

```
name: 'ID_Buchung'
```

```
},
```

```
fieldLabel: 'Termin*',
```

```
allowBlank:false,
```

```
format: 'd.m.Y',
```

```
xtype:'datefield',
```

```
name: 'terminAnreise'
```

```
},
```

```
fieldLabel: 'Ersatztermin',
```

```
format: 'd.m.Y',
```

```
xtype:'datefield',
```

```
name: 'ersatzTerminAnreise'
```

```
},
```

```
fieldLabel: 'Package*',
```

```
xtype:'combo',
```

```
disabled:true,
```

```
allowBlank:false,
```

```
editable:false,
```

```
forceSelection:true,
```

```
triggerAction:'all',
```

```
mode: 'local',
```

```

store: new Ext.data.Store({
    proxy: new Ext.data.HttpProxy({
        url: 'index.php',
        method: 'POST'
    }),
    reader: new Ext.data.JsonReader({
        root: 'results',
        totalProperty: 'total',
        id: 'ID_Package'
    },Ext.data.Record.create([
        {name: 'ID_Package', type: 'int'},
        {name: 'packagename', type: 'string'}
    ])),
    baseParams:{
        start: 0,
        cmd: "GetPackage"
    },
    autoLoad: true
}),
valueField: 'ID_Package',
displayField: 'packagename',
name: 'packagename'
},
fieldLabel: 'Organisation',
readOnly:true,
disabled:true,
name: 'Organisation'
},
fieldLabel: 'Ansprechperson',
readOnly:true,
disabled:true,
name: 'Ansprechperson'
},
fieldLabel: 'Kinder',
name: 'kinder'
},
fieldLabel: 'Erwachsene',
name: 'erwachsene'
},
fieldLabel: 'Weiblich',
name: 'female'
},
fieldLabel: 'Männlich',
name: 'male'
}
]
});
};

var BuchungChangeFS2 = new Ext.form.FieldSet({
    labelAlign: 'left',
    labelWidth: 115,
    layout:'form',
    columnWidth: .50,
    defaults: {width: 130},
    bodyStyle:'padding:10px',
    defaultType: 'textfield',
    height: 'auto',
    border: false,
    items:[{
        fieldLabel: 'Resy Buchungsnr.',
        allowBlank:false,
        name: 'resyBuchungsNr'
    },
    {
        fieldLabel: 'Vegetarier',
        name: 'vegetarier'
    },
    {
        fieldLabel: 'Rel. Vorschriften',
        name: 'relVorschriften'
    }
];
}
);

```

```

        xtype:'textarea',
        name: 'relVorschriften'
    },{
        fieldLabel: 'Allergien',
        xtype:'textarea',
        name: 'allergien'
    },{
        fieldLabel: 'Buchungsstatus*',
        xtype:'combo',
        allowBlank:false,
        editable:false,
        forceSelection:true,
        triggerAction:'all',
        mode: 'local',
        store: stati,
        valueField: 'Buchungsstatus',
        displayField: 'Buchungsstatus',
        name: 'buchungsStatus'
    }
]
});

var BuchungChangeFP = new Ext.form.FormPanel({
    labelAlign: 'left',
    id:'PartnerForm',
    title:'Daten',
    monitorValid:true,
    layout:'column',
    url:'index.php',
    items: [BuchungChangeFS1,BuchungChangeFS2],
    buttons: [{
        text:'save',
        iconCls:'save',
        formBind: true,
        handler: function() {
            BuchungChangeFP.getForm().updateRecord(buchungGrid.getSelectionModel().getSelected());
            //das Polling für den Store wieder aktivieren...
            buchungStore._pollEnabled = true;
            win.close();
        }
    },
    {
        text: 'cancel',
        handler: function() {
            //das Polling für den Store wieder aktivieren...
            buchungStore._pollEnabled = true;
            win.close();
        }
    }
]);
BuchungChangeFP.getForm().loadRecord(rec);

/* BEREICH FÜR DIE HINWEISE***** */
/************ */

var HinweiseStore = new Ext.data.Store({
    proxy: new Ext.data.HttpProxy({
        url: 'index.php',
        method: 'POST'
    }),
    baseParams:{
        cmd: "FillBuchungshinweisGrid",
        ID_Buchung: rec.data.ID_Buchung
    },
    reader: new Ext.data.JsonReader({
        root: 'results',
        totalProperty: 'total'
    },Ext.data.Record.create([
        {name: 'ID_Buchungsinfo', type:'int'}
    ]));
});

```

```

        {name: 'infoDatum', type:'date', dateFormat: 'Y-m-d'},
        {name: 'infotext', type:'string'},
        {name: 'user', type:'string'}
    })
})
});

HinweiseStore.load();

var expander = new Ext.ux.grid.RowExpander({
    tpl : new Ext.Template(
        '<p><b>Hinweistext:</b><br />{infotext}</p>'
    )
});

var HinweiseGrid = new Ext.grid.GridPanel({
    title:'Hinweise',
    stripeRows: true,
    height:300,
    viewConfig: {
        forceFit:true
    },
    animCollapse: false,
    store: HinweiseStore,
    columns: [
        expander,
        {header: "Datum", width: 30, sortable: true, xtype: 'datecolumn', format: 'd.m.Y', dataIndex: 'infoDatum'},
        {header: "User / Hinweis", width: 135, sortable: false, dataIndex: 'user'}
    ],
    sm: new Ext.grid.RowSelectionModel({
        singleSelect:true
    }),
    plugins: expander,
    tbar: new Ext.Toolbar({
        items: [
            {
                width:50,
                iconCls:'add',
                handler:function() {
                    var neuerHinweisFS = new Ext.form.FieldSet({
                        labelAlign: 'left',
                        labelWidth: 150,
                        layout:'form',
                        title:'neuer Hinweis',
                        bodyStyle:'padding:10px',
                        defaults: {width: 200},
                        defaultType: 'textfield',
                        height: 'auto',
                        border: false,
                        items:[{
                            fieldLabel: 'Hinweis',
                            xtype: 'textarea',
                            name: 'infotext'
                        }]
                    });
                    var neuerHinweisFP = new Ext.form.FormPanel({
                        labelAlign: 'left',
                        monitorValid:true,
                        url:'index.php',
                        items: [neuerHinweisFS],
                        buttons: [{
                            text: 'Speichern',
                            formBind: true,
                            iconCls:'save',
                            handler: function(){
                                neuerHinweisFP.getForm().submit();
                            }
                        }]
                    });
                }
            }
        ]
    })
});

```

```

        method:'POST',
        waitTitle:'Connecting',
        waitMsg:'Sending data...!',
        params: {
            cmd:"InsertBuchungsHinweis",
            ID_Buchung:rec.data.ID_Buchung
        },
        success:function(form, action) {
            var responseData = Ext.util.JSON.decode(action.response.responseText);
            if(responseData.success == true) {
                HinweiseStore.reload();
                HinweiseStore.commitChanges();
                Ext.example.msg('Status', 'Erfolgreich angelegt!');
            } else {
                Ext.example.msg('Fehler', 'Keine Berechtigung');
            }
            win.close();
        },
        failure:function(form, action) {
            var responseJSON = Ext.decode(action.response.responseText);
            handleException(responseJSON);
        }
    });
},
{
    text: 'Cancel',
    handler: function() {win.close();}
});
};

var win = new Ext.Window({
    title:'neuer Hinweis',
    closable:true,
    width:420,
    border:true,
    resizable:false,
    modal:true,
    items: [neuerHinweisFP]
});
win.show(this);
},
{
width:50,
iconCls:'delete',
handler:function() {
    Ext.Ajax.request({
        waitTitle:'Connecting',
        waitMsg:'Sending data...!',
        url: 'index.php',
        params: {
            cmd: "DeleteBuchungsHinweis",
            ID_BuchungsHinweis: HinweiseGrid.getSelectionModel().getSelected().data.ID_Buchungsinfo
        },
        failure:function(response,options){
            Ext.example.msg('Status', 'nicht gelöscht, Errors occurred!');
        },
        success:function(response,options){
            var responseJSON = Ext.decode(response.responseText);

            if(responseJSON.error==true) { //FEHLERBEHANDLUNG
                handleException(responseJSON);
            } else {
                if(responseJSON.success == true) {
                    HinweiseStore.reload();
                    HinweiseStore.commitChanges();
                    Ext.example.msg('Status', 'Erfolgreich gelöscht!');
                }
            }
        }
    });
}
}

```

```

                } else Ext.example.msg('Fehler!', 'Leider kein Zugriff');
            }
        });
    });
},
bbar: new Ext.Toolbar({
    items: ['>', {
        text: ' o.k.',
        width:100,
        iconCls:'grant',
        handler:function() {
            //das Polling für den Store wieder aktivieren...
            buchungStore._pollEnabled = true;
            win.close();
        }
    }]
});
};

/* BEREICH FÜR DIE TIMETABLE***** */
var TimetableStore = new Ext.data.Store({
    proxy: new Ext.data.HttpProxy({
        url: 'index.php',
        method: 'POST'
    }),
    autoLoad:'true',
    baseParams:{
        cmd: "FillTimetableGrid",
        ID_Buchung: rec.data.ID_Buchung
    },
    reader: new Ext.data.JsonReader({
        root: 'results',
        totalProperty: 'total'
    },Ext.data.Record.create([
        {name: 'ID_LeistungsZeitpunkt', type:'int'},
        {name: 'Leistung', type:'string'},
        {name: 'EchtDatum', type:'date', dateFormat: 'Y-m-d'},
        {name: 'EchtUhrzeit', type:'string'}
    ])
);
};

TimetableStore.addListener('update',function(st,rec,op) {
    //Im Objekt obj werden als Eigenschaften alle Änderungen gespeichert. In der Schleife wird für jede dieser
    //ein eigener AJAX-Call abgesetzt, der den geänderten Wert in der Datenbank aktualisiert.

    var obj = eval(rec.getChanges());
    if (typeof(obj) == "object")
        for (var j in obj) {
            Ext.Ajax.request({
                waitTitle:'Connecting',
                waitMsg:'Sending data...',
                url: 'index.php',
                params: {
                    cmd: "Update",
                    key: 'ID_LeistungsZeitpunkt',
                    keyId: rec.data.ID_LeistungsZeitpunkt,
                    table: "tLeistungsZeitpunkt",
                    field: j,
                    value: obj[j]
                },
                failure:function(response,options){
                    Ext.example.msg('Status', 'not Updated, Errors occurred!');
                }
            });
        }
    }
);
}
);

```

#### Eigenschaften

```

        },
        success:function(response,options){
            var responseJSON = Ext.decode(response.responseText);

            if(responseJSON.error==true) { //FEHLERBEHANDLUNG
                handleException(responseJSON);
            } else {
                if(responseJSON.success == true) {
                    TimetableStore.reload();
                    TimetableStore.commitChanges();
                    Ext.example.msg('Status', 'Updated successfully!');
                } else Ext.example.msg('Fehler!', 'Leider kein Zugriff');
            }
        });
    });
};

var TimetableGrid = new Ext.grid.EditorGridPanel({
    title:'Zeiten',
    stripeRows: true,
    height:300,
    viewConfig: {
        forceFit:true
    },
    store: TimetableStore,
    columns: [
        {header: "Leistung", width: 210, sortable: false, dataIndex: 'Leistung'},
        { header: "Datum",
            width: 90,
            sortable: true,
            xtype: 'datecolumn',
            format: 'd.m.Y',
            dataIndex: 'EchtDatum',
            editor: new Ext.form.DateField({
                format: 'd.m.Y'
            })
        },
        { header: "Uhrzeit",
            width: 90,
            sortable: false,
            dataIndex: 'EchtUhrzeit',
            editor: new Ext.form.TextField({allowBlank: false})
        }
    ],
    sm: new Ext.grid.RowSelectionModel({
        singleSelect:true
    }),
    bbar: new Ext.Toolbar({
        items: ['->',{
            text: ' o.k.',
            width:100,
            iconCls:'grant',
            handler:function() {
                //das Polling für den Store wieder aktivieren...
                buchungStore._pollEnabled = true;
                win.close();
            }
        }]
    });
};

/*
 * TABPANEL, DER DIE TIMETABLE, DIE HINWEISE und die BUCHUNGSDATEN BEINHALTET*****
 */
var BuchungChangeTabPanel = new Ext.TabPanel({
    activeTab: 0,

```

```

        items:[BuchungChangeFP, TimetableGrid, HinweiseGrid]
    });

/* TABPANEL, DER DIE TIMETABLE, DIE HINWEISE und die BUCHUNGSDATEN BEINHALTET*****/
var win = new Ext.Window({
title:'Buchung bearbeiten',
closable:true,
width:600,
height:390,
layout:'fit',
border:true,
resizable:false,
modal:true,
items: [BuchungChangeTabPanel]
}).show(this);
} else {Ext.example.msg('Fehler', 'Bitte wählen Sie einen Datensatz aus!');}
}

//SENDEN VON MAILS ÜBER DEN COMMAND cmdMail.php
function MailSenden() {
if(buchungGrid.getSelectionModel().hasSelection()) {
    var MailFS = new Ext.form.FieldSet({
        labelAlign: 'left',
        labelWidth: 150,
        layout:'form',
        title:'Mail',
        bodyStyle:'padding:10px',
        defaults: {width: 200},
        defaultType: 'textfield',
        height: 'auto',
        border: false,
        items:[
            {
                fieldLabel: 'Betreff',
                xtype: 'textfield',
                name: 'subject'
            },
            {
                fieldLabel: 'Text',
                xtype: 'textarea',
                name: 'body'
            }
        ]
    });

    var neuMailFP = new Ext.form.FormPanel({
        labelAlign: 'left',
        monitorValid:true,
        url:'index.php',
        items: [MailFS],
        buttons: [{text: 'Send', formBind: true, iconCls:'email', handler: function(){
            neuMailFP.getForm().submit({
                method:'POST',
                waitTitle:'Connecting',
                waitMsg:'Sending data...',
                params: {
                    cmd:'Mail',
                    to:buchungGrid.getSelectionModel().getSelected().data.email
                },
                success:function(form, action) {
                    Ext.example.msg('Success', 'Mail erfolgreich gesendet!');
                    win.close();
                }
            });
        }}]
    });
}
}

```

```

        },
        failure:function(form, action) {
            var responseJSON = Ext.decode(action.response.responseText);
            handleException(responseJSON);
        }
    });
}

{
    text: 'Cancel',
    handler: function() {win.close();}
}
});

var win = new Ext.Window({
    title:'Kunden-Mail',
    closable:true,
    width:420,
    border:true,
    resizable:false,
    modal:true,
    items: [neuMailFP]
});
win.show(this);
} else {
    Ext.example.msg('Fehler', 'Kein Datensatz ausgewählt.');
}
}

/*****
***** DIE GRID IM MAINPANEL *****/
/*
//renderer, um das Statusfeld zu färben.
function colorStatus(value, metaData, record, rowIndex, colIndex, store){
    if(value=='offen') {
        metaData.css = 'redtext';
    }
    if(value=='bezahlt') {
        metaData.css = 'greentext';
    }
    if(value=='abgeschlossen') {
        metaData.css = 'greentext';
    }
    return value;
}
*/
//Pfad für Druck-Layout-Einstellungs-CSS:
Ext.ux.GridPrinter.stylesheetPath = 'Templates/Application/css/grid_print.css';

var buchungGrid = new Ext.grid.EditorGridPanel({
    title:'Buchungen',
    frame: false,
    border:true,
    iconCls: 'buchungen',
    id:'buchungGrid',
    closable:true,
    store: buchungStore,
    columns:[
        {header: "Nr", width: 65, sortable: true, dataIndex: 'ID_Buchung'},
        {header: "Erstellt am", width: 105, sortable: true, xtype: 'datecolumn', format: 'd.m.Y', dataIndex: 'erstelltAm'},
        {
            header: "Status",
            width: 115,
            /*renderer:colorStatus*/
        }
    ]
});

```

```

        sortable: true,
        dataIndex: 'buchungsStatus',
        editor: new Ext.form.ComboBox({
            triggerAction: 'all',
            mode: 'local',
            store: stati,
            editable:false,
            lazyRender: true,
            valueField: 'Buchungsstatus',
            displayField: 'Buchungsstatus',
            listClass: 'x-combo-list-small'
        })
    },
    {header: "Termin", width: 100, sortable: true, xtype: 'datecolumn', format: 'd.m.Y', dataIndex: 'terminAnreise'},
    {header: "Kinder", width: 65, sortable: true, dataIndex: 'kinder'},
    {header: "Erw.", width: 65, sortable: true, dataIndex: 'erwachsene'},
    {header: "Package", width: 135, sortable: true, dataIndex: 'packagename'},
    {header: "Organisation", width: 135, sortable: true, dataIndex: 'Organisation'},
    {header: "Ansprechperson", width: 150, sortable: true, dataIndex: 'Ansprechperson'}
],
stripeRows: true,
viewConfig: {
    forceFit:true
},
layout:'fit',
region:'center',
sm: new Ext.grid.RowSelectionModel({
    singleSelect:true
}),
bbar: new Ext.PagingToolbar({
    pageSize: 10,
    store: buchungStore,
    displayInfo: true,
    plugins: new Ext.ux.ProgressBarPager()
}),
tbar: new Ext.Toolbar({
    items: [
        {
            tooltip:'Buchung anzeigen',
            iconCls:'lesen',
            id:'BuchungShowButton',
            width:50,
            handler: BuchungAnzeigen
        },
        {
            tooltip:'Buchung bearbeiten',
            iconCls:'schreiben',
            id:'BuchungEditButton',
            width:50,
            handler: BuchungBearbeiten
        },
        {
            tooltip:'Mail an Kunden senden',
            iconCls:'email',
            id:'MailSendenButton',
            width:50,
            handler:MailSenden
        },
        {
            tooltip:'Tabelle drucken',
            iconCls:'print',
            id:'DruckenButton',
            width:50,
            handler:function() {
                //nachträglich eininstalliertes extjs-Plugin
                Ext.ux.GridPrinter.print(buchungGrid);
            }
        },
        {
            '->':{
                xtype:'combo',
                typeAhead: false,
                triggerAction:'all'
            }
        }
    ]
})
}

```

```

        mode: 'local',
        width:100,
        store: stati,
        editable:false,
        id:'comboFilter',
        value:'offen',
        valueField: 'Buchungsstatus',
        displayField: 'Buchungsstatus'
    },{
        xtype:'button',
        enableToggle: true,
        width:70,
        id:'btnFilter',
        text:'gefiltert',
        pressed:true,
        toggleHandler: function() {
            if(Ext.getCmp('btnFilter').pressed) {
                Ext.getCmp('btnFilter').setText('gefiltert');
            }
            else {
                Ext.getCmp('btnFilter').setText('ungefiltert');
            }
            buchungStore.setBaseParam('filter', Ext.getCmp('btnFilter').pressed);
            buchungStore.setBaseParam('comboFilter', Ext.getCmp('comboFilter').getValue());
            buchungStore.load({params:{start:0, limit:10}});
        },
        tooltip:'Filter aus Combobox benutzen'
    },'-','Suchen',{
        xtype:'textfield',
        name:'searchfield',
        id:'buchungssuchfeld',
        enableKeyEvents:true,
        emptyText:'Suchkriterien'
    })
});
};

Ext.getCmp('comboFilter').addListerner('select',function() {
    buchungStore.setBaseParam('comboFilter', Ext.getCmp('comboFilter').getValue());
    buchungStore.load({params:{start:0, limit:10}});
});

//dieser Event wird bei jedem Keyup-Event des Suchfeldes ausgelöst. Wir machen nach jedem Buchstaben einen AJAX-Request
und
//schießen das Ergebnis dieses neuen Requests an den Store. Hoffentlich ist der Datenbankzugriff schnell genug damit sich
dieser geile
//moderne Effekt den ich mir vorstelle auch zu 100% entfaltet... -> wenn dieser Kommentar in der Endversion noch enthalten
ist, dann gehts :)
Ext.getCmp('buchungssuchfeld').addListerner('keyup',function(tf,e) {
    buchungStore.reload({params:{suchen:tf.getValue(),start:0, limit:10}});
});

//beim Schließen des Buchungs-Tabs müssen auch sämtliche erzeugte Variablen wieder zerstört werden, damit der Referenz-
Count auf 0 gestellt wird
//damit der Garbage-Collector seine Arbeit erledigen kann.
buchungGrid.addListerner('beforedestroy',function() {
    buchungStore.destroy();
});
//die ersten 10 Datensätze in den Store laden (Event an die Grid)
pollForChanges(buchungGrid,true);
//hier wird die gerade erstellte GridForm-Kombination an das TabPanel im Center-Bereich geadded
Ext.getCmp('tabcenter').add(buchungGrid).show();
//doLayout ist notwendig, wenn ein Item in einen bereits gerenderten Container nachträglich
//eingepflegt wird. So wie die grid in das Tabpanel.
//Ext.getCmp('bl2_center').doLayout();
} else Ext.getCmp('tabcenter').setActiveTab('buchungGrid');
}

```

## 10.4 Jugendherberge

```
*****
-----Funktion, um einen Tab zum Mainbereich zu adden.-----
Ich habe mich für diese Variante entschieden, da somit das Initiiieren des Programmes flotter geht.
Beim Instanziieren eines neuen Tabs für das Center-Panel wird der jeweilige Programmteil nachgeladen.
Ich bin der Meinung das diese Herangehensweise den Workflow des Users am wenigsten beeinträchtigt.
Technisch wäre es auch möglich, alle Tabs beim Ext.onReady() Event zu laden. Man hätte dadurch den Vorteil
einer verkürzten Zeit während des Betriebes. Dafür dauert das initiiierende Laden des Programmes um ein viel-
faches länger und nicht benötigte Programmteile werden gleich von Beginn an immer mitgeladen.
*****/
```

```
function addJugendherbergenTab(){
    //zuerst gehört gecheckt, ob es den Tab nicht vielleicht schon gibt.
    //wenn ja, wird er nur mehr focusiert, wenn nein werden die Grids, Forms und Stores erstellt
    //und alles läuft wie geschmiert :)
    if(document.getElementById('JhbTeilung') == null) {
```

```
*****
***** Hier startet der Hauptbereich für die Jungen Hotels. Liegt im Zentrum des Hauptpanels *****/
*****/
```

```
// create the data store Junge Hotels
var jhbStore = new Ext.data.Store({
    proxy: new Ext.data.HttpProxy({
        url: 'index.php',
        method: 'POST'
    }),
    remoteSort:true,
    baseParams:{
        cmd: "FillJhbGrid"
    },
    sortInfo: {field:'jhb', direction:'ASC'},
    reader: new Ext.data.JsonReader({
        root: 'results',
        totalProperty: 'total'
    },Ext.data.Record.create([
        {name: 'ID_Jhb', type:'int'},
        {name: 'jhb', type:'string'},
        {name: 'adresse', type:'string'},
        {name: 'plz', type:'string'},
        {name: 'ort', type:'string'}
    ])
);
//das Polling nach neuen Datensätzen soll so lange true sein, bis ein Datensatz in Bearbeitung steht. Während dieser
Bearbeitung
//wird diese Eigenschaft auf false gesetzt und somit ein reload des stores verhindert.
//jhbsStore._pollEnabled = false;
jhbsStore.addListerner('update',function(st,rec,op) {
    //Hier brauchen wir einen kleinen Trick den ich (Gott sei Dank)
    //aus dem Netz der Netze geklaut habe! Wir müssen ja für jedes upgedatete Feld
    //einen Ajax-Update-Call machen. Ich hatte aber von rec.getChanges() nur ein Objekt
    //indem ALLE geänderten Werte gespeichert waren. Diese kurze aber echt geile Schleife
    //löst mein Problem und setzt für jede Property des Objects einen AJAX-Update-Call ab
    //mit den geänderten Werten! Genial einfach, einfach genial!!
    var obj = eval(rec.getChanges());
    if (typeof(obj) == "object") {
        for (var j in obj) { //das ist die Schleife aus dem I-Netz! TOLLE SACHE!
            Ext.Ajax.request({ //hier wird der AJAX-Call abgesetzt.
                waitTitle:'Connecting',
                waitMsg:'Sending data...',
                url: 'index.php',
                params: {
                    cmd:"Update",

```

```

        key: 'ID_Jhb',
        keyID: rec.data.ID_Jhb,
        table: "tJhb",
        field: j,
        value: obj[j]
    },
    failure:function(response,options){
        Ext.example.msg('Status', 'not Updated, Errors occurred!');
    },
    success:function(response,options){
        var responseData = Ext.util.JSON.decode(response.responseText);
        if(responseData.success == true) {
            jhbStore.reload();
            jhbStore.commitChanges();
            Ext.example.msg('Status', 'Updated successfully!');
        } else {
            Ext.example.msg('Fehler!', 'Leider kein Zugriff');
        }
    }
});
});

//Pfad für Druck-Layout-Einstellungs-CSS:
Ext.ux.GridPrinter.stylesheetPath = 'Templates/Application/css/grid_print.css';

var JhbGrid = new Ext.grid.GridPanel({
    title:'Junge Hotels',
    frame: false,
    border:true,
    iconCls: 'herbergen',
    id:'JhbGrid',
    closable:true,
    store: jhbStore,
    columns: [
        {header: "Nr", width: 15, sortable: false, dataIndex: 'ID_Jhb'},
        {header: "Junge Hotels", width: 50, sortable: false, dataIndex: 'jhb'},
        {header: "Adresse", width: 50, sortable: false, dataIndex: 'adresse'},
        {header: "Plz", width: 20, sortable: false, dataIndex: 'plz'},
        {header: "Ort", width: 50, sortable: false, dataIndex: 'ort'}
    ],
    stripeRows: true,
    viewConfig: {
        forceFit:true
    },
    region:'center',
    sm: new Ext.grid.RowSelectionModel({
        singleSelect:true
    }),
    bbar: new Ext.PagingToolbar({
        pageSize: 10,
        store: jhbStore,
        displayInfo: true,
        plugins: new Ext.ux.ProgressBarPager()
    }),
    tbar: new Ext.Toolbar({
        items: [
            {
                tooltip:'Junge Hotels anzeigen',
                iconCls:'lesen',
                width:50,
                id:'JhbShowButton',
                handler:function() {
                    var rec = JhbGrid.getSelectionModel().getSelected();
                    if(rec)
                    {

```

```

//Datensatz Junge Hotels Anzeigen
var showJhb = new Ext.form.FieldSet({
    labelAlign: 'left',
    labelWidth: 150,
    layout:'form',
    defaults: {width: 200},
    bodyStyle:'padding:10px',
    defaultType: 'textfield',
    height: 'auto',
    border: false,
    items:[{
        fieldLabel: 'Hotel-ID',
        readOnly:true,
        name: 'ID_Jhb'
    },{
        fieldLabel: 'Junges Hotel*',
        allowBlank:false,
        name: 'jhb'
    },{
        fieldLabel: 'Adresse',
        allowBlank:false,
        name: 'adresse'
    },{
        fieldLabel: 'PLZ*',
        allowBlank:false,
        name: 'plz'
    },{
        fieldLabel: 'Ort*',
        allowBlank:false,
        name: 'ort'
    }]
});

var showJhbFP = new Ext.form.FormPanel({
    labelAlign: 'left',
    id:'ticketForm',
    items: [
        //xtype:'tabpanel',
        activeTab: 0,
        defaults:{autoHeight:true},
        items:[showJhb]
    ],
    buttons: [{text:'OK',
        handler:function() {win.close();}}]
});

var win = new Ext.Window({
    title:'Daten Junge Hotels anzeigen',
    closable:true,
    width:420,
    border:true,
    resizable:false,
    modal:true,
    items: [showJhbFP]
});
showJhbFP.getForm().loadRecord(JhbGrid.getSelectionModel().getSelected());
win.show(this);
} else {Ext.example.msg('Fehler', 'Bitte wählen Sie einen Datensatz aus!');}
}

tooltip:'Junge Hotels bearbeiten',
iconCls:'schreiben',
width:50,
id:'JhbEditButton',
handler:function() {
    //das Polling des Stores wird ausgeschalten, solange bis die Bearbeitung des Datensatzes abgeschlossen ist.
    //jhbsStore._pollEnabled = false;
}
}

```

```

//der in der Grid selektierte Record aus dem Store wird vollständig in der Variable rec gespeichert, und in
das FormPanel zum Bearbeiten geladen.
var rec = JhbGrid.getSelectionModel().getSelected();
//wenn diese if-Abfrage false ergibt ist kein Datensatz in der Grid selektiert und es wird ein entsprechender
Hinweis ausgegeben.
if(rec)
{
    //Daten Junge Hotels Ändern
    var writeJhbAndern = new Ext.form.FieldSet({
        labelAlign: 'left',
        labelWidth: 150,
        layout:'form',
        defaults: {width: 200},
        bodyStyle:'padding:10px',
        defaultType: 'textfield',
        //title:'Ändern der Daten',
        height: 'auto',
        border: false,
        items:[{
            fieldLabel: 'Hotel-ID',
            readOnly:true,
            name: 'ID_Jhb'
        },{
            fieldLabel: 'Junges Hotel*',
            allowBlank:false,
            name: 'jhb'
        },{
            fieldLabel: 'Adresse',
            allowBlank:false,
            name: 'adresse'
        },{
            fieldLabel: 'PLZ*',
            allowBlank:false,
            name: 'plz'
        },{
            fieldLabel: 'Ort*',
            allowBlank:false,
            name: 'ort'
        }]
    });
    var writeJhbFP = new Ext.form.FormPanel({
        labelAlign: 'left',
        id:'JhbForm',
        monitorValid:true,
        items: [
            //xtype:'tabpanel',
            activeTab: 0,
            defaults:{autoHeight:true},
            items:[writeJhbAndern]
        ],
        buttons: [{ text:'save',
            iconCls:'save',
            formBind: true,
            handler: function() {
                writeJhbFP.getForm().updateRecord(JhbGrid.getSelectionModel().getSelected());
                //das Polling für den Store wieder aktivieren...
                //jhbsStore._pollEnabled = true;
                win.close();
            }
        },
        { text: 'cancel',
            handler: function() {
                //das Polling für den Store wieder aktivieren...
                //jhbsStore._pollEnabled = true;
                win.close();
            }
        }]
    });
}

```

```

        }
    ]);

    var win = new Ext.Window({
        title:'Daten Junge Hotels ändern',
        closable:true,
        width:420,
        border:true,
        resizable:false,
        modal:true,
        items: [writeJhbFP]
    });
    writeJhbFP.getForm().loadRecord(JhbGrid.getSelectionModel().getSelected());
    win.show(this);
} else {Ext.example.msg('Fehler', 'Bitte wählen Sie einen Datensatz aus!');}
},
{
    tooltip:'Junge Hotels neu anlegen',
    iconCls:'add',
    id:'JhbNewButton',
    width:50,
    handler:function() {
        var neuJhb = new Ext.form.FieldSet({
            labelAlign: 'left',
            labelWidth: 150,
            layout:'form',
            defaults: {width: 200},
            defaultType: 'textfield',
            bodyStyle:'padding:10px',
            //title:'Junges Hotel neu anlegen',
            height: 'auto',
            border: false,
            items:[{
                fieldLabel: 'Junges Hotel*',
                allowBlank:false,
                name: 'jhb'
            },
            {
                fieldLabel: 'Adresse',
                allowBlank:false,
                name: 'adresse'
            },
            {
                fieldLabel: 'PLZ*',
                allowBlank:false,
                name: 'plz'
            },
            {
                fieldLabel: 'Ort*',
                allowBlank:false,
                name: 'ort'
            }
        });
        var formjhbanlegen = new Ext.form.FormPanel({
            labelAlign: 'left',
            id:'ticketForm',
            url:'index.php',
            buttons: [{
                text: 'Save',
                formBind: true,
                iconCls:'save',
                handler: function(){
                    formjhbanlegen.getForm().submit({
                        method:'POST',
                        waitTitle:'Connecting',
                        waitMsg:'Sending data...',
                        params: {
                            cmd:'InsertJhb'
                        }
                    });
                }
            }]
        });
    }
}

```



```

success:function(response,options){
    var responseData = Ext.util.JSON.decode(response.responseText);
    if(responseData.success == true) {
        jhbStore.reload();
        jhbStore.commitChanges();
        Ext.example.msg('Status', 'Deleted successfully!');
    } else {
        Ext.example.msg('Fehler', 'Keine Berechtigung');
    }
}
});  

jhbStore.reload();
},  

icon: Ext.MessageBox.QUESTION
});  

} else {Ext.example.msg('Fehler', 'Bitte wählen Sie einen Datensatz aus!');}  

},  

iconCls:'delete',  

id:'JhbDeleteButton',  

width:50,  

},  

{  

    tooltip:'Tabelle drucken',  

    iconCls:'print',  

    id:'DruckenButton',  

    width:50,  

    handler:function() {
        //nachträglich eininstalliertes extjs-Plugin
        Ext.ux.GridPrinter.print(leistungsGrid);
    }
},  

{  

    xtype:'textfield',  

    name:'searchfield',  

    id:'jhbsuchfeld',  

    enableKeyEvents:true,  

    emptyText:'Junge Hotels suchen...'  

}
});
});  

//dieser Event wird bei jedem Keyup-Event des Suchfeldes ausgelöst. Wir machen nach jedem Buchstaben einen AJAX-  

Request und  

//schießen das Ergebnis dieses neuen Requests an den Store. Hoffentlich ist der Datenbankzugriff schnell genug damit sich  

dieser geile  

//moderne Effekt den ich mir vorstelle auch zu 100% entfaltet... --> wenn dieser Kommentar in der Endversion noch enthalten  

ist, dann gehts :)  

Ext.getCmp('jhbsuchfeld').addKeyListener('keyup',function(tf,e) {
    jhbStore.reload({params:{suchen:tf.getValue(),start:0, limit:10}});
});  

//Jedesmal, wenn eine Zeile in der Grid ausgewählt wird, werden die Daten im jhbpckStore neu geladen!!  

//Dieser Listener lauscht auf den 'rowselect' - Event, den das SelectionModel der Grid aussendet.  

JhbGrid.getSelectionModel().addListener('rowselect',function() {
    jhbpckStore.load({
        params:{ID_Jhb:JhbGrid.getSelectionModel().getSelected().data.ID_Jhb},
        callback: function(rec,opti,succ) {
            var JSON_Data = JhbGrid.getStore().reader.jsonData;
            if(JSON_Data.error==true) { //FEHLERBEHANDLUNG
                Ext.example.msg("Type: "+JSON_Data.type, JSON_Data.message); //weiter Unterscheidung mit Fehler-
            } else {
                if(JSON_Data.total == 0){
                    Ext.example.msg('Status', 'Noch keine Packages für dieses Hotel angelegt...');  

                }
            }
        }
    });
}

```

**Objekt**

```

    });
    JhbPackGrid.getTopToolbar().setVisible(true);
    JhbPackGrid.setTitle("Packages von: "+JhbGrid.getSelectionModel().getSelected().data.jhb);
};

//die ersten 10 Datensätze in den Store laden (Event an die Grid)
pollForChanges(JhbGrid,true);

/********************* Hier startet der Bereich für die Packages im Ost-Bereich des Hauptpanels. *****/
/********************* Dieser Store versorgt die folgende Grid mit Einträgen. Es sind alle Datensätze, die auf die Jugendherberge bezogen in der Tabelle
//tJhb_Package gespeichert sind!
var jhbpackStore = new Ext.data.Store({
    proxy: new Ext.data.HttpProxy({
        url: 'index.php',
        method: 'POST'
    }),
    remoteSort:true,
    baseParams:{},
        cmd: "FillJhbPackGrid"
},
reader: new Ext.data.JsonReader({
    root: 'results',
    totalProperty: 'total'
},Ext.data.Record.create([
    {name: 'ID_Package', type:'int'},
    {name: 'packagename', type:'string'}
])
)
);
;

//Dieser Store versorgt die ComboBox in der folgenden EditorGrid mit den Packages
//Es werden alle Packages gespeichert!
var packageStore = new Ext.data.Store({
    proxy: new Ext.data.HttpProxy({
        url: 'index.php',
        method: 'POST'
    }),
    reader: new Ext.data.JsonReader({
        root: 'results',
        totalProperty: 'total',
        id: 'ID_Package'
},Ext.data.Record.create([
    {name: 'ID_Package', type: 'int'},
    {name: 'packagename', type: 'string'}
]),
baseParams:{
    start: 0,
    cmd: "GetPackage"
},
autoLoad: true
});

var JhbPackGrid = new Ext.grid.EditorGridPanel({
    frame: true,
    enableHdMenu: false,
    border:true,
    width:200,
    id:'JhbPackGrid',
    tbar: new Ext.Toolbar({
        hidden: true,

```

```

        items: [
            {
                tooltip:'Package hinzufügen',
                iconCls:'add',
                id:'addPackage',
                width:50,
                handler: function() {
                    //Diese Button wird ein neues Element mit dem Standartnamen 'neues Package' in den jhbpakcStore
                    hinzufügen.
                    //ACHTUNG: Dieses Element existiert nur im Store und wird nicht in die Datenbank geschrieben.
                    //Erst wenn der Standartname durch wählen in der Combobox abgeändert wird, wird in die Datenbank
                    geschrieben
                    //konkret passiert das Schreiben in die Datenbank im Listener der am ColumnModel-Editor der ComboBox
                    auf den 'change'-Event lauscht.
                    var Package = JhbPackGrid.getStore().recordType;
                    var p = new Package({
                        packagename: 'neues Package'
                    });
                    JhbPackGrid.stopEditing();
                    jhbpakcStore.insert(0, p);
                    JhbPackGrid.startEditing(0, 0);
                }
            },
            {
                tooltip:'Package löschen',
                iconCls:'delete',
                id:'delPackage',
                width:50,
                handler: function() {
                    Ext.Ajax.request({
                        waitTitle:'Connecting',
                        waitMsg:'Sending data...',
                        url: 'index.php',
                        params: {
                            cmd:"Handlejhbpak",
                            ID_Jhb: JhbGrid.getSelectionModel().getSelected().data.ID_Jhb,
                            ID_Package: JhbPackGrid.getSelectionModel().getSelected().data.ID_Package,
                            remove: 1
                        },
                        failure:function(response,options){
                            Ext.example.msg('Status', 'not Deleted, Errors occurred!');
                        },
                        success:function(response,options){
                            var responseData = Ext.util.JSON.decode(response.responseText);
                            if(responseData.success == true) {
                                jhbpakcStore.reload();
                                jhbpakcStore.commitChanges();
                                Ext.example.msg('Erfolg', 'Package erfolgreich vom Standort entfernt!');
                            } else {
                                Ext.example.msg('Fehler!', 'Dieses Package ist bereits auf diesem Standort zugeordnet!');
                            }
                        }
                    });
                }
            }
        ],
        store: jhbpakcStore,
        title: 'Package',
        clicksToEdit:2,
        //Das Columnmodel in der Editorgrid ist etwas erweitert, im Vergleich zu der normalen Grid.
        //Es wird zusätzlich ein Editor benötigt, um festzulegen, wie die Änderung der Gridzelle abgehandelt werden soll.
        columns: [
            {
                width: 50,
                dataIndex: 'packagename',
                editor: new Ext.form.ComboBox({
                    fieldLabel: 'Package*',
                    xtype:'combo',
                    allowBlank:false,

```

```

        editable:false,
        id:'currentCombo',
        forceSelection:true,
        name:'packagename',
        triggerAction:'all',
        mode: 'local',
        //um alle Packages aus der DB in dieser Combobox anzeigen zu können, müssen wir einen Store erstellen
        //und diesen über den CmdGetPackage-Command befüllen lassen.
        store: packageStore,
        valueField: 'ID_Package',
        displayField: 'packagename',
        //Dieser Listener lauscht auf den Select-Event der Combobox in der EditorGrid
        //Wenn sich die Combobox ändert, wird die Änderung in die Datenbank geschrieben
        //und der Store für die Jhb_Pack Grid wird neu geladen (damit auch der neue Wert enthalten ist)
        listeners:{
            'select':{
                fn: function(combo, record, index) {
                    //Leider gibt es anscheinend (auch lt. API) keine einfachere Möglichkeit, im select-Event die ID des
                    alten
                    //Wertes, also des Wertes der vor dem selecten in der Combo gestanden hat, herauszufinden. Den
                    Text bekommt man allerdings.
                    //Mit dieser Schleife durchlaufe ich den gesamten Packagestore und suche nach der
                    entsprechenden ID. Die ID wird benötigt, um
                    //in der Datenbanktabelle auch Datensätze updaten zu können. Ohne diese ID kann man den
                    entsprechenden Datensatz nicht lokalisieren.
                    var oldValue;
                    //zugegeben, dieses Array sieht echt hässlich aus. Zur Erklärung: im select-Event steht das Objekt
                    "combo" zur Verfügung
                    //ausgehend von diesem Objekt kann man sehr viele Dinge erreichen. Z.B. ist combo.store der
                    hinter der Combobox stehende Store.
                    //Auf diesem Store liegt das Objekt "data". In diesem "Data"-Objekt liegt jetzt endlich das Array, in
                    dem die gesamten Wertpaare als Objekt gespeichert sind.
                    //Der Name dieses Arrays ist "items". Ein Element des Arrays "Items" ist wiederum ein Objekt mit
                    vielen Eigenschaften. Eine Eigenschaft davon heißt "json".
                    //Und auf dieser "json" Eigenschaft liegen jetzt endgültig die Werte als String, gespeichert.
                    for (var i = 0; i < combo.store.data.items.length; ++i) {
                        if(combo.store.data.items[i].json.packagename == combo.startValue) {
                            oldValue = combo.store.data.items[i].json.ID_Package;
                        }
                    }
                    Ext.Ajax.request({
                        waitTitle:'Connecting',
                        waitMsg:'Sending data...!',
                        url: 'index.php',
                        params: {
                            cmd:"HandleJhbPack",
                            ID_Jhb: JhbGrid.getSelectionModel().getSelected().data.ID_Jhb,
                            ID_Package: record.data.ID_Package,
                            oldText: combo.startValue,
                            oldValue: oldValue
                        },
                        failure:function(response,options){
                            Ext.example.msg('Status', 'not Updated, Errors occurred!');
                        },
                        success:function(response,options){
                            var responseData = Ext.util.JSON.decode(response.responseText);
                            if(responseData.success == true) {
                                jhbpakStore.reload();
                                jhbpakStore.commitChanges();
                                Ext.example.msg('Erfolg', 'Package erfolgreich dem Standort zugeordnet!');
                            } else {
                                jhbpakStore.reload();
                                jhbpakStore.commitChanges();
                                Ext.example.msg('Fehler!', 'Dieses Package ist bereits auf diesem Standort
                                zugeordnet!');
                            }
                        }
                    });
                }
            }
        };
    }
}

```

```

        });
    });
},
//Der Listener auf dem Blur Event ist notwendig, damit die Combobox keinen Zahlenwert anzeigt, falls man
//in den Eingabemodus wechselt, keine Eingabe tatigt und danach den Eingabemodus wieder verlsst.
//der Blur-Event wird ausgelst, wenn der Eingabemodus der Combobox beendet wird.
'blur':{
    fn: function() {jhbpPackStore.reload();}
}
},
],
stripeRows: true,
viewConfig: {
    forceFit:true
},
region:'east',
sm: new Ext.grid.RowSelectionModel({
    singleSelect:true
})
});
};

/////////////////////////////////////////////////////////////////
//***** Mergen der beiden Teile JHB und PACKAGES in ein einziges Panel *****/
/////////////////////////////////////////////////////////////////

var JhbTeilung = new Ext.Panel({
    title:'Junge Hotels',
    closable:true,
    id:'JhbTeilung',
    layout:'border',
    iconCls:'herbergen',
    items:[JhbGrid,JhbPackGrid]
});

//beim Schlieen des Jungen Hotels-Tabs mussen auch samtliche erzeugte Variablen wieder zerstort werden, damit der
//Referenz-Count auf 0 gestellt wird
//damit der Garbage-Collector seine Arbeit erledigen kann.
JhbTeilung.addListener('beforedestroy',function() {
    jhbStore.destroy();
    jhbpackStore.destroy();
    packageStore.destroy();
});

//hier wird das erstellte Teilungsgrid an das TabPanel im Center-Bereich geadded
Ext.getCmp('tabcenter').add(JhbTeilung).show();
//doLayout ist notwendig, wenn ein Item in einen bereits gerenderten Container nachtrglich
//eingepflegt wird. So wie das Panel in das Tabpanel.
Ext.getCmp('bl2_center').doLayout();

} else Ext.getCmp('tabcenter').setActiveTab('JhbTeilung');
}
}

```

## 10.5 Kunden

```

-----Funktion, um einen Tab zum Mainbereich zu adden.-----
Ich habe mich fur diese Variante entschieden, da somit das Initiieren des Programmes flotter geht.
Beim Instanziieren eines neuen Tabs fr das Center-Panel wird der jeweilige Programmteil nachgeladen.
Ich bin der Meinung das diese Herangehensweise den Workflow des Users am wenigsten beeintrchtigt.
Technisch wre es auch mglich, alle Tabs beim Ext.onReady() Event zu laden. Man htte dadurch den Vorteil
einer verkrzten Zeit whrend des Betriebes. Dafr dauert das initiiерende Laden des Programmes um ein viel-

```

faches länger und nicht benötigte Programmteile werden gleich von Beginn an immer mitgeladen.  
\*\*\*\*\*

```

function addKundenTab(){
    //zuerst gehört gecheckt, ob es den Tab nicht vielleicht schon gibt.
    //wenn ja, wird er nur mehr fokussiert, wenn nein werden die Grids, Forms und Stores erstellt
    if(document.getElementById("kundenGrid") == null) {

        // create the data store
        var kundenStore = new Ext.data.Store({
            proxy: new Ext.data.HttpProxy({
                url: 'index.php',
                method: 'POST'
            }),
            baseParams:{
                cmd: "FillKundenGrid"
            },
            sortInfo: {field:'ID_Kunde', direction:'DESC'},
            reader: new Ext.data.JsonReader({
                root: 'results',
                totalProperty: 'total'
            },Ext.data.Record.create([
                {name: 'ID_Kunde', type:'int'},
                {name: 'kategoriek', type:'string'},
                {name: 'organisation', type:'string'},
                {name: 'adresse', type:'string'},
                {name: 'plz', type:'string'},
                {name: 'ort', type:'string'},
                {name: 'telefon', type:'string'},
                {name: 'fax', type:'string'},
                {name: 'email', type:'string'},
                {name: 'resykd', type:'string'},
                {name: 'username', type:'string'},
                {name: 'letzteBearbeitung', type:'date', dateFormat: 'Y-m-d'},
                {name: 'bemerkung', type:'string'},
                {name: 'erstelltAm', type: 'date', dateFormat: 'Y-m-d'}
            ])
        })
    });

    //das Polling nach neuen Datensätzen soll so lange true sein, bis ein Datensatz in Bearbeitung steht. Während dieser
    Bearbeitung
    //wird diese Eigenschaft auf false gesetzt und somit ein reload des stores verhindert.

    kundenStore.addListener('update',function(st,rec,op) {
        //Hier brauchen wir einen kleinen Trick den ich (Gott sei Dank)
        //aus dem Netz der Netze geklaut habe! Wir müssen ja für jedes upgedatede Feld
        //einen Ajax-Update-Call machen. Ich hatte aber von rec.getChanges() nur ein Objekt
        //indem ALLE geänderten Werte gespeichert waren. Diese kurze aber echt geile Schleife
        //löst mein Problem und setzt für jede Property des Objects einen AJAX-Update-Call ab
        //mit den geänderten Werten! Genial einfach, einfach genial!!
        var obj = eval(rec.getChanges());
        if (typeof(obj) == "object") {
            for (var j in obj) { //das ist die Schleife aus dem I-Netz! TOLLE SACHE!
                Ext.Ajax.request({ //hier wird der AJAX-Call abgesetzt.
                    waitTitle:'Connecting',
                    waitMsg:'Sending data...',
                    url: 'index.php',
                    params: {
                        cmd:"Update",
                        key: 'ID_Kunde',
                        keyId: rec.data.ID_Kunde,
                        table:"tKunde",
                        field:j,
                        value: obj[j]
                    },
                    failure:function(response,options){
                        Ext.example.msg('Status', 'not Updated, Errors occurred!');
                    }
                });
            }
        }
    });
}

```

```

        },
        success:function(response,options){
            var responseData = Ext.util.JSON.decode(response.responseText);
            if(responseData.success == true){
                kundenStore.reload();
                kundenStore.commitChanges();
                Ext.example.msg('Status', 'Updated successfully!');
            } else {
                Ext.example.msg('Fehler!', 'Leider kein Zugriff');
            }
        }
    });
});
}

//Pfad für Druck-Layout-Einstellungs-CSS:
Ext.ux.GridPrinter.stylesheetPath = 'Templates/Application/css/grid_print.css';

var kundenGrid = new Ext.grid.GridPanel({
    title:'Kunden',
    frame: false,
    border:true,
    iconCls: 'kunden',
    id:kundenGrid',
    closable:true,
    store: kundenStore,
    columns: [
        {header: "Nr", width: 65, sortable: false, dataIndex: 'ID_Kunde'},
        {header: "Übernommen am", width: 95, sortable: false, xtype: 'datecolumn', format: 'd.m.Y', dataIndex: 'erstelltAm'},
        {header: "Organisation", width: 135, sortable: false, dataIndex: 'organisation'},
        {header: "PLZ", width: 115, sortable: false, dataIndex: 'plz'},
        {header: "Ort", width: 115, sortable: false, dataIndex: 'ort'},
        {header: "Resy Knr.", width: 115, sortable: false, dataIndex: 'resykd'},
        {header: "eMail", width: 115, sortable: false, dataIndex: 'email'}
    ],
    stripeRows: true,
    viewConfig: {
        forceFit:true
    },
    layout:'fit',
    region:'center',
    sm: new Ext.grid.RowSelectionModel({
        singleSelect:true
    }),
    bbar: new Ext.PagingToolbar({
        pageSize: 10,
        store: kundenStore,
        displayInfo: true,
        plugins: new Ext.ux.ProgressBarPager()
    }),
    tbar: new Ext.Toolbar({
        items: [
            {
                tooltip:'Kunde anzeigen',
                iconCls:'lesen',
                id:'KundenShowButton',
                width:50,
                handler:function() {

                    var rec = kundenGrid.getSelectionModel().getSelected();
                    if(rec)
                    {
                        //Datensatz Anzeigen - Fieldset (Kundendaten)
                        var Kundanz = new Ext.form.FieldSet({
                            labelAlign: 'left',

```

```

labelWidth: 150,
layout:'form',
defaults: {width: 200},
bodyStyle:'padding:10px',
defaultType: 'textfield',
title:'Kundendaten',
height: 'auto',
border: false,
items:[{
    fieldLabel: 'Kunde-ID',
    disabled:true,
    readOnly:true,
    name: 'ID_Kunde'
},{ 
    fieldLabel: 'Kategorie',
    readOnly:true,
    name: 'kategoriek'
},{ 
    fieldLabel: 'Organisation',
    readOnly:true,
    name: 'organisation'
},{ 
    fieldLabel: 'Adresse',
    readOnly:true,
    name: 'adresse'
},{ 
    fieldLabel: 'Plz',
    readOnly:true,
    hideTrigger:true,
    name: 'plz'
},{ 
    fieldLabel: 'Ort',
    readOnly:true,
    hideTrigger:true,
    name: 'ort'
},{ 
    fieldLabel: 'Telefon',
    readOnly:true,
    name: 'telefon'
},{ 
    fieldLabel: 'Fax',
    readOnly:true,
    name: 'fax'
},{ 
    fieldLabel: 'E-Mail',
    readOnly:true,
    name: 'email'
},{ 
    fieldLabel: 'ResyKdNr.',
    readOnly:true,
    name: 'resyk'
},{ 
    fieldLabel: 'Letzte Bearbeitung',
    readOnly:true,
    hideTrigger:true,
    format: 'd.m.Y',
    xtype:'datefield',
    name: 'letzteBearbeitung'
},{ 
    fieldLabel: 'Username',
    readOnly:true,
    name: 'username'
},{ 
    fieldLabel: 'Bemerkung',
    readOnly:true,
    xtype: 'textarea',
    name: 'bemerkung'
}
]

```

```

        },
        fieldLabel: 'Erstellt am',
        readOnly:true,
        hideTrigger:true,
        format: 'd.m.Y',
        xtype:'datefield',
        name: 'erstelltAm'
    })
});

var showKundanze = new Ext.form.FormPanel({
    labelAlign: 'left',
    id:'ticketForm',
    items: [{
        //xtype:'tabpanel',
        activeTab: 0,
        defaults:{autoHeight:true},
        items:[Kundanz]
    }],
    buttons: [{text:'Ok',
        handler:function() {win.close();}}]
});

var win = new Ext.Window({
    title:'Kunden-Datensatz anzeigen',
    closable:true,
    width:420,
    border:true,
    resizable:false,
    modal:true,
    items: [showKundanze]
});
showKundanze.getForm().loadRecord(kundenGrid.getSelectionModel().getSelected());
win.show(this);
} else {Ext.example.msg('Fehler', 'Bitte wählen Sie einen Datensatz aus!');}
}

{

    tooltip:'Kunde bearbeiten',
    iconCls:'schreiben',
    id:'KundenEditButton',
    width:50,
    handler:function() {

        //der in der Grid selektierte Record aus dem Store wird vollständig in der Variable rec gespeichert, und in
        das FormPanel zum Bearbeiten geladen.
        var rec = kundenGrid.getSelectionModel().getSelected();
        //wenn diese if-Abfrage false ergibt ist kein Datensatz in der Grid selektiert und es wird ein entsprechender
        Hinweis ausgegeben.
        if(rec)
        {
            //Datensatz Ändern - Fieldset (Kundendaten)
            var Kunden = new Ext.form.FieldSet{
                labelAlign: 'left',
                labelWidth: 150,
                layout:'form',
                defaults: {width: 200},
                bodyStyle:'padding:10px',
                defaultType: 'textfield',
                title:'Kundendaten',

```

```

height: 'auto',
border: false,
items:[{
    fieldLabel: 'Kunden-ID',
    name: 'ID_Kunde',
    disabled:true,
    readOnly:true
},{

    fieldLabel: 'Kategorie*',
    xtype:'combo',
    allowBlank:false,
    editable:false,
    forceSelection:true,
    triggerAction:'all',
    mode: 'local',
    store: new Ext.data.ArrayStore({
        fields: ['KategorieText'],
        data: [['Privat'],['Schule'],['Firma']]
    }),
    valueField: 'KategorieText',
    displayField: 'KategorieText',
    name: 'kategoriek'
},{

    fieldLabel: 'Organisation',
    name: 'organisation'
},{

    fieldLabel: 'Adresse*',
    allowBlank:false,
    name: 'adresse'
},{

    fieldLabel: 'Plz*',
    allowBlank:false,
    name: 'plz'
},{

    fieldLabel: 'Ort*',
    allowBlank:false,
    name: 'ort'
},{

    fieldLabel: 'Telefon',
    name: 'telefon'
},{

    fieldLabel: 'Fax',
    name: 'fax'
},{

    fieldLabel: 'E-Mail',
    name: 'email'
},{

    fieldLabel: 'Resy KdNr.',
    name: 'resykd'
},{

    fieldLabel: 'Bemerkung',
    xtype: 'textarea',
    name: 'bemerkung'
}
]
});

var fpKundanze = new Ext.form.FormPanel({
    labelAlign: 'left',
    id:'ticketForm',
    items: [{

        //xtype:'tabpanel',
        activeTab: 0,
        defaults:{autoHeight:true},
        items:[Kunden]
    }],
    ...
});

```

```

        buttons: [{ text:'save',
                      iconCls:'save',
                      formBind: true,
                      handler: function() {
                        fpKundanze.getForm().updateRecord(kundenGrid.getSelectionModel().getSelected());
                        win.close();
                      }
                    },
                    { text: 'cancel',
                      handler: function() {
                        win.close();
                      }
                    }
                  ],
                });

var win = new Ext.Window({
  title:'Kunden-Datensatz bearbeiten',
  closable:true,
  width:420,
  border:true,
  resizable:false,
  modal:true,
  items: [fpKundanze]
});
fpKundanze.getForm().loadRecord(kundenGrid.getSelectionModel().getSelected());
win.show(this);
} else {Ext.example.msg('Fehler', 'Bitte wählen Sie einen Datensatz aus!');}
}

{

  tooltip:'neuer Kunde',
  iconCls:'add',
  id:'KundenNewButton',
  width:50,
  handler:function()
{
  var FsetKundeanlegen = new Ext.form.FieldSet({
    labelAlign: 'left',
    labelWidth: 150,
    layout:'form',
    defaults: {width: 200},
    defaultType: 'textfield',
    bodyStyle:'padding:10px',
    title:'Personendaten',
    height: 'auto',
    border: false,
    items:[{
      fieldLabel: 'Kategorie*',
      xtype:'combo',
      allowBlank:false,
      editable:false,
      forceSelection:true,
      triggerAction:'all',
      mode: 'local',
      store: new Ext.data.ArrayStore({
        fields: ['KategorieText'],
        data: [['Privat'],['Schule'],['Firma']]
      }),
      valueField: 'KategorieText',
      displayField: 'KategorieText',
      name: 'kategorie'
    }]
  });
}

```

```

        },
        fieldLabel: 'Organisation',
        name: 'organisation'
    },
    fieldLabel: 'Adresse*',
    allowBlank:false,
    name: 'adresse'
},
fieldLabel: 'PLZ*',
allowBlank:false,
name: 'plz'
},
fieldLabel: 'Ort*',
allowBlank:false,
name: 'ort'
},
fieldLabel: 'Telefon',
name: 'telefon'
},
fieldLabel: 'Fax',
name: 'fax'
},
fieldLabel: 'eMail',
vtype:'email',
name: 'email'
},
fieldLabel: 'Resy KdNr.',
name: 'resykd'
},
fieldLabel: 'Bemerkung',
name: 'bemerkung',
xtype: 'textarea'
}
});
var neuKundenneuFP = new Ext.form.FormPanel({
    labelAlign: 'left',
    id:'ticketForm',
    monitorValid:true,
    url:'index.php',
    items: [
        //xtype:'tabpanel',
        activeTab: 0,
        defaults:{autoHeight:true},
        items:[FsetKundeanlegen]
    ],
    buttons: [
        {
            text: 'Save',
            formBind: true,
            iconCls:'save',
            handler: function(){
                neuKundenneuFP.getForm().submit({
                    method:'POST',
                    waitTitle:'Connecting',
                    waitMsg:'Sending data...',
                    params: {
                        cmd:'InsertKunden'
                    },
                    success:function(form, action){
                        if(action.result.success == true) {
                            Ext.example.msg('Status', 'Saved successfully!');
                            kundenStore.reload();
                        } else {
                            Ext.example.msg('Fehler', 'Nicht gespeichert!!');
                        }
                        win.close();
                    }
                });
            }
        }
    ]
});

```

```

failure:function(form, action){
    if(action.failureType == 'server'){
        Ext.example.msg('Not Saved. Server-Error Occured.');
    }else{
        Ext.example.msg('Warning!', 'Server is unreachable at the Moment: ' +
        action.response.responseText);
    }
}
});
},
text: 'Cancel',
handler: function() {win.close();}
});
});
var win = new Ext.Window({
    title:'Kunden-Datensatz hinzufügen',
    closable:true,
    width:420,
    border:true,
    resizable:false,
    modal:true,
    items: [neuKundenneuFP]
});
win.show(this);
},
width:50
}
//Test
//Ext.example.msg('Button', 'neuer Kunde');

{
    tooltip:'Kunde löschen',
    iconCls:'delete',
    id:'KundenDeleteButton',
    width:50,
    handler:function() {
        //test

        var rec = kundenGrid.getSelectionModel().getSelected();
        if(rec)
        {
            Ext.Msg.show({
                title:'wirklich löschen?',
                msg: 'Sie sind dabei, eine Anfrage zu löschen. Wollen Sie diese Anfrage wirklich löschen?',
                buttons: Ext.Msg.YESNO,
                fn: function(buttonID) {
                    if(buttonID=='yes') {
                        Ext.Ajax.request({           //hier wird der AJAX-Call abgesetzt.
                            waitTitle:'Connecting',
                            waitMsg:'updating data...',
                            url: 'index.php',
                            params: {
                                cmd:"Update",
                                key: 'ID_Kunde',
                                keyID: kundenGrid.getSelectionModel().getSelected().data.ID_Kunde,
                                table:"tKunde",
                                field: 'aktiv',
                                value:0
                            },
                            failure:function(response,options){
                                Ext.example.msg('Status', 'Not deleted! Error Occured!');
                            },
                            success:function(response,options){

```

```

var responseData = Ext.util.JSON.decode(response.responseText);
if(responseData.success == true) {
    kundenStore.reload();
    kundenStore.commitChanges();
    Ext.example.msg('Status', 'Deleted successfully!');
} else {
    Ext.example.msg('Fehler', 'Keine Berechtigung');
}
}
});
kundenStore.reload();
},
icon: Ext.MessageBox.QUESTION
});
} else {Ext.example.msg('Fehler', 'Bitte wählen Sie einen Datensatz aus!');}
},
// Test
//Ext.example.msg('Button', 'Kunde löschen');
},
tooltip:'Kunde Ansprechsperson',
iconCls:'user',
id:'KundenUserButton',
width:50,
handler:function() {
    var rec = kundenGrid.getSelectionModel().getSelected();
    if(rec)
    {
        // create the data store
        var AnsprechpersonStore = new Ext.data.Store({
            proxy: new Ext.data.HttpProxy({
                url: 'index.php',
                method: 'POST'
            }),
            baseParams:{
                cmd: "FillAnsprechpersonenGrid"
            },
            reader: new Ext.data.JsonReader({
                root: 'results',
                totalProperty: 'total'
            },Ext.data.Record.create([
                {name: 'ID_Ansprechperson', type:'int'},
                {name: 'nachname', type:'string'},
                {name: 'vorname', type:'string'},
                {name: 'telefon', type:'string'},
                {name: 'email', type:'string'},
                {name: 'letzteBearbeitung', type:'date', dateFormat: 'Y-m-d'},
                {name: 'bemerkung', type:'string'},
                {name: 'username', type:'string'}
            ])
        });
    }
};

onStore.load({params:{ID_Kunde:kundenGrid.getSelectionModel().getSelected().data.ID_Kunde}});
//das Polling nach neuen Datensätzen soll so lange true sein, bis ein Datensatz in Bearbeitung steht.
Bearbeitung
//wird diese Eigenschaft auf false gesetzt und somit ein reload des stores verhindert.
AnsprechpersonStore.addListener('update',function(st,rec,op) {
//Hier brauchen wir einen kleinen Trick den ich (Gott sei Dank)
//aus dem Netz der Netze geklaut habe! Wir müssen ja für jedes upgedateete Feld
//einen Ajax-Update-Call machen. Ich hatte aber von rec.getChanges() nur ein Objekt
//indem ALLE geänderten Werte gespeichert waren. Diese kurze aber echt geile Schleife
//löst mein Problem und setzt für jede Property des Objects einen AJAX-Update-Call ab
//mit den geänderten Werten! Genial einfach, einfach genial!!
var obj = eval(rec.getChanges());

```

```

if (typeof(obj) == "object")
    for (var j in obj) { //das ist die Schleife aus dem I-Netz! TOLLE SACHE!
        Ext.Ajax.request({ //hier wird der AJAX-Call abgesetzt.
            waitTitle:'Connecting',
            waitMsg:'Sending data...',
            url: 'index.php',
            params: {
                cmd:"Update",
                key: 'ID_Anspprechperson',
                keyID: rec.data.ID_Anspprechperson,
                table:"tAnspprechperson",
                field: j,
                value: obj[j]
            },
            failure:function(response,options){
                Ext.example.msg('Status', 'not Updated, Errors occured!');
            },
            success:function(response,options){
                var responseData = Ext.util.JSON.decode(response.responseText);
                if(responseData.success == true) {
                    AnspprechpersonStore.reload();
                    AnspprechpersonStore.commitChanges();
                    Ext.example.msg('Status', 'Updated successfully!');
                } else {
                    Ext.example.msg('Fehler!', 'Leider kein Zugriff');
                }
            }
        });
    });
}

var AnspprechpersonGrid = new Ext.grid.EditorGridPanel({
    frame: false,
    border:true,
    height: 'auto',
    id:'AnspprechpersonGrid',
    closable:true,
    store: AnspprechpersonStore,
    columns: [
        {
            header: 'Nr',
            width: 10,
            dataIndex: 'ID_Anspprechperson'
        },
        {
            header: 'Nachname*',
            width: 10,
            dataIndex: 'nachname',
            editor: new Ext.form.TextField({
                allowBlank:false
            })
        },
        {
            header: 'Vorname*',
            width: 10,
            dataIndex: 'vorname',
            editor: new Ext.form.TextField({
                allowBlank:false
            })
        },
        {
            header: 'eMail*',
            width: 10,
            dataIndex: 'email',
            editor: new Ext.form.TextField({
                allowBlank:false
            })
        },
        {
            header: 'Telefon*',
            width: 10,
            dataIndex: 'telefon'
        }
    ]
});

```

```

dataIndex: 'telefon',
editor: new Ext.form.TextField({
    allowBlank:false
}),
},
},
header: 'Bemerkung',
width: 10,
dataIndex: 'bemerkung',
editor: new Ext.form.TextField({}),
},
header: "Letzte Bearbeitung",
width: 10,
sortable: false,
xtype: 'datecolumn',
format: 'd.m.Y',
dataIndex: 'letzteBearbeitung'
}),
stripeRows: true,
viewConfig: {
    forceFit:true
},
layout:'fit',
region:'center',
sm: new Ext.grid.RowSelectionModel({
    singleSelect:true
}),
tbar: new Ext.Toolbar({
    items: [
        {
            tooltip:'neue Ansprechsperson',
            iconCls:'add',
            id:'AnsprechspersonNewButton',
            width:50,
            handler:function()
            {
}
}
]
}),
var FsetAnsprechspersonanlegen = new Ext.form.FieldSet({
    labelAlign: 'left',
    labelWidth: 150,
    layout:'form',
    defaults: {width: 200},
    defaultType: 'textfield',
    bodyStyle:'padding:10px',
    title:'Ansprechspersonen-Daten',
    height: 'auto',
    border: false,
    items:[
        {
            fieldLabel: 'Nachname*',
            allowBlank:false,
            name: 'nachname'
        },
        {
            fieldLabel: 'Vorname*',
            allowBlank:false,
            name: 'vorname'
        },
        {
            fieldLabel: 'Telefon*',
            allowBlank:false,
            name: 'telefon'
        },
        {
            fieldLabel: 'eMail*',
            allowBlank:false,
            vtype:'email',
            name: 'email'
        },
        {
            fieldLabel: 'Bemerkung',
            name: 'bemerkung',
            xtype: 'textarea'
        }
    ]
});

```

```

        }]
    });

    var neuAnsprechspersonneuFP = new Ext.form.FormPanel({
        labelAlign: 'left',
        id:'ticketForm',
        monitorValid:true,
        url:'index.php',
        items: [
            {
                activeTab: 0,
                defaults:{autoHeight:true},
                items:[fsetAnsprechspersonanlegen]
            },
            buttons: [
                {
                    text: 'Save',
                    formBind: true,
                    iconCls:'save',
                    handler: function(){
                        neuAnsprechspersonneuFP.getForm().submit({
                            method:'POST',
                            waitTitle:'Connecting',
                            waitMsg:'Sending data...',
                            params: {
                                cmd:'InsertAnsprechperson',
                                ID_Kunde:
kundenGrid.getSelectionModel().getSelected().data.ID_Kunde
                            },
                            success:function(form, action){
                                if(action.result.success == true) {
                                    Ext.example.msg('Status', 'Saved successfully!');
                                    AnsprechspersonStore.reload();
                                } else {
                                    Ext.example.msg('Fehler', 'Nicht gespeichert!!');
                                }
                                win.close();
                            },
                            failure:function(form, action){
                                if(action.failureType == 'server'){
                                    Ext.example.msg('Not Saved. Server-Error Occured.');
                                }else{
                                    Ext.example.msg('Warning!', 'Server is unreachable at the
Moment: ' + action.response.responseText);
                                }
                                win.close();
                            }
                        });
                    }
                },
                {
                    text: 'Cancel',
                    handler: function() {win.close();}
                }
            ],
            var win = new Ext.Window({
                title:'Kunden-Datensatz hinzufügen',
                closable:true,
                width:420,
                border:true,
                resizable:false,
                modal:true,
                items: [neuAnsprechspersonneuFP]
            });
            win.show(this);
        },
        width:50
    });
}

```

```

        },
        tooltip:'Ansprechsperson löschen',
        iconCls:'delete',
        id:'AnsprechspersonDeleteButton',
        width:50,
        handler:function() {
            var rec = AnsprechspersonGrid.getSelectionModel().getSelected();
            if(rec) {
                Ext.Msg.show({
                    title:'wirklich löschen?',
                    msg:'Sie sind dabei, eine Anfrage zu löschen. Wollen Sie diese
Anfrage wirklich löschen?',
                    buttons: Ext.Msg.YESNO,
                    fn: function(buttonID) {
                        if(buttonID=='yes') {
                            Ext.Ajax.request({ //hier wird der AJAX-Call
                                waitTitle:'Connecting',
                                waitMsg:'updating data...',
                                url: 'index.php',
                                params: {
                                    cmd:"Update",
                                    key: 'ID_Ansprechperson',
                                    keyID:
                                    table:"tAnsprechperson",
                                    field: 'aktiv',
                                    value: 0
                                },
                                failure:function(response,options){
                                    Ext.example.msg('Status', 'Not deleted!
Error Occured!');
                                },
                                success:function(response,options){
                                    var responseData =
Ext.util.JSON.decode(response.responseText);
                                    AnsprechspersonStore.commitChanges();
                                    Ext.example.msg('Status', 'Deleted
successfully!');
                                } else {
                                    Ext.example.msg('Fehler', 'Keine
Berechtigung');
                                }
                            });
                            AnsprechspersonStore.reload();
                        }
                    },
                    icon: Ext.MessageBox.QUESTION
                });
            } else {Ext.example.msg('Fehler', 'Bitte wählen Sie einen Datensatz
aus!');}
        }
    }
}

```

```

var Ansprechspersonfenster = new Ext.Window({
    title:'Ansprechpersonen',
    closable:true,
    width:600,
    height: 450,
    layout:'fit',
    border:true,
    resizable:false,
    modal:true,
    items: [AnsprechspersonGrid]
});

Ansprechspersonfenster.show(this);
} else {Ext.example.msg('Fehler', 'Bitte wählen Sie einen Datensatz aus!');}

},
};

tooltip:'Kunde Mail',
iconCls:'email',
id:'KundenMailButton',
width:50,
handler:function() {

if(kundenGrid.getSelectionModel().hasSelection()) {

    var MailFS = new Ext.form.FieldSet({
        labelAlign: 'left',
        labelWidth: 150,
        layout:'form',
        title:'Mail',
        bodyStyle:'padding:10px',
        defaults: {width: 200},
        defaultType: 'textfield',
        height: 'auto',
        border: false,
        items:[{
            fieldLabel: 'Betreff',
            xtype: 'textfield',
            name: 'subject'
        },
        {
            fieldLabel: 'Text',
            xtype: 'textarea',
            name: 'body'
        }]
    });

    var neuMailFP = new Ext.form.FormPanel({
        labelAlign: 'left',
        monitorValid:true,
        url:'index.php',
        items: [MailFS],
        buttons: [{text: 'Send', formBind: true, iconCls:'email', handler: function(){
            neuMailFP.getForm().submit({
                method:'POST',
                waitTitle:'Connecting',
                waitMsg:'Sending data...',
                params: {
                    cmd:'Mail',
                    to:kundenGrid.getSelectionModel().getSelected().data.email
                },
                success:function(form, action) {

```

```

        Ext.example.msg('Success', 'Mail erfolgreich gesendet!');
        win.close();
    },
    failure:function(form, action) {
        var responseJSON = Ext.decode(action.response.responseText);
        handleException(responseJSON);
    }
});
},
text: 'Cancel',
handler: function() {win.close();}
});

var win = new Ext.Window({
    title:'Kunden-Mail',
    closable:true,
    width:420,
    border:true,
    resizable:false,
    modal:true,
    items: [neuMailFP]
});
win.show(this);
} else {
    Ext.example.msg('Fehler', 'Kein Datensatz ausgewählt.');
}
},
width:50
},{
tooltip:'Tabelle drucken',
iconCls:'print',
id:'DruckenButton',
width:50,
handler:function() {
    //nachträglich eininstalliertes extjs-Plugin
    Ext.ux.GridPrinter.print(kundenGrid);
}
}
};

//Test

//Ext.example.msg('Button', 'Kunde Mail');

'->',{
    xtype:'textfield',
    name:'searchfield',
    id:'kundensuchfeld',
    enableKeyEvents:true,
    emptyText:'Kunde suchen...'
}
});
};

//dieser Event wird bei jedem Keyup-Event des Suchfeldes ausgelöst. Wir machen nach jedem Buchstaben einen AJAX-Request
und
//schießen das Ergebnis dieses neuen Requests an den Store. Hoffentlich ist der Datenbankzugriff schnell genug damit sich
dieser geile
//moderne Effekt den ich mir vorstelle auch zu 100% entfaltet... -> wenn dieser Kommentar in der Endversion noch enthalten
ist, dann gehts :)
Ext.getCmp('kundensuchfeld').addKeyListener('keyup',function(tf,e) {
    kundenStore.reload({params:{suchen:tf.getValue(),start:0, limit:10}});
});

```

```
//beim Schließen des Kunden-Tabs müssen auch sämtliche erzeugte Variablen wieder zerstört werden, damit der Referenz-Count auf 0 gestellt wird
//damit der Garbage-Collector seine Arbeit erledigen kann.
kundenGrid.addListener('beforedestroy',function() {
    kundenStore.destroy();
});
//die ersten 10 Datensätze in den Store laden (Event an die Grid)
pollForChanges(kundenGrid,true);
//hier wird die gerade erstellte GridForm-Kombination an das TabPanel im Center-Bereich geadded
Ext.getCmp('tabcenter').add(kundenGrid).show();
//doLayout ist notwendig, wenn ein Item in einen bereits gerenderten Container nachträglich
//eingepflegt wird. So wie die grid in das Tabpanel.
Ext.getCmp('bl2_center').doLayout();
} else Ext.getCmp('tabcenter').setActiveTab('kundenGrid');
}
```

## 10.6 Leistungen

```
*****-----Funktion, um einen Tab zum Mainbereich zu adden.-----*****
Ich habe mich für diese Variante entschieden, da somit das Initialisieren des Programmes flotter geht.
Beim Instanziieren eines neuen Tabs für das Center-Panel wird der jeweilige Programmteil nachgeladen.
Ich bin der Meinung dass diese Herangehensweise den Workflow des Users am wenigsten beeinträchtigt.
Technisch wäre es auch möglich, alle Tabs beim Ext.onReady() Event zu laden. Man hätte dadurch den Vorteil
einer verkürzten Zeit während des Betriebes. Dafür dauert das initialisierende Laden des Programmes um ein viel-
faches länger und nicht benötigte Programmteile werden gleich von Beginn an immer mitgeladen.
*****-----/
```

```
function addLeistungenTab(){
    //zuerst gehört gecheckt, ob es den Tab nicht vielleicht schon gibt.
    //wenn ja, wird er nur mehr fokussiert, wenn nein werden die Grids, Forms und Stores erstellt
    //und alles läuft wie geschmiert :)
    if(document.getElementById('leistungsGrid') == null) {

        // create the data store
        var leistungStore = new Ext.data.Store({
            proxy: new Ext.data.HttpProxy({
                url: 'index.php',
                method: 'POST'
            }),
            remoteSort:false,
            sortInfo: {field:'ID_Leistung', direction:'DESC'},
            baseParams:{
                cmd: "FillLeistungGrid"
            },
            reader: new Ext.data.JsonReader({
                root: 'results',
                totalProperty: 'total'
            },Ext.data.Record.create([
                {name: 'ID_Leistung', type:'int'},
                {name: 'Leistung', type:'string'},
                {name: 'StandardUhrzeit', type:'string'},
                {name: 'LeistungsBemerkung', type:'string'},
                {name: 'ID_Partner', type:'int'},
                {name: 'firmenname', type:'string'},
            ])
        });
        //das Polling nach neuen Datensätzen soll so lange true sein, bis ein Datensatz in Bearbeitung steht. Während dieser
        Bearbeitung
        //wird diese Eigenschaft auf false gesetzt und somit ein reload des stores verhindert.
        leistungStore._pollEnabled = true;
        leistungStore.addListener('update',function(st,rec,op) {

            var obj = eval(rec.getChanges());
            if (typeof(obj) == "object")

```

```

for (var j in obj) { //das ist die Schleife aus dem I-Netz! TOLLE SACHE!
    var fieldwert = j;

    if(j == "firmenname") { //In der Grid ist das Package ein String,
        fieldwert = "ID_Partner"; //Darstellen tun wir einen String, updaten eine ID.
    }

    Ext.Ajax.request({//hier wird der AJAX-Call abgesetzt.
        waitTitle:'Connecting',
        waitMsg:'Sending data...',
        url: 'index.php',
        params: {
            cmd:"Update",
            key: 'ID_Leistung',
            keyID: rec.data.ID_Leistung,
            table:"tLeistung",
            field: fieldwert,
            value: obj[j]
        },
        failure:function(response,options){
            Ext.example.msg('Status', 'not Updated, Errors occured!');
        },
        success:function(response,options){
            var responseData = Ext.util.JSON.decode(response.responseText);
            if(responseData.success == true){
                leistungStore.reload();
                leistungStore.commitChanges();
                Ext.example.msg('Status', 'Updated successfully!');
            } else {
                Ext.example.msg('Fehler!', 'Leider kein Zugriff');
            }
        }
    });
}

//Pfad für Druck-Layout-Einstellungs-CSS:
Ext.ux.GridPrinter.stylesheetPath = 'Templates/Application/css/grid_print.css';

var leistungsGrid = new Ext.grid.GridPanel({
    title:'Leistung',
    frame: false,
    border:true,
    iconCls: 'leistungen',
    id:'leistungsGrid',
    closable:true,
    store: leistungStore,
    columns: [
        {header: "Nr", width: 65, sortable: true, dataIndex: 'ID_Leistung'},
        {header: "Leistungsname", width: 115, sortable: true, dataIndex: 'Leistung'},
        {header: "Standard-Uhrzeit", width: 135, sortable: true, dataIndex: 'StandardUhrzeit'},
        {header: "Partnername", width:135, sortable: true, dataIndex: 'firmenname'}
    ],
    stripeRows: true,
    viewConfig: {
        forceFit:true
    },
    layout:'fit',
    region:'center',
    sm: new Ext.grid.RowSelectionModel({
        singleSelect:true
    }),
    bbar: new Ext.PagingToolbar({
        pageSize: 10,
        store: leistungStore,

```

```

displayInfo: true,
plugins: new Ext.ux.ProgressBarPager()
}),
tbar: new Ext.Toolbar({
    items: [
        {
            tooltip:'Leistung anzeigen',
            iconCls:'lesen',
            id:'LeistungShowButton',
            width:50,
            handler:function() {
                var rec = leistungsGrid.getSelectionModel().getSelected();
                if(rec)
                {
                    //Datensatz Anzeigen - 1. Fieldset (Buchungsdaten)
                    var Leistunganz = new Ext.form.FieldSet({
                        labelAlign: 'left',
                        labelWidth: 150,
                        layout:'form',
                        defaults: {width: 200},
                        bodyStyle:'padding:10px',
                        defaultType: 'textfield',
                        title:'Leistungsdaten',
                        height: 'auto',
                        border: false,
                        items:[{
                            fieldLabel: 'Partnername',
                            disabled:true,
                            readOnly:true,
                            name: 'firmenname'
                        },{
                            fieldLabel: 'Leistung-ID',
                            disabled:true,
                            readOnly:true,
                            name: 'ID_Leistung'
                        },{
                            fieldLabel: 'Leistung',
                            readOnly:true,
                            name: 'Leistung'
                        },{
                            fieldLabel: 'Standard Uhrzeit',
                            readOnly:true,
                            name: 'StandardUhrzeit'
                        },{
                            fieldLabel: 'Leistungs Bemerkung',
                            readOnly:true,
                            xtype:'textarea',
                            name: 'LeistungsBemerkung'
                        }]
                    });
                }
            });

var showLeistunganz = new Ext.form.FormPanel({
    labelAlign: 'left',
    id:'ticketForm',
    items: [
        {
            //xtype:'tabpanel',
            activeTab: 0,
            defaults:{autoHeight:true},
            items:[Leistunganz]
        },
        buttons: [{text:'Ok',
            handler:function() {win.close();}}]
    ];
};

var win = new Ext.Window({
    title:'Leistungs-Datensatz anzeigen',

```

```

        closable:true,
        width:420,
        border:true,
        resizable:false,
        modal:true,
        items: [showLeistunganze]
    });
    showLeistunganze.getForm().loadRecord(leistungsGrid.getSelectionModel().getSelected());
    win.show(this);
} else {Ext.example.msg("Fehler", 'Bitte wählen Sie einen Datensatz aus!');}
}

},{
    tooltip:'Leistung bearbeiten',
    iconCls:'schreiben',
    id:'LeistungEditButton',
    width:50,
    handler:function() {
        leistungStore._pollEnabled = false;
        //der in der Grid selektierte Record aus dem Store wird vollständig in der Variable rec gespeichert, und in
        das FormPanel zum Bearbeiten geladen.
        var rec = leistungsGrid.getSelectionModel().getSelected();
        //wenn diese if-Abfrage false ergibt ist kein Datensatz in der Grid selektiert und es wird ein entsprechender
        Hinweis ausgegeben.
        if(rec)
        {
            //Datensatz Ändern - Fieldset (Buchungsdaten)
            var writeLeistung = new Ext.form.FieldSet({
                labelAlign: 'left',
                labelWidth: 150,
                layout:'form',
                defaults: {width: 200},
                bodyStyle:'padding:10px',
                defaultType: 'textfield',
                title:'Leistungsdaten',
                height: 'auto',
                border: false,
                items:[{
                    fieldLabel: 'Partner',
                    xtype:'combo',
                    allowBlank:false,
                    editable:false,
                    forceSelection:true,
                    triggerAction:'all',
                    mode: 'local',
                    store: new Ext.data.Store({
                        proxy: new Ext.data.HttpProxy({
                            url: 'index.php',
                            method: 'POST'
                        }),
                        reader: new Ext.data.JsonReader({
                            root: 'results',
                            totalProperty: 'total',
                            id: 'ID_Partner'
                        },Ext.data.Record.create([
                            {name: 'ID_Partner', type: 'int'},
                            {name: 'firmenname', type: 'string'}
                        ])),
                        baseParams:{cmd: "GetPartner"
                    },
                    autoLoad: true
                }),
                valueField: 'ID_Partner',
                displayField: 'firmenname',
                name: 'firmenname'
            })
        }
    }
}

```

```

        },
        fieldLabel: 'Leistung-ID',
        disabled:true,
        readOnly:true,
        name: 'ID_Leistung'
    },{
        fieldLabel: 'Leistung',
        name: 'Leistung'
    },{
        fieldLabel: 'Standard Uhrzeit',
        name: 'StandardUhrzeit'
    },{
        fieldLabel: 'Leistungs Bemerkung',
        xtype:'textarea',
        name: 'LeistungsBemerkung'
    }]
});

var fpLeitunganze = new Ext.form.FormPanel({
    labelAlign: 'left',
    id:'ticketForm',
    items: [{{
        activeTab: 0,
        defaults:{autoHeight:true},
        items:[writeLeistung]
    }},
    buttons: [{text:'save',
        iconCls:'save',
        formBind: true,
        handler: function() {
            fpLeitunganze.getForm().updateRecord(leistungsGrid.getSelectionModel().getSelected());
            //das Polling für den Store wieder aktivieren...
            leistungStore._pollEnabled = true;
            win.close();
        }
    },
    {{
        text: 'cancel',
        handler: function() {
            //das Polling für den Store wieder aktivieren...
            leistungStore._pollEnabled = true;
            win.close();
        }
    }]
}]);
}

var win = new Ext.Window({
    title:'Leistungs-Datensatz bearbeiten',
    closable:true,
    width:420,
    border:true,
    resizable:false,
    modal:true,
    items: [fpLeitunganze]
});
fpLeitunganze.getForm().loadRecord(leistungsGrid.getSelectionModel().getSelected());
win.show(this);
} else {Ext.example.msg('Fehler', 'Bitte wählen Sie einen Datensatz aus!');}
}

{
    tooltip:'neue Leistung',
    iconCls:'add',
    id:'LeistungNewButton',
    width:50,
    handler:function() {

```

```

var FsetLeistunganlegen = new Ext.form.FieldSet({
    labelAlign: 'left',
    labelWidth: 150,
    layout:'form',
    defaults: {width: 200},
    defaultType: 'textfield',
    bodyStyle:'padding:10px',
    title:'Leistungsdaten',
    height: 'auto',
    border: false,
    items:[{
        fieldLabel: 'Partner*',
        xtype:'combo',
        allowBlank:false,
        editable:false,
        forceSelection:true,
        triggerAction:'all',
        mode: 'local',
        store: new Ext.data.Store({
            proxy: new Ext.data.HttpProxy({
                url: 'index.php',
                method: 'POST'
            }),
            reader: new Ext.data.JsonReader({
                root: 'results',
                totalProperty: 'total',
                id: 'ID_Partner'
            },Ext.data.Record.create([
                {name: 'ID_Partner', type: 'int'},
                {name: 'firmenname', type: 'string'}
            ])),
            baseParams:{start: 0, cmd: "GetPartner"}
        },
        autoLoad: true
    }),
    valueField: 'ID_Partner',
    displayField: 'firmenname',
    hiddenName: 'ID_Partner',
    hiddenId:'hiddenLeistungIdCombo',
    name:'ID_Partner'
},{
    fieldLabel: 'Leistung',
    name: 'Leistung'
},{
    fieldLabel: 'Standard Uhrzeit',
    name: 'StandardUhrzeit'
},{
    fieldLabel: 'Leistungs Bemerkung',
    name: 'LeistungsBemerkung'
}]
});

var neuLeistungneuFP = new Ext.form.FormPanel({
    labelAlign: 'left',
    id:'ticketForm',
    monitorValid:true,
    url:'index.php',
    items: [
        activeTab: 0,
        defaults:{autoHeight:true},
        items:[FsetLeistunganlegen]
    ]
});

```

```

        }],
        buttons: [{
            text: 'Save',
            formBind: true,
            iconCls:'save',
            handler: function(){
                neuLeistungneuFP.getForm().submit({
                    method:'POST',
                    waitTitle:'Connecting',
                    waitMsg:'Sending data...',
                    params: {
                        cmd:'InsertLeistung'
                    },
                    success:function(form, action){
                        if(action.result.success == true) {
                            Ext.example.msg('Status', 'Saved successfully!');
                            leistungStore.reload();
                        } else {
                            Ext.example.msg('Fehler', 'Nicht gespeichert!!');
                        }
                        win.close();
                    },
                    failure:function(form, action){
                        if(action.failureType == 'server'){
                            Ext.example.msg('Not Saved. Server-Error Occured.');
                        }else{
                            Ext.example.msg('Warning!', 'Server is unreachable at the Moment: ' +
                            action.response.responseText);
                        }
                        win.close();
                    }
                });
            }
        ],
        text: 'Cancel',
        handler: function() {win.close();}
    });
};

var win = new Ext.Window({
    title:'Leistungs-Datensatz hinzufügen',
    closable:true,
    width:420,
    border:true,
    resizable:false,
    modal:true,
    items: [neuLeistungneuFP]
});
win.show(this);
},
width:50
},
width:50
},
tooltip:'Leistung löschen',
iconCls:'delete',
id:'LeistungDeleteButton',
width:50,
handler:function() {

    var rec = leistungsGrid.getSelectionModel().getSelected();
    if(rec)
    {
        Ext.Msg.show({
            title:'wirklich löschen?',
            msg: 'Sie sind dabei, eine Anfrage zu löschen. Wollen Sie diese Anfrage wirklich löschen?',
            buttons: Ext.Msg.YESNO,
            fn: function(buttonID) {
                if(buttonID=='yes') {
                    Ext.Ajax.request({ //hier wird der AJAX-Call abgesetzt

```

```

        waitTitle:'Connecting',
        waitMsg:'updating data...',
        url: 'index.php',
        params: {
            cmd:"Update",
            key: 'ID_Leistung',
            keyID: leistungsGrid.getSelectionModel().getSelected().data.ID_Leistung,
            table:"tLeistung",
            field: 'aktiv',
            value: 0
        },
        failure:function(response,options){
            Ext.example.msg('Status', 'Not deleted! Error Occured!');
        },
        success:function(response,options){
            var responseData = Ext.util.JSON.decode(response.responseText);
            if(responseData.success == true) {
                leistungStore.reload();
                leistungStore.commitChanges();
                Ext.example.msg('Status', 'Deleted successfully!');
            } else {
                Ext.example.msg('Fehler', 'Keine Berechtigung');
            }
        }
    });
    leistungStore.reload();
}
},
icon: Ext.MessageBox.QUESTION
});
} else {Ext.example.msg('Fehler', 'Bitte wählen Sie einen Datensatz aus!');}
},
{
tooltip:'Tabelle drucken',
iconCls:'print',
id:'DruckenButton',
width:50,
handler:function() {
    //nachträglich eininstalliertes extjs-Plugin
    Ext.ux.GridPrinter.print(leistungsGrid);
}
},'>|{
xtype:'textfield',
name:'searchfield',
id:'leistungssuchfeld',
enableKeyEvents:true,
emptyText:'Leistung suchen...'
}
);
//dieser Event wird bei jedem Keyup-Event des Suchfeldes ausgelöst. Wir machen nach jedem Buchstaben einen AJAX-Request
und
//schießen das Ergebnis dieses neuen Requests an den Store. Hoffentlich ist der Datenbankzugriff schnell genug damit sich
dieser geile
//moderne Effekt den ich mir vorstelle auch zu 100% entfaltet... --> wenn dieser Kommentar in der Endversion noch enthalten
ist, dann gehts :(
Ext.getCmp('leistungssuchfeld').addKeyListener('keyup',function(tf,e) {
    leistungStore.reload({params:{suchen:tf.getValue(),start:0, limit:10}});
});

//beim Schließen des Buchungs-Tabs müssen auch sämtliche erzeugte Variablen wieder zerstört werden, damit der Referenz-
Count auf 0 gestellt wird
//damit der Garbage-Collector seine Arbeit erledigen kann.
leistungsGrid.addDestroyListener('beforedestroy',function() {
    leistungStore.destroy();
});

```

```

//die ersten 10 Datensätze in den Store laden (Event an die Grid)
pollForChanges(leistungsGrid,true);
//hier wird die gerade erstellte GridForm-Kombination an das TabPanel im Center-Bereich geadded
Ext.getCmp('tabcenter').add(leistungsGrid).show();
//doLayout ist notwendig, wenn ein Item in einen bereits gerenderten Container nachträglich
//eingepflegt wird. So wie die grid in das Tabpanel.
Ext.getCmp('bl2_center').doLayout();

} else Ext.getCmp('tabcenter').setActiveTab('leistungsGrid');
}

```

## 10.7 Package

```

*****-----Funktion, um einen Tab zum Mainbereich zu adden.-----
Ich habe mich für diese Variante entschieden, da somit das Initiiieren des Programmes flotter geht.
Beim Instanziieren eines neuen Tabs für das Center-Panel wird der jeweilige Programmteil nachgeladen.
Ich bin der Meinung das diese Herangehensweise den Workflow des Users am wenigsten beeinträchtigt.
Technisch wäre es auch möglich, alle Tabs beim Ext.onReady() Event zu laden. Man hätte dadurch den Vorteil
einer verkürzten Zeit während des Betriebes. Dafür dauert das initiiierende Laden des Programmes um ein viel-
faches länger und nicht benötigte Programmteile werden gleich von Beginn an immer mitgeladen.
*****-----/
```

```

function addPackagesTab(){
    //zuerst gehört gecheckt, ob es den Tab nicht vielleicht schon gibt.
    //wenn ja, wird er nur mehr focusiert, wenn nein werden die Grids, Forms und Stores erstellt
    //und alles läuft wie geschmiert : )
    if(document.getElementById('packageTab') == null) {

        /***** Hier startet der Hauptbereich für die Packages. Liegt im Zentrum des Hauptpanels *****/
        /*****-----/
```

// create the data store Junge Hotels

```

        var packStore = new Ext.data.Store({
            proxy: new Ext.data.HttpProxy({
                url: 'index.php',
                method: 'POST'
            }),
            remoteSort:true,
            baseParams:{
                cmd: "FillPackageGrid"
            },
            reader: new Ext.data.JsonReader({
                root: 'results',
                totalProperty: 'total'
            },Ext.data.Record.create([
                {name: 'ID_Package', type:'int'},
                {name: 'packagename', type:'string'},
                {name: 'pdfPfad', type:'string'},
                {name: 'username', type:'string'},
                {name: 'letzteBearbeitung', type:'date', dateFormat:'Y-m-d'}
            ])
        });
        //packStore._pollEnabled = false;
        packStore.addListener('update',function(st,rec,op) {
            //Hier brauchen wir einen kleinen Trick den ich (Gott sei Dank)
            //aus dem Netz der Netze geklaut habe! Wir müssen ja für jedes upgedatede Feld
            //einen Ajax-Update-Call machen. Ich hatte aber von rec.getChanges() nur ein Objekt
            //indem ALLE geänderten Werte gespeichert waren. Diese kurze aber echt geile Schleife
            //löst mein Problem und setzt für jede Property des Objects einen AJAX-Update-Call ab
            //mit den geänderten Werten! Genial einfach, einfach genial!!
        });
    }
}
```

```

var obj = eval(rec.getChanges());
if (typeof(obj) == "object")
    for (var j in obj) { //das ist die Schleife aus dem I-Netz! TOLLE SACHE!
        Ext.Ajax.request({ //hier wird der AJAX-Call abgesetzt.
            waitTitle:'Connecting',
            waitMsg:'Sending data...',
            url: 'index.php',
            params: {
                cmd:"Update",
                key: 'ID_Package',
                keyID: rec.data.ID_Package,
                table:"tPackage",
                field: j,
                value: obj[j]
            },
            failure:function(response,options){
                Ext.example.msg('Status', 'not Updated, Errors occurred!');
            },
            success:function(response,options){
                var responseData = Ext.util.JSON.decode(response.responseText);
                if(responseData.success == true){
                    packStore.reload();
                    packStore.commitChanges();
                    Ext.example.msg('Status', 'Updated successfully!');
                } else {
                    Ext.example.msg('Fehler!', 'Leider kein Zugriff');
                }
            }
        });
    }
};

//Pfad für Druck-Layout-Einstellungs-CSS:
Ext.ux.GridPrinter.stylesheetPath = 'Templates/Application/css/grid_print.css';

var PackGrid = new Ext.grid.GridPanel({
    frame: false,
    border:true,
    id:'PackGrid',
    store: packStore,
    columns: [
        {header: "Package", width: 50, sortable: false, dataIndex: 'packagename'},
        {header: "Pdf-Pfad", width: 50, sortable: false, dataIndex: 'pdfPfad'},
        {header: "User", width: 50, sortable: false, dataIndex: 'username'}
    ],
    stripeRows: true,
    viewConfig: {
        forceFit:true
    },
    //layout:'fit',
    region:'center',
    sm: new Ext.grid.RowSelectionModel({
        singleSelect:true
    }),
    bbar: new Ext.PagingToolbar({
        pageSize: 10,
        store: packStore,
        displayInfo: true,
        plugins: new Ext.ux.ProgressBarPager()
    }),
    tbar: new Ext.Toolbar({
        items: [
            tooltip:'Packages anzeigen',
            iconCls:'lesen',
            id:'PackShowButton',
            handler:function() {

```

```

var rec = PackGrid.getSelectionModel().getSelected();
if(rec)
{
    //Datensatz Package Anzeigen
    var showPackage = new Ext.form.FieldSet({
        labelAlign: 'left',
        labelWidth: 150,
        layout:'form',
        defaults: {width: 200},
        bodyStyle:'padding:10px',
        defaultType: 'textfield',
        height: 'auto',
        border: false,
        items:[{
            fieldLabel: 'Package-ID',
            readOnly:true,
            name: 'ID_Package'
        },{
            fieldLabel: 'Package',
            readOnly:true,
            name: 'packagename'
        },{
            fieldLabel: 'Pdf-Pfad',
            allowBlank:false,
            name: 'pdfPfad'
        },{
            fieldLabel: 'Zuletzt bearbeitet von',
            readOnly:true,
            name: 'username'
        },{
            fieldLabel: 'Zuletzt bearbeitet am',
            readOnly:true,
            hideTrigger:true,
            disabled:true,
            format: 'd.m.Y',
            xtype:'datefield',
            name: 'letzteBearbeitung'
        }]
    });
}

var showPackageFP = new Ext.form.FormPanel({
    labelAlign: 'left',
    id:'ticketForm',
    items: [{{
        activeTab: 0,
        defaults:{autoHeight:true},
        items:[showPackage]
    }}, {
        buttons: [{text:'Ok',
            handler:function() {win.close();}}]
    }];
});

var win = new Ext.Window({
    title:'Daten Packages anzeigen',
    closable:true,
    width:420,
    border:true,
    resizable:false,
    modal:true,
    items: [showPackageFP]
});
showPackageFP.getForm().loadRecord(PackGrid.getSelectionModel().getSelected());
win.show(this);
} else {Ext.example.msg("Fehler", 'Bitte wählen Sie einen Datensatz aus!');}
},

```

```

width:50,
},
tooltip:'Packe bearbeiten',
iconCls:'schreiben',
id:'PackEditButton',
handler:function() {
    //das Polling des Stores wird ausgeschalten, solange bis die Bearbeitung des Datensatzes abgeschlossen ist.
    //packStore._pollEnabled = false;
    //der in der Grid selektierte Record aus dem Store wird vollständig in der Variable rec gespeichert, und in
das FormPanel zum Bearbeiten geladen.
var rec = PackGrid.getSelectionModel().getSelected();
//wenn diese if-Abfrage false ergibt ist kein Datensatz in der Grid selektiert und es wird ein entsprechender
Hinweis ausgegeben.
if(rec)
{
    //Daten des Packages Ändern
    var writePackAendern = new Ext.form.FieldSet({
        labelAlign: 'left',
        labelWidth: 150,
        layout:'form',
        defaults: {width: 200},
        bodyStyle:'padding:10px',
        defaultType: 'textfield',
        height: 'auto',
        border: false,
        items:[{
            fieldLabel: 'Package ID',
            readOnly:true,
            name: 'ID_Package'
        },{
            fieldLabel: 'Package*',
            allowBlank:false,
            name: 'packagename'
        },{
            fieldLabel: 'Pfad',
            allowBlank:false,
            name: 'pdfPfad'
        }]
    });

    var writePackFP = new Ext.form.FormPanel({
        labelAlign: 'left',
        id:'PackForm',
        monitorValid:true,
        items: [{{
            activeTab: 0,
            defaults:{autoHeight:true},
            items:[writePackAendern]
        }},
        buttons: [{text:'save',
            iconCls:'save',
            formBind: true,
            handler: function() {

writePackFP.getForm().updateRecord(PackGrid.getSelectionModel().getSelected());
                //das Polling für den Store wieder aktivieren...
                //packStore._pollEnabled = true;
                win.close();
            }
        },{
            text: 'cancel',
            handler: function() {
                //das Polling für den Store wieder aktivieren...
                //packStore._pollEnabled = true;
                win.close();
            }
        }]
    });
}

```

```

        ]];
    });

    var win = new Ext.Window({
        title:'Daten Packages ändern',
        closable:true,
        width:420,
        border:true,
        resizable:false,
        modal:true,
        items: [writePackFP]
    });
    writePackFP.getForm().loadRecord(PackGrid.getSelectionModel().getSelected());
    win.show(this);
} else {Ext.example.msg('Fehler', 'Bitte wählen Sie einen Datensatz aus!');}
},
width:50,
};

{
    tooltip:'Packages neu anlegen',
    iconCls:'add',
    id:'PackNewButton',
    width:50,
    handler:function() {
        var neuPack = new Ext.form.FieldSet({
            labelAlign: 'left',
            labelWidth: 150,
            layout:'form',
            defaults: {width: 200},
            defaultType: 'textfield',
            bodyStyle:'padding:10px',
            height: 'auto',
            border: false,
            items:[{
                fieldLabel: 'Package*',
                allowBlank:false,
                name: 'packagename'
            },
            {
                fieldLabel: 'Pfad',
                allowBlank:false,
                name: 'pdfPfad'
            }]
        });
        var formpackanlegen = new Ext.form.FormPanel({
            labelAlign: 'left',
            id:'packForm',
            url:'index.php',
            buttons: [
                {
                    text: 'Save',
                    formBind: true,
                    iconCls:'save',
                    handler: function(){
                        formpackanlegen.getForm().submit({
                            method:'POST',
                            waitTitle:'Connecting',
                            waitMsg:'Sending data...',
                            params: {
                                cmd:'InsertPackage'
                            },
                            success:function(form, action){
                                if(action.result.success == true) {
                                    Ext.example.msg('Status', 'Saved successfully!');
                                    packStore.reload();
                                } else {
                                    Ext.example.msg('Fehler', 'Nicht gespeichert!!');
                                }
                            }
                        });
                    }
                }
            ]
        });
    }
});
```

```

        anlegenpackwindow.close();
    },
    failure:function(form, action){
        if(action.failureType == 'server'){
            Ext.example.msg('Not Saved. Server-Error Occured.');
        }else{
            Ext.example.msg('Warning!', 'Server is unreachable at the Moment: '+
action.response.responseText);
        }
    });
}
},
text: 'Cancel',
handler: function() {anlegenpackwindow.close();}
],
items: [neuPack]
});

var anlegenpackwindow = new Ext.Window ({
    title:'Packages anlegen',
    closable:true,
    width:500,
    border:true,
    resizable:false,
    modal:true,
    items: [formpackanlegen]
});
anlegenpackwindow.show(this);

}
},
tooltip:'Package löschen',
handler: function() {
var rec = PackGrid.getSelectionModel().getSelected();
if(rec)
{
Ext.Msg.show({
    title:'wirklich löschen?',
    msg: 'Sie sind dabei, ein Package zu löschen. Wollen Sie dieses Junges Hotel wirklich löschen?',
    buttons: Ext.Msg.YESNO,
    fn: function(buttonID) {
        if(buttonID=='yes') {
            Ext.Ajax.request({           //hier wird der AJAX-Call abgesetzt.
                waitTitle:'Connecting',
                waitMsg:'updating data...',
                url: 'index.php',
                params: {
                    cmd:"Update",
                    key: 'ID_Package',
                    keyID: PackGrid.getSelectionModel().getSelected().data.ID_Package,
                    table:"tPackage",
                    field: 'aktiv',
                    value: 0
                },
                failure:function(response,options){
                    Ext.example.msg('Status', 'Not deleted! Error Occured!');
                },
                success:function(response,options){
                    var responseData = Ext.util.JSON.decode(response.responseText);
                    if(responseData.success == true) {
                        packStore.reload();
                        packStore.commitChanges();
                        Ext.example.msg('Status', 'Deleted successfully!');
                    } else {
                
```

```

                Ext.example.msg('Fehler', 'Keine Berechtigung');
            }
        }
    });
    packStore.reload();
}
},
icon: Ext.MessageBox.QUESTION
});
} else {Ext.example.msg('Fehler', 'Bitte wählen Sie einen Datensatz aus!');}
},
iconCls:'delete',
id:'PackageDeleteButton',
width:50,
};

{
tooltip:'Tabelle drucken',
iconCls:'print',
id:'DruckenButton',
width:50,
handler:function() {
    //nachträglich eininstalliertes extjs-Plugin
    Ext.ux.GridPrinter.print(PackGrid);
}
},'->',{
xtype:'textfield',
name:'searchfield',
id:'packsuchfeld',
enableKeyEvents:true,
emptyText:'Packages suchen...'
})
});
};

//dieser Event wird bei jedem Keyup-Event des Suchfeldes ausgelöst. Wir machen nach jedem Buchstaben einen AJAX-Request und
//schießen das Ergebnis dieses neuen Requests an den Store. Hoffentlich ist der Datenbankzugriff schnell genug damit sich
dieser geile
//moderne Effekt den ich mir vorstelle auch zu 100% entfaltet... --> wenn dieser Kommentar in der Endversion noch enthalten
ist, dann gehts :)
Ext.getCmp('packsuchfeld').addKeyListener('keyup',function(tf,e) {
    packStore.reload({params:{suchen:tf.getValue(),start:0, limit:10}});
});

// Jedesmal, wenn eine Zeile in der Grid ausgewählt wird, werden die Daten im packleistStore neu geladen
// Dieses Listener lauscht auf den 'rowselect' Event, den das Selection Model der Grid aussendet.

PackGrid.getSelectionModel().addKeyListener('rowselect',function() {
    packleistungStore.load({
        params:{ID_Package:PackGrid.getSelectionModel().getSelected().data.ID_Package},
        callback: function(rec,opti,succ) {
            var JSON_Data = PackGrid.getStore().reader.jsonData;
            if(JSON_Data.error==true) { //FEHLERBEHANDLUNG
                Ext.example.msg("Type: "+JSON_Data.type, JSON_Data.message); //weiter Unterscheidung mit Fehler-
            } else {
                if(JSON_Data.total == 0) {
                    Ext.example.msg('Status', 'Noch keine Leistungen für dieses Package angelegt...!');
                }
            }
        }
    });
    PackLeistungGrid.getTopToolbar().setVisible(true);
    PackLeistungGrid.setTitle('Leistungen von: '+PackGrid.getSelectionModel().getSelected().data.packagename);
});

//die ersten 10 Datensätze in den Store laden (Event an die Grid)
pollForChanges(PackGrid,true);

```

Objekt

```

*****/
***** Hier startet der Bereich für die Leistungen im Ost-Bereich des Hauptpanels *****/
*****/
//Der Store packleistungStore versorgt die Grid PackLeistungGrid mit den Einträgen die auf die Packages in der
//Tabelle tpackageleistung bezogen sind.

var packleistungStore = new Ext.data.Store({
    proxy: new Ext.data.HttpProxy({
        url:'index.php',
        method:'POST'
    }),
    remoteSort: true,
    baseParams:{
        cmd: "FillPackLeistungGrid"
    },
    reader: new Ext.data.JsonReader({
        root: 'results',
        totalProperty: 'total'
    },Ext.data.Record.create([
        {name: 'ID_Leistung', type:'int'},
        {name: 'Leistung', type: 'string'},
        {name: 'leistungstag', type: 'int'},
        {name: 'StandardUhrzeit', type: 'string'}
    ])
)
});
packleistungStore.addListener('update',function(st,rec,op) {

    var obj = eval(rec.getChanges());
    if(typeof(obj) == "object")
        for (var j in obj) { //das ist die Schleife aus dem I-Netz! TOLLE SACHE!
            Ext.Ajax.request({ //hier wird der AJAX-Call abgesetzt.
                waitTitle:'Connecting',
                waitMsg:'Sending data...',
                url: 'index.php',
                params: {
                    cmd:"UpdatePackLeistung",
                    key1: 'ID_Package',
                    key2: 'ID_Leistung',
                    keyID1: PackGrid.getSelectionModel().getSelected().data.ID_Package,
                    keyID2: PackLeistungGrid.getSelectionModel().getSelected().data.ID_Leistung,
                    table:"tPackageleistung",
                    field: j,
                    value: obj[j]
                },
                failure:function(response,options){
                    Ext.example.msg('Status', 'not Updated, Errors occurred!');
                },
                success:function(response,options){
                    var responseData = Ext.util.JSON.decode(response.responseText);
                    if(responseData.success == true){
                        packleistungStore.reload();
                        packleistungStore.commitChanges();
                        Ext.example.msg('Status', 'Updated successfully!');
                    } else {
                        Ext.example.msg('Fehler!', 'Leider kein Zugriff');
                    }
                }
            });
        }
    });

// Der Store leistungStore versorgt die ComboBox in der EditorGrid mit den Leistungen
// Es werden alle Leistungen gespeichert!

```

```

var leistungStore = new Ext.data.Store({
    proxy: new Ext.data.HttpProxy({
        url: 'index.php',
        method: 'POST'
    }),
    reader: new Ext.data.JsonReader({
        root: 'results',
        totalProperty: 'total',
        id: 'ID_Leistung'
    },Ext.data.Record.create([
        {name: 'ID_Leistung', type: 'int'},
        {name: 'Leistung', type: 'string'}
    ])
),
baseParams:{
    start: 0,
    cmd: "GetLeistung"
}
});

leistungStore.load();

var PackLeistungGrid = new Ext.grid.EditorGridPanel({
    frame: true,
    enableHdMenu: false,
    border: true,
    width: 300,
    id: 'PackLeistungGrid',
    tbar: new Ext.Toolbar({
        hidden: true,
        items:[{
            tooltip:'Leistung hinzufügen',
            iconCls:'add',
            id:'addLeistung',
            width:50,
            handler: function() {
                //Diese Button wird ein neues Element mit dem Standartnamen 'neues Package' in den packleistungStore
                hinzufügen.
                //ACHTUNG: Dieses Element existiert nur im Store und wird nicht in die Datenbank geschrieben.
                //Erst wenn der Standartname durch wählen in der Combobox abgeändert wird, wird in die Datenbank
                geschrieben
                //konkret passiert das Schreiben in die Datenbank im Listener der am ColumnModel-Editor der ComboBox
                auf den 'change'-Event lauscht.
                var Leistung = PackLeistungGrid.getStore().recordType;
                var l = new Leistung({
                    Leistung: 'neue Leistung'
                });
                PackLeistungGrid.stopEditing();
                packleistungStore.insert(0, l);
                PackLeistungGrid.startEditing(0, 0);
            }
        },{
            tooltip:'Leistung löschen',
            iconCls:'delete',
            id:'delleistung',
            width:50,
            handler: function() {
                Ext.Ajax.request({
                    waitTitle:'Connecting',
                    waitMsg:'Sending data...',
                    url: 'index.php',
                    params: {
                        cmd:"HandlePackLeistung",
                        ID_Package: PackGrid.getSelectionModel().getSelected().data.ID_Package,
                        ID_Leistung: PackLeistungGrid.getSelectionModel().getSelected().data.ID_Leistung,
                        remove: 1
                    }
                });
            }
        }]
    })
});

```

```

    },
    failure:function(response,options){
        Ext.example.msg('Status', 'not Deleted, Errors occurred!');
    },
    success:function(response,options){
        var responseData = Ext.util.JSON.decode(response.responseText);
        if(responseData.success == true) {
            packleistungStore.reload();
            packleistungStore.commitChanges();
            Ext.example.msg('Erfolg', 'Leistung erfolgreich vom Package entfernt!');
        } else {
            Ext.example.msg('Fehler!', 'Diese Leistung ist bereits auf diesem Package zugeordnet!');
        }
    }
});
},
store: packleistungStore,
title: 'Leistung',
clicksToEdit:2,
//Das Columnmodel in der Editorgrid ist etwas erweitert, im Vergleich zu der normalen Grid.
//Es wird zusätzlich ein Editor benötigt, um festzulegen, wie die Änderung der Gridzelle abgehandelt werden soll.
columns: [
{
    width: 30,
    header: 'Leistung',
    dataIndex: 'Leistung',
    editor: new Ext.form.ComboBox({
fieldLabel: 'Leistung*',
        xtype:'combo',
        allowBlank:false,
        editable:false,
        id:'currentCombo',
        forceSelection:true,
        name:'Leistung',
        triggerAction:'all',
        mode: 'local',
        //um alle Leistungen aus der DB in dieser Combobox anzeigen zu können, müssen wir einen Store erstellen
        //und diesen über den CmdGetLeistung-Command befüllen lassen.
        store: leistungStore,
        valueField: 'ID_Leistung',
        displayField: 'Leistung',
        //Dieser Listener lauscht auf den Select-Event der Combobox in der EditorGrid
        //Wenn sich die Combobox ändert, wird die Änderung in die Datenbank geschrieben
        //und der Store für die Jhb_Pack Grid wird neu geladen (damit auch der neue Wert enthalten ist)
        listeners:{
            'select': {
                fn: function(combo, record, index) {

                    //Leider gibt es anscheinend (auch lt. API) keine einfachere Möglichkeit, im select-Event die ID des
                    //Wertes, also des Wertes der vor dem selecten in der Combo gestanden hat, herauszufinden. Den
                    nmt man allerdings.
                    //Mit dieser Schleife durchlaufe ich den gesamten Packagestore und suche nach der
                    enden ID. Die ID wird benötigt, um
                    //in der Datenbanktabelle auch Datensätze updaten zu können. Ohne diese ID kann man den
                    enden Datensatz nicht lokalisieren.
                    var oldValue;
                    //zugegeben, dieses Array sieht echt hässlich aus. Zur Erklärung: im select-Event steht das Objekt
                    ur Verfügung
                    //ausgehend von diesem Objekt kann man sehr viele Dinge erreichen. Z.B. ist combo.store der
                    Combobox stehende Store.
                    //Auf diesem Store liegt das Objekt "data". In diesem "Data"-Objekt liegt jetzt endlich das Array, in
                    esamten Wertpaare als Objekt gespeichert sind.
                    //Der Name dieses Arrays ist "items". Ein Element des Arrays "Items" ist wiederum ein Objekt mit
                    nschaften. Eine Eigenschaft davon heißt "isOn".
                }
            }
        }
    }
}
]
});
```

```

//Und auf dieser "json" Eigenschaft liegen jetzt endgültig die Werte als String, gespeichert.
for (var i = 0; i < combo.store.data.items.length; ++i) {
    if(combo.store.data.items[i].json.Leistung == combo.startValue) {
        oldValue = combo.store.data.items[i].json.ID_Leistung;
    }
}
Ext.Ajax.request({
    waitTitle:'Connecting',
    waitMsg:'Sending data...',
    url: 'index.php',
    params: {
        cmd:"HandlePackLeistung",
        ID_Package: PackGrid.getSelectionModel().getSelected().data.ID_Package,
        ID_Leistung: record.data.ID_Leistung,
        //leistungstag: record.data.leistungstag,
        oldText: combo.startValue,
        oldValue: oldValue
    },
    failure:function(response,options){
        Ext.example.msg('Status', 'not Updated, Errors occurred!');
    },
    success:function(response,options){
        var responseData = Ext.util.JSON.decode(response.responseText);
        if(responseData.success == true){
            packleistungStore.reload();
            packleistungStore.commitChanges();
            Ext.example.msg('Erfolg', 'Leistung erfolgreich dem Package zugeordnet!');
        } else {
            packleistungStore.reload();
            packleistungStore.commitChanges();
            Ext.example.msg('Fehler!', 'Diese Leistung ist bereits auf diesem Package
zugeordnet!');
        }
    }
});
//Der Listener auf dem Blur Event ist notwendig, damit die Combobox keinen Zahlenwert anzeigt, falls man
//in den Eingabemodus wechselt, keine Eingabe tätigt und danach den Eingabemodus wieder verlässt.
//der Blur-Event wird ausgelöst, wenn der Eingabemodus der Combobox beendet wird.
'blur': {
    fn: function() {packleistungStore.reload();}
}
})
},
header: 'Tag',
width: 10,
 dataIndex: 'leistungstag',
 editor: new Ext.form.TextField({
    allowBlank:false
}),
header: 'Uhrzeit',
width: 30,
 dataIndex: 'StandardUhrzeit',
 //editor: new Ext.form.TextField(
 //    allowBlank: false
 //),
 stripeRows: true,
 viewConfig: {
    forceFit:true
}
]

```

```

region:'east',
sm: new Ext.grid.RowSelectionModel({
    singleSelect:true
}),
});

var packageTab = new Ext.Panel({
    title:'Packages',
    closable:true,
    id:'packageTab',
    layout:'border',
    iconCls:'packages',
    items:[PackGrid,PackLeistungGrid]
});

//beim Schließen des Package-Tabs müssen auch sämtliche erzeugte Variablen wieder zerstört werden, damit der Referenz-
Count auf 0 gestellt wird
//damit der Garbage-Collector seine Arbeit erledigen kann.
packageTab.addListener('beforedestroy',function() {
    packStore.destroy();
    packleistungStore.destroy();
    leistungStore.destroy();
});

//hier wird die gerade erstellte GridForm-Kombination an das TabPanel im Center-Bereich geadded
Ext.getCmp('tabcenter').add(packageTab).show();
//doLayout ist notwendig, wenn ein Item in einen bereits gerenderten Container nachträglich
//eingepflegt wird. So wie die grid in das Tabpanel.
Ext.getCmp('bl2_center').doLayout();

} else Ext.getCmp('tabcenter').setActiveTab('packageTab');
}

```

## 10.8 Partner

```

*****
-----Funktion, um einen Tab zum Mainbereich zu adden.
-----
Ich habe mich für diese Variante entschieden, da somit das Initiiieren des Programmes flotter geht.
Beim Instanziieren eines neuen Tabs für das Center-Panel wird der jeweilige Programmteil nachgeladen.
Ich bin der Meinung das diese Herangehensweise den Workflow des Users am wenigsten beeinträchtigt.
Technisch wäre es auch möglich, alle Tabs beim Ext.onReady() Event zu laden. Man hätte dadurch den Vorteil
einer verkürzten Zeit während des Betriebes. Dafür dauert das initiiierende Laden des Programmes um ein viel-
faches länger und nicht benötigte Programmteile werden gleich von Beginn an immer mitgeladen.
*****/
```

```

function addPartnerTab(){
    //zuerst gehört gecheckt, ob es den Tab nicht vielleicht schon gibt.
    //wenn ja, wird er nur mehr focusiert, wenn nein werden die Grids, Forms und Stores erstellt
    //und alles läuft wie geschmiert :)
    if(document.getElementById('partnerTab') == null) {

        // create the data store
        var partnerStore = new Ext.data.Store({
            proxy: new Ext.data.HttpProxy({
                url: 'index.php',
                method: 'POST'
            }),
            baseParams:{
                cmd: "FillPartnerGrid"
            },

```

```

remoteSort:false,
sortInfo: {field:'ID_Partner', direction:'DESC'},
reader: new Ext.data.JsonReader({
    root: 'results',
    totalProperty: 'total'
},Ext.data.Record.create([
    {name: 'ID_Partner', type:'int'},
    {name: 'firmenname', type:'string'},
    {name: 'vorname', type:'string'},
    {name: 'nachname', type:'string'},
    {name: 'adresse', type:'string'},
    {name: 'plz', type:'string'},
    {name: 'ort', type:'string'},
    {name: 'tel', type:'string'},
    {name: 'email', type:'string'},
    {name: 'letzteBearbeitung', type: 'date', dateFormat: 'Y-m-d'}
])
)
});
//das Polling nach neuen Datensätzen soll so lange true sein, bis ein Datensatz in Bearbeitung steht. Während dieser
Bearbeitung
//wird diese Eigenschaft auf false gesetzt und somit ein reload des stores verhindert.
partnerStore._pollEnabled = true;
partnerStore.addListener('update',function(st,rec,op){
    //Hier brauchen wir einen kleinen Trick den ich (Gott sei Dank)
    //aus dem Netz der Netze geklaut habe! Wir müssen ja für jedes upgedatete Feld
    //einen Ajax-Update-Call machen. Ich hatte aber von rec.getChanges() nur ein Objekt
    //indem ALLE geänderten Werte gespeichert waren. Diese kurze aber echt geile Schleife
    //löst mein Problem und setzt für jede Property des Objects einen AJAX-Update-Call ab
    //mit den geänderten Werten! Genial einfach, einfach genial!!
    var obj = eval(rec.getChanges());
    if (typeof(obj) == "object"){
        for (var j in obj) { //das ist die Schleife aus dem I-Netz! TOLLE SACHE!
            Ext.Ajax.request({ //hier wird der AJAX-Call abgesetzt.
                waitTitle:'Connecting',
                waitMsg:'Sending data...',
                url: 'index.php',
                params: {
                    cmd:"Update",
                    key: 'ID_Partner',
                    keyID: rec.data.ID_Partner,
                    table:"tPartner",
                    field: j,
                    value: obj[j]
                },
                failure:function(response,options){
                    Ext.example.msg('Status', 'not Updated, Errors occurred!');
                },
                success:function(response,options){
                    var responseJSON = Ext.decode(response.responseText);

                    if(responseJSON.error==true) { //FEHLERBEHANDLUNG
                        handleException(responseJSON);
                    } else {
                        if(responseJSON.success == true) {
                            partnerStore.reload();
                            partnerStore.commitChanges();
                            Ext.example.msg('Status', 'Updated successfully!');
                        } else Ext.example.msg('Fehler!', 'Leider kein Zugriff');
                    }
                }
            });
        }
    }
});
});
```

```
//Pfad für Druck-Layout-Einstellungs-CSS:
Ext.ux.GridPrinter.stylesheetPath = 'Templates/Application/css/grid_print.css';

var partnerGrid = new Ext.grid.GridPanel({
    title:'Partner',
    frame: false,
    border:true,
    iconCls: 'partner',
    id:'partnerGrid',
    closable:true,
    store: partnerStore,
    columns: [
        {header: "Nr", width: 65, sortable: true, dataIndex: 'ID_Partner'},
        {header: "Firmenname", width: 115, sortable: true, dataIndex: 'firmenname'},
        {header: "Vorname", width: 115, sortable: true, dataIndex: 'vorname'},
        {header: "Nachname", width: 115, sortable: true, dataIndex: 'nachname'},
        {header: "Adresse", width: 90, sortable: true, dataIndex: 'adresse'},
        {header: "PLZ", width: 50, sortable: true, dataIndex: 'plz'},
        {header: "Ort", width: 90, sortable: true, dataIndex: 'ort'},
        {header: "Telefon", width: 115, sortable: true, dataIndex: 'tel'},
        {header: "eMail", width: 115, sortable: true, dataIndex: 'email'},
        {header: "Zuletzt bearb.", width: 90, sortable: true, xtype: 'datecolumn', format: 'd.m.Y', dataIndex: 'letzteBearbeitung'}
    ],
    stripeRows: true,
    viewConfig: {
        forceFit:true
    },
    layout:'fit',
    region:'center',
    sm: new Ext.grid.RowSelectionModel({
        singleSelect:true
    }),
    bbar: new Ext.PagingToolbar({
        pageSize: 10,
        store: partnerStore,
        displayInfo: true,
        plugins: new Ext.ux.ProgressBarPager()
    }),
    tbar: new Ext.Toolbar({
        items: [{tooltip:'Partner anzeigen',
            iconCls:'lesen',
            id:'partnerShowButton',
            handler:function() {
                var rec = partnerGrid.getSelectionModel().getSelected();
                if(rec)
                {
                    //Datensatz Partner Anzeigen
                    var showPartnerFS = new Ext.form.FieldSet({
                        labelAlign: 'left',
                        labelWidth: 150,
                        layout:'form',
                        defaults: {width: 200},
                        bodyStyle:'padding:10px',
                        defaultType: 'textfield',
                        height: 'auto',
                        border: false,
                        items:[{
                            fieldLabel: 'Partner-ID',
                            readOnly:true,
                            disabled:true,
                            name: 'ID_Partner'
                        },
                        {
                            fieldLabel: 'Firma/Partnername*',
                            readOnly:true,
                            name: 'firmenname'
                        }
                    });
                    ...
                }
            }
        }]
    })
});
```

```

        fieldLabel: 'Vorname*',  

        readOnly:true,  

        name: 'vorname'  

    },  

    fieldLabel: 'Nachname*',  

    readOnly:true,  

    name: 'nachname'  

},  

fieldLabel: 'Adresse',  

readOnly:true,  

name: 'adresse'  

},  

fieldLabel: 'PLZ*',  

readOnly:true,  

name: 'plz'  

},  

fieldLabel: 'Ort*',  

readOnly:true,  

name: 'ort'  

},  

fieldLabel: 'Telefon*',  

readOnly:true,  

name: 'tel'  

},  

fieldLabel: 'E-Mail*',  

readOnly:true,  

name: 'email'  

},  

fieldLabel: 'Zuletzt bearbeitet*',  

readOnly:true,  

hideTrigger:true,  

format: 'd.m.Y',  

xtype:'datefield',  

name: 'letzteBearbeitung'  

}]  

});  

var showPartnerFP = new Ext.form.FormPanel({  

    labelAlign: 'left',  

    id:'partnerForm',  

    items: [showPartnerFS],  

    buttons: [{text:'Ok',  

        handler:function() {win.close();}}]  

});  

var win = new Ext.Window({  

    title:'Partnerdaten anzeigen',  

    closable:true,  

    width:420,  

    border:true,  

    resizable:false,  

    modal:true,  

    items: [showPartnerFP]  

});  

showPartnerFP.getForm().loadRecord(partnerGrid.getSelectionModel().getSelected());  

win.show(this);  

} else {Ext.example.msg('Fehler', 'Bitte wählen Sie einen Datensatz aus!');}  

},  

width:50  

},  

tooltip:'Partner bearbeiten',  

iconCls:'schreiben',  

id:'PartnerEditButton',  

handler:function() {  

    //das Polling des Stores wird ausgeschalten, solange bis die Bearbeitung des Datensatzes abgeschlossen ist.  

    partnerStore._pollEnabled = false;  

    //der in der Grid selektierte Record aus dem Store wird vollständig in der Variable rec gespeichert, und in  

    das FormPanel zum Bearbeiten geladen.
}

```

```

var rec = partnerGrid.getSelectionModel().getSelected();
//wenn diese if-Abfrage false ergibt ist kein Datensatz in der Grid selektiert und es wird ein entsprechender
Hinweis ausgegeben.
if(rec)
{
    //Datensatz Ändern
    var writePartnerFS = new Ext.form.FieldSet({
        labelAlign: 'left',
        labelWidth: 150,
        layout:'form',
        defaults: {width: 200},
        bodyStyle:'padding:10px',
        defaultType: 'textfield',
        height: 'auto',
        border: false,
        items:[{
            fieldLabel: 'Partner-ID',
            disabled:true,
            name: 'ID_Partner'
        },{
            fieldLabel: 'Firmenname',
            name: 'firmenname'
        },{
            fieldLabel: 'Vorname',
            allowBlank:false,
            name: 'vorname'
        },{
            fieldLabel: 'Nachname',
            allowBlank:false,
            name: 'nachname'
        },{
            fieldLabel: 'Adresse',
            allowBlank:false,
            name: 'adresse'
        },{
            fieldLabel: 'PLZ',
            allowBlank:false,
            name: 'plz'
        },{
            fieldLabel: 'Ort',
            allowBlank:false,
            name: 'ort'
        },{
            fieldLabel: 'Telefon',
            name: 'tel'
        },{
            fieldLabel: 'E-Mail',
            name: 'email'
        }
    ],
    buttons: [
        {
            text:'save',
            iconCls:'save',
            formBind: true,
            handler: function() {
                writePartnerFP.getForm().updateRecord(partnerGrid.getSelectionModel().getSelected());
                //das Polling für den Store wieder aktivieren...
                partnerStore._pollEnabled = true;
                win.close();
            }
        },
        {
            text: 'cancel',
            handler: function() {
                //das Polling für den Store wieder aktivieren...
                partnerStore._pollEnabled = true;
                win.close();
            }
        }
    ]
});

```

```

        });
    });

    var writePartnerFP = new Ext.form.FormPanel({
        labelAlign: 'left',
        id:'PartnerForm',
        monitorValid:true,
        items: [writePartnerFS]
    });

    var win = new Ext.Window({
        title:'Partner-Datensatz ändern',
        closable:true,
        width:420,
        border:true,
        resizable:false,
        modal:true,
        items: [writePartnerFP]
    });
    writePartnerFP.getForm().loadRecord(partnerGrid.getSelectionModel().getSelected());
    win.show(this);
} else {Ext.example.msg("Fehler", 'Bitte wählen Sie einen Datensatz aus!');}
},
width:50
},{

    tooltip:'neuer Partner',
    iconCls:'add',
    id:'PartnerNewButton',
    handler: function() {
        //Datensatz Neuer Partner
        var neuPartnerFS = new Ext.form.FieldSet({
            labelAlign: 'left',
            labelWidth: 150,
            layout:'form',
            defaults: {width: 200},
            bodyStyle:'padding:10px',
            defaultType: 'textfield',
            height: 'auto',
            border: false,
            items:[{
                fieldLabel: 'Firmenname',
                name: 'firmenname'
            },{
                fieldLabel: 'Vorname',
                allowBlank:false,
                name: 'vorname'
            },{
                fieldLabel: 'Nachname',
                allowBlank:false,
                name: 'nachname'
            },{
                fieldLabel: 'Adresse',
                allowBlank:false,
                name: 'adresse'
            },{
                fieldLabel: 'PLZ',
                allowBlank:false,
                name: 'plz'
            },{
                fieldLabel: 'Ort',
                allowBlank:false,
                name: 'ort'
            },{
                fieldLabel: 'Telefon',
                name: 'tel'
            },
            {
                fieldLabel: 'E-Mail',
                name: 'email'
            }
        });
    }
});
}

```

```

                name: 'email'
            }];
        });
    var neuPartnerFP = new Ext.form.FormPanel({
        labelAlign: 'left',
        id:'PartnerForm',
        monitorValid:true,
        url:'index.php',
        items: [neuPartnerFS],
        buttons: [{
            text: 'Save',
            formBind: true,
            iconCls:'save',
            handler: function(){
                neuPartnerFP.getForm().submit({
                    method:'POST',
                    waitTitle:'Connecting',
                    waitMsg:'Sending data...',
                    params: {
                        cmd:'InsertPartner'
                    },
                    success:function(form, action){
                        if(action.result.success == true) {
                            Ext.example.msg('Status', 'Saved successfully!');
                            partnerStore.reload();
                        } else {
                            Ext.example.msg('Fehler', 'Nicht gespeichert!!');
                        }
                        win.close();
                    },
                    failure:function(form, action){
                        if(action.failureType == 'server'){
                            Ext.example.msg('Not Saved. Server-Error Occured.');
                        }else{
                            Ext.example.msg('Warning!', 'Server is unreachable at the Moment: ' +
action.responseText);
                        }
                        win.close();
                    }
                });
            }
        },{
            text: 'Cancel',
            handler: function() {win.close();}
        });
    });

    var win = new Ext.Window({
        title:'Partner-Datensatz hinzufügen',
        closable:true,
        width:420,
        border:true,
        resizable:false,
        modal:true,
        items: [neuPartnerFP]
    });
    win.show(this);
},
width:50
},{
    tooltip:'Partner löschen',
    iconCls:'delete',
    id:'PartnerDeleteButton',
    handler: function() {
        var rec = partnerGrid.getSelectionModel().getSelected();
        if(rec)
        {

```

```

Ext.Msg.show({
    title:'wirklich löschen?',
    msg: 'Sie sind dabei, einen Partner zu löschen. Wollen Sie diesen Partner wirklich löschen?',
    buttons: Ext.Msg.YESNO,
    fn: function(buttonID) {
        if(buttonID=='yes') {
            Ext.Ajax.request({           //hier wird der AJAX-Call abgesetzt.
                waitTitle:'Connecting',
                waitMsg:'updating data...',
                url: 'index.php',
                params: {
                    cmd:"Update",
                    key: 'ID_Partner',
                    keyID: partnerGrid.getSelectionModel().getSelected().data.ID_Partner,
                    table:"tPartner",
                    field: 'aktiv',
                    value: 0
                },
                failure:function(response,options){
                    Ext.example.msg('Status', 'Not deleted! Error Occured!');
                },
                success:function(response,options){
                    var responseData = Ext.util.JSON.decode(response.responseText);
                    if(responseData.success == true) {
                        partnerStore.reload();
                        partnerStore.commitChanges();
                        Ext.example.msg('Status', 'Deleted successfully!');
                    } else {
                        Ext.example.msg('Fehler', 'Keine Berechtigung');
                    }
                }
            });
            partnerStore.reload();
        }
    },
    icon: Ext.MessageBox.QUESTION
});
} else {Ext.example.msg('Fehler', 'Bitte wählen Sie einen Datensatz aus!');},
iconCls:'delete',
width:50
},{
    tooltip:'Partner Mail',
    iconCls:'email',
    id:'KundenMailButton',
    width:50,
    handler:function() {
        if(partnerGrid.getSelectionModel().hasSelection()) {
            var MailFS = new Ext.form.FieldSet({
                labelAlign: 'left',
                labelWidth: 150,
                layout:'form',
                title: 'Mail',
                bodyStyle:'padding:10px',
                defaults: {width: 200},
                defaultType: 'textfield',
                border: false,
                items:[{
                    fieldLabel: 'Betreff',
                    xtype: 'textfield',
                    name: 'subject'
                },{
                    fieldLabel: 'Text',
                    xtype: 'textarea',
                    name: 'body'
                }]
            });
        }
    }
}

```

```

    });

    var neuMailFP = new Ext.form.FormPanel({
        labelAlign: 'left',
        monitorValid:true,
        url:'index.php',
        items: [MailFS],
        buttons: [{
            text: 'Send',
            formBind: true,
            iconCls:'email',
            handler: function(){
                neuMailFP.getForm().submit({
                    method:'POST',
                    waitTitle:'Connecting',
                    waitMsg:'Sending data...',
                    params: {
                        cmd:'Mail',
                        to:partnerGrid.getSelectionModel().getSelected().data.email
                    },
                    success:function(form, action) {
                        Ext.example.msg('Success', 'Mail erfolgreich gesendet!');
                        win.close();
                    },
                    failure:function(form, action) {
                        var responseJSON = Ext.decode(action.response.responseText);
                        handleException(responseJSON);
                    }
                });
            }
        },{
            text: 'Cancel',
            handler: function() {win.close();}
        }];
    });

    var win = new Ext.Window({
        title:'Partner-Mail',
        closable:true,
        width:420,
        border:true,
        resizable:false,
        modal:true,
        items: [neuMailFP]
    });
    win.show(this);
} else {
    Ext.example.msg('Fehler', 'Kein Datensatz ausgewählt.');
}
},
width:50
},
tooltip:'Tabelle drucken',
iconCls:'print',
id:'DruckenButton',
width:50,
handler:function() {
    //nachträglich eininstalliertes extjs-Plugin
    Ext.ux.GridPrinter.print(partnerGrid);
}
},'->{
    xtype:'textfield',
    name:'searchfield',
    id:'partnersuchfeld',
    enableKeyEvents:true,
    emptyText:'Partner suchen...'
}
}

```

```

        });
    });

    //dieser Event wird bei jedem Keyup-Event des Suchfeldes ausgelöst. Wir machen nach jedem Buchstaben einen AJAX-Request
    und
    //schließen das Ergebnis dieses neuen Requests an den Store. Hoffentlich ist der Datenbankzugriff schnell genug damit sich
    dieser geile
    //moderne Effekt den ich mir vorstelle auch zu 100% entfaltet... -> wenn dieser Kommentar in der Endversion noch enthalten
    ist, dann gehts :)
    Ext.getCmp('partnersuchfeld').addListener('keyup',function(tf,e) {
        partnerStore.reload({params:{suchen:tf.getValue(),start:0, limit:10}});
    });

    //beim Schließen des Partner-Tabs müssen auch sämtliche erzeugte Variablen wieder zerstört werden, damit der Referenz-
    Count auf 0 gestellt wird
    //damit der Garbage-Collector seine Arbeit erledigen kann.
    partnerGrid.addListener('beforedestroy',function() {
        partnerStore.destroy();
    });
    //die ersten 10 Datensätze in den Store laden (Event an die Grid)
    pollForChanges(partnerGrid,true);
    //hier wird die gerade erstellte GridForm-Kombination an das TabPanel im Center-Bereich geadded
    Ext.getCmp('tabcenter').add(partnerGrid).show();
    //doLayout ist notwendig, wenn ein Item in einen bereits gerenderten Container nachträglich
    //eingepflegt wird. So wie die grid in das Tabpanel.
    Ext.getCmp('bl2_center').doLayout();
} else Ext.getCmp('tabcenter').setActiveTab('partnerGrid');
}

```

## 10.9 User

```

*****-----Funktion, um einen Tab zum Mainbereich zu adden.-----*****
Ich habe mich für diese Variante entschieden, da somit das Initialisieren des Programmes flotter geht.
Beim Instanziieren eines neuen Tabs für das Center-Panel wird der jeweilige Programmteil nachgeladen.
Ich bin der Meinung das diese Herangehensweise den Workflow des Users am wenigsten beeinträchtigt.
Technisch wäre es auch möglich, alle Tabs beim Ext.onReady() Event zu laden. Man hätte dadurch den Vorteil
einer verkürzten Zeit während des Betriebes. Dafür dauert das initialisierende Laden des Programmes um ein viel-
faches länger und nicht benötigte Programmteile werden gleich von Beginn an immer mitgeladen.
*****-----*****

```

```

function addUserTab(){
    //zuerst gehört gecheckt, ob es den Tab nicht vielleicht schon gibt.
    //wenn ja, wird er nur mehr focusiert, wenn nein werden die Grids, Forms und Stores erstellt
    if(document.getElementById("userGrid") == null) {

        // create the data store
        var userStore = new Ext.data.Store({
            proxy: new Ext.data.HttpProxy({
                url: 'index.php',
                method: 'POST'
            }),
            remoteSort:true,
            baseParams:{
                cmd: "FillUserGrid"
            },
            sortInfo: {field:'ID_User', direction:'ASC'},
            reader: new Ext.data.JsonReader({
                root: 'results',
                totalProperty: 'total'
            },Ext.data.Record.create([
                {name: 'ID_User', type:'int'},
                {name: 'vorname', type:'string'},
                {name: 'nachname', type:'string'},
                {name: 'email', type:'string'},
                {name: 'adresse', type:'string'}
            ])
        });
    }
}

```

```

        {name: 'plz', type:'string'},
        {name: 'ort', type:'string'},
        {name: 'right_User', type:'boolean'},
        {name: 'right_Anfrage', type:'boolean'},
        {name: 'right_Buchung', type:'boolean'},
        {name: 'right_Leistung', type:'boolean'},
        {name: 'right_Package', type:'boolean'},
        {name: 'right_Partner', type:'boolean'},
        {name: 'right_Auswertungen', type:'boolean'},
        {name: 'right_Kunde', type:'boolean'},
        {name: 'right_Jugendherbergen', type:'boolean'}
    ],
    }
);
//das Polling nach neuen Datensätzen soll so lange true sein, bis ein Datensatz in Bearbeitung steht. Während dieser
Bearbeitung
//wird diese Eigenschaft auf false gesetzt und somit ein reload des stores verhindert.
userStore.addListener('update',function(st,rec,op){
    //Hier brauchen wir einen kleinen Trick den ich (Gott sei Dank)
    //aus dem Netz der Netze geklaut habe! Wir müssen ja für jedes upgedatede Feld
    //einen Ajax-Update-Call machen. Ich hatte aber von rec.getChanges() nur ein Objekt
    //indem ALLE geänderten Werte gespeichert waren. Diese kurze aber echt geile Schleife
    //löst mein Problem und setzt für jede Property des Objects einen AJAX-Update-Call ab
    //mit den geänderten Werten! Genial einfach, einfach genial!!
    var obj = eval(rec.getChanges());
    if(typeof(obj) == "object"){
        for (var j in obj) { //das ist die Schleife aus dem I-Netz! TOLLE SACHE!
            Ext.Ajax.request({ //hier wird der AJAX-Call abgesetzt.
                waitTitle:'Connecting',
                waitMsg:'Sending data...',
                url: 'index.php',
                params: {
                    cmd:"Update",
                    key: 'ID_User',
                    keyID: rec.data.ID_User,
                    table:"tUser",
                    field: j,
                    value: obj[j]
                },
                failure:function(response,options){
                    Ext.example.msg('Status', 'not Updated, Errors occurred!');
                },
                success:function(response,options){
                    var responseJSON = Ext.decode(response.responseText);

                    if(responseJSON.error==true) { //FEHLERBEHANDLUNG
                        handleException(responseJSON);
                    } else {
                        if(responseJSON.success == true) {
                            userStore.reload();
                            userStore.commitChanges();
                            Ext.example.msg('Status', 'Updated successfully!');
                        } else Ext.example.msg('Fehler!', 'Leider kein Zugriff');
                    }
                }
            });
        }
    });
}

var checkJHB = new Ext.grid.CheckColumn({header: 'JHB', dataIndex: 'right_Jugendherbergen', width: 55});
var checkUser = new Ext.grid.CheckColumn({header: 'User', dataIndex: 'right_User', width: 55});
var checkAnfrage = new Ext.grid.CheckColumn({header: 'Anfrage', dataIndex: 'right_Anfrage', width: 55});
var checkBuchung = new Ext.grid.CheckColumn({header: 'Buchung', dataIndex: 'right_Buchung', width: 55});
var checkLeistung = new Ext.grid.CheckColumn({header: 'Leistung', dataIndex: 'right_Leistung', width: 55});
var checkPackage = new Ext.grid.CheckColumn({header: 'Package', dataIndex: 'right_Package', width: 55});
var checkPartner = new Ext.grid.CheckColumn({header: 'Partner', dataIndex: 'right_Partner', width: 55});
var checkAuswertung = new Ext.grid.CheckColumn({header: 'Ausw.', dataIndex: 'right_Auswertungen', width: 55});

```

```

var checkKunde = new Ext.grid.CheckColumn({header: 'Kunde', dataIndex: 'right_Kunde', width: 55});

var userGrid = new Ext.grid.EditorGridPanel({
    title:'User',
    plugins: [checkUser, checkAnfrage, checkBuchung, checkLeistung, checkPackage, checkPartner,
              checkAuswertung,checkKunde,checkJHB],
    clicksToEdit:1,
    frame: false,
    border:true,
    iconCls: 'user',
    id:'userGrid',
    closable:true,
    store: userStore,
    columns: [
        {header: "Nr", width: 35, sortable: false, dataIndex: 'ID_User'},
        {header: "Vorname", width: 90, sortable: false, dataIndex: 'vorname'},
        {header: "Nachname", width: 90, sortable: false, dataIndex: 'nachname'},
        {header: "eMail", width: 140, sortable: false, dataIndex: 'email'},
        checkUser, checkAnfrage, checkBuchung, checkLeistung, checkPackage, checkPartner,
        checkAuswertung,checkKunde,checkJHB
    ],
    stripeRows: true,
    viewConfig: {
        forceFit:true
    },
    layout:'fit',
    region:'center',
    sm: new Ext.grid.RowSelectionModel({
        singleSelect:true
    }),
    bbar: new Ext.PagingToolbar({
        pageSize: 10,
        store: userStore,
        displayInfo: true,
        plugins: new Ext.ux.ProgressBarPager()
    }),
    tbar: new Ext.Toolbar({
        items: [{
            tooltip:'neuer User',
            iconCls:'add',
            id:'UserNewButton',
            width:50,
            handler:function() {

                //Datensatz Neuer User
                var neuUserFS = new Ext.form.FieldSet({
                    labelAlign: 'left',
                    labelWidth: 150,
                    layout:'form',
                    defaults: {width: 200},
                    bodyStyle:'padding:10px',
                    defaultType: 'textfield',
                    height: 'auto',
                    border: false,
                    items:[{
                        fieldLabel: 'Vorname',
                        allowBlank:false,
                        name: 'vorname'
                    },{
                        fieldLabel: 'Nachname',
                        allowBlank:false,
                        name: 'nachname'
                    },{
                        fieldLabel: 'eMail',
                        allowBlank:false,
                        name: 'email'
                    }]
                });
            }
        }]
    })
});

```

```

        fieldLabel: 'Adresse',
        allowBlank:false,
        name: 'adresse'
    },
    fieldLabel: 'PLZ',
    allowBlank:false,
    name: 'plz'
},
fieldLabel: 'Ort',
allowBlank:false,
name: 'ort'
}
});
var neuUserFP = new Ext.form.FormPanel({
labelAlign: 'left',
id:'UserForm',
monitorValid:true,
url:'index.php',
items: [neuUserFS],
buttons: [
text: 'Save',
formBind: true,
iconCls:'save',
handler: function(){
    neuUserFP.getForm().submit({
method:'POST',
waitTitle:'Connecting',
waitMsg:'Sending data...',
params: {
cmd:'InsertUser'
},
success:function(form, action){
if(action.result.success == true) {
Ext.example.msg('Status', 'Saved successfully!');
userStore.reload();
} else {
Ext.example.msg('Fehler', 'Nicht gespeichert!!');
}
win.close();
},
failure:function(form, action){
if(action.failureType == 'server'){
Ext.example.msg('Not Saved. Server-Error Occured.');
}else{
Ext.example.msg('Warning!', 'Server is unreachable at the Moment: ' +
action.response.responseText);
}
win.close();
}
});
},
text: 'Cancel',
handler: function() {win.close();}
]);
};

var win = new Ext.Window({
title:'User-Datensatz hinzufügen',
closable:true,
width:420,
border:true,
resizable:false,
modal:true,
items: [neuUserFP]
});
win.show(this);

```

```

        }
    },
    tooltip:'User löschen',
    iconCls:'delete',
    id:'UserDeleteButton',
    width:50,
    handler:function() {
        var rec = userGrid.getSelectionModel().getSelected();
        if(rec)
        {
            Ext.Msg.show({
                title:'wirklich löschen?',
                msg: 'Sie sind dabei, einen User zu löschen. Wollen Sie diesen User wirklich löschen?',
                buttons: Ext.Msg.YESNO,
                fn: function(buttonID) {
                    if(buttonID=='yes') {
                        Ext.Ajax.request({ //hier wird der AJAX-Call abgesetzt.
                            waitTitle:'Connecting',
                            waitMsg:'updating data...',
                            url: "index.php",
                            params: {
                                cmd:"Update",
                                key: 'ID_User',
                                keyID: userGrid.getSelectionModel().getSelected().data.ID_User,
                                table:"tUser",
                                field: 'aktiv',
                                value: 0
                            },
                            failure:function(response,options){
                                Ext.example.msg('Status', 'Not deleted! Error Occured!');
                            },
                            success:function(response,options){
                                var responseData = Ext.util.JSON.decode(response.responseText);
                                if(responseData.success == true) {
                                    userStore.reload();
                                    userStore.commitChanges();
                                    Ext.example.msg('Status', 'Deleted successfully!');
                                } else {
                                    Ext.example.msg('Fehler', 'Keine Berechtigung');
                                }
                            }
                        });
                        userStore.reload();
                    }
                },
                icon: Ext.MessageBox.QUESTION
            });
        } else {Ext.example.msg('Fehler', 'Bitte wählen Sie einen Datensatz aus!');}
    },
    tooltip:'User bearbeiten',
    iconCls:'schreiben',
    id:'WriteUserButton',
    width:50,
    handler:function() {
        //das Polling des Stores wird ausgeschalten, solange bis die Bearbeitung des Datensatzes abgeschlossen ist.
        userStore._pollEnabled = false;
        //der in der Grid selektierte Record aus dem Store wird vollständig in der Variable rec gespeichert, und in
        das FormPanel zum Bearbeiten geladen.
        var rec = userGrid.getSelectionModel().getSelected();
        //wenn diese if-Abfrage false ergibt ist kein Datensatz in der Grid selektiert und es wird ein entsprechender
        Hinweis ausgegeben.
        if(rec)
        {
            //Datensatz Ändern
            var writeUserFS = new Ext.form.FieldSet({
                labelAlign: 'left',

```

```

labelWidth: 150,
layout:'form',
defaults: {width: 200},
bodyStyle:'padding:10px',
defaultType: 'textfield',
title:'Userdaten',
height: 'auto',
border: false,
items:[{
    fieldLabel: 'User-ID',
    disabled:true,
    name: 'ID_User'
 },{
    fieldLabel: 'Vorname',
    allowBlank:false,
    name: 'vorname'
 },{
    fieldLabel: 'Nachname',
    allowBlank:false,
    name: 'nachname'
 },{
    fieldLabel: 'eMail',
    allowBlank:false,
    name: 'email'
 },{
    fieldLabel: 'Adresse',
    allowBlank:false,
    name: 'adresse'
 },{
    fieldLabel: 'PLZ',
    allowBlank:false,
    name: 'plz'
 },{
    fieldLabel: 'Ort',
    allowBlank:false,
    name: 'ort'
 }],
});
var writeUserFP = new Ext.form.FormPanel({
    labelAlign: 'left',
    id:'UserForm',
    monitorValid:true,
    items: [writeUserFS],
    buttons: [{ text:'save',
        iconCls:'save',
        formBind: true,
        handler: function() {
            writeUserFP.getForm().updateRecord(userGrid.getSelectionModel().getSelected());
                //das Polling für den Store wieder aktivieren...
                userStore._pollEnabled = true;
                win.close();
            }
        },{
            text: 'cancel',
            handler: function() {
                //das Polling für den Store wieder aktivieren...
                userStore._pollEnabled = true;
                win.close();
            }
        }
    ]
});
var win = new Ext.Window({
    title:'Userdaten ändern',
    closable:true,

```

```

        width:420,
        border:true,
        resizable:false,
        modal:true,
        items: [writeUserFP]
    });
    writeUserFP.getForm().loadRecord(userGrid.getSelectionModel().getSelected());
    win.show(this);
} else {Ext.example.msg("Fehler", 'Bitte wählen Sie einen Datensatz aus!');}

}

}, {
    tooltip:'User Mail',
    iconCls:'email',
    id:'KundenMailButton',
    width:50,
    handler:function() {

        if(userGrid.getSelectionModel().hasSelection()) {

            var MailFS = new Ext.form.FieldSet({
                labelAlign: 'left',
                labelWidth: 150,
                layout:'form',
                title:'Mail',
                bodyStyle:'padding:10px',
                defaults: {width: 200},
                defaultType: 'textfield',
                height: 'auto',
                border: false,
                items:[{
                    fieldLabel: 'Betreff',
                    xtype: 'textfield',
                    name: 'subject'
                },{
                    fieldLabel: 'Text',
                    xtype: 'textarea',
                    name: 'body'
                }]
            });

            var neuMailFP = new Ext.form.FormPanel({
                labelAlign: 'left',
                monitorValid:true,
                url:'index.php',
                items: [MailFS],
                buttons: [{
                    text: 'Send',
                    formBind: true,
                    iconCls:'email',
                    handler: function(){
                        neuMailFP.getForm().submit({
                            method:'POST',
                            waitTitle:'Connecting',
                            waitMsg:'Sending data...',
                            params: {
                                cmd:'Mail',
                                to:userGrid.getSelectionModel().getSelected().data.email
                            },
                            success:function(form, action) {
                                Ext.example.msg('Success', 'Mail erfolgreich gesendet!');
                                win.close();
                            },
                            failure:function(form, action) {
                                var responseJSON = Ext.decode(action.response.responseText);
                                handleException(responseJSON);
                            }
                        });
                    }
                }]
            });
        }
    }
}

```

```

        });
    },
    text: 'Cancel',
    handler: function() {win.close();}
});
});

var win = new Ext.Window{
    title:'User-Mail',
    closable:true,
    width:420,
    border:true,
    resizable:false,
    modal:true,
    items: [neuMailFP]
});
win.show(this);
} else {
    Ext.example.msg('Fehler', 'Kein Datensatz ausgewählt.');
}
},
width:50
});
};

//beim Schließen des User-Tabs müssen auch sämtliche erzeugte Variablen wieder zerstört werden, damit der Referenz-Count
auf 0 gestellt wird
//damit der Garbage-Collector seine Arbeit erledigen kann.
userGrid.addListener('beforedestroy',function() {
    userStore.destroy();
});
//die ersten 10 Datensätze in den Store laden (Event an die Grid)
//kein Polling bei den Usern....
userStore.load({params:{start:0, limit:10}});
//hier wird die gerade erstellte GridForm-Kombination an das TabPanel im Center-Bereich geadded
Ext.getCmp('tabcenter').add(userGrid).show();
//doLayout ist notwendig, wenn ein Item in einen bereits gerenderten Container nachträglich
//eingepflegt wird. So wie die grid in das Tabpanel.
Ext.getCmp('bl2_center').doLayout();
} else Ext.getCmp('tabcenter').setActiveTab('userGrid');
}
}

```

## 11 Application

Das Template, welches den kompletten JS-Sourcecode für den Aufbau und die Darstellung der Applikation im Client-Browser zu ermöglichen. Alle Darstellungsformen befinden sich hier und werden mithilfe des JS-Frameworks ExtJS dargestellt. Diese Applikation erzeugt asynchrone AJAX-Calls an den Server, genauer gesagt werden damit die Commands am Server angesprochen.

### 11.1 Javascript

```
*****
Hauptbereich, Aufbau des Viewportes und Verschachtelung des Grundlayouts

Structure in a nutshell: Sehr grob ausgedrückt, wird das Layout wie folgt aufgebaut -
Der Viewport (das ist die Trägerkomponente des gesamten Dokumentes [document.body]) wird in ein Border-Layout
unterteilt. Dieses Borderlayout beinhaltet einen Nordbereich, indem das Logo steht, einen Westbereich für
die Navigation und natürlich einen Center-Bereich. Das Zentrum dieses Borderlayouts ist für sich betrachtet
nochmals ein eigenes Border Layout. Diesmal mit einem Süd-Bereich für zusätzliche Infos und wieder einem
```

Center-Bereich. In diesen Center-Bereich kommt ein TabPanel (eine Trägerkomponente für Registerkarten). Nach betätigen eines Buttons in der Navigation wird eine Funktion aufgerufen, die für den aufgerufenen Part alle notwendigen Stores und Panels generiert, die Steuerelemente zum Aufrufen der PHP-Commands bereitstellt und das Gesamtergebnis dieser Funktion in den TabPanel als neue Registerkarte einhängt. Nach dem Schließen einer Registerkarte wird der komplette Speicherbereich der Stores und Panels wieder freigegeben.

In der Application.js steht der strukturelle Aufbau des User Interfaces. Hier ist der Aufbau des Gerüsts und der Navigation das Hauptaugenmerk. Die Steuerelemente der einzelnen Navigationspunkte finden Sie im jeweiligen "NAME\_DES\_MENÜPUNKTES.JS"-File

Dieses Programm benötigt zum Laufen:

- PHP5.X od. höher
- mySQL 5.X od. höher (nur mit InnoDB)

```
******/
```

//Die onReady-function wird ausgeführt, wenn der Aufbau der Seite abgeschlossen wird.  
//Deswegen ist es der ideale Ansatzpunkt, hier den Javascript-Applikationscode auszuführen.

```
Ext.onReady(function(){
```

```
//Diese clientseitige Try-Catch Fehlerverwaltung soll nur Fehler abfangen, die im Javascript-Code  
//auftreten könnten. Alle Fehler, die aus asynchronen Ajax-Calls heraus auftreten und im PHP (am Server)  
//den Ursprung haben, werden gesondert in den Callback bzw. Success-Funktionen der Requests gehandhabt.  
//Hilfsfunktion zur Ausgabe einer Server-Exception ist die Funktion handleException();
```

```
try {
```

```
//Für die Darstellung des Logos unterscheiden wird die User des IE6 zum Rest der Userschaft.  
//Im IE6 können PNG-Grafiken mit Alphawert nicht richtig dargestellt werden. In diesem Fall  
//verwenden wir einfach ein gif-Image.
```

```
var logo;  
if(/MSIE 6/.test(navigator.userAgent)) {  
    logo=;  
} else {  
    logo=;  
}
```

```
//Das Blank_Image ist ein 1px großes Gif mit dem Alpha-Wert 100%.  
//Es wird für die korrekte Darstellung der Trigger im aktuellen extjs-Framework benötigt  
//Anscheinend ist hier noch ein Fehler in dieser Version des Frameworks versteckt.  
//Durch manuelles Setzen der Konstante BLANK_IMAGE_URL wird dieses Fehlverhalten umgangen.  
Ext.BLANK_IMAGE_URL = 'Resources/ext-3.0.3/resources/images/default/s.gif';
```

```
-----NORTHBOX -----  
//Ein leeres Panel, das das Logo und den Head-Balken beeinhaltet.  
//-----
```

```
var bl1_northbox = new Ext.Panel({  
    region:'north',  
    height:60,  
    border:false,  
    //margins: '0 0 0 0',  
    id:'northbox',  
    html:logo,  
    layout:'fit',  
    collapsible:false,  
    collapsed:false  
});
```

```
//die Mailadresse des Anmelders, sowie die Rechte ins JS holen  
//um damit den Aufbau des Interfaces zu beeinflussen.  
Ext.Ajax.request({  
    url: 'index.php',  
    params:{  
        cmd:'GetRights'  
    },  
    method: 'POST',  
    //Alles "Rechteabhängige" muss in dieser Success-Funktion gecoded werden, da
```

```

//der Call ja asynchron passiert und der Aufbau des restlichen Programmes bereits
//beendet ist wenn der Response mit den Rights vom Server hier ankommt. Deshalb kann man
//beim Aufbau außerhalb dieser Success-Funktion auch nicht auf das Objekt responseJSON Zugreifen.
//Dieses Objekt wird erst erzeugt, wenn der Response vom Server ankommt und die Success-Funktion auslöst.
success:function(response,options){
    //Requesten der Rechte und speichern im Objekt "responseJSON"
    var responseJSON = Ext.decode(response.responseText);

    if(responseJSON.error==true) { //FEHLERBEHANDLUNG
        handleException(responseJSON);
    } else{
        Ext.getCmp('btnuser').setVisible(responseJSON.results[0].right_user);
        Ext.getCmp('btنانfrage').setVisible(responseJSON.results[0].right_anfrage);
        Ext.getCmp('btnbuchung').setVisible(responseJSON.results[0].right_buchung);
        Ext.getCmp('btnleistungen').setVisible(responseJSON.results[0].right_leistung);
        Ext.getCmp('btnpackage').setVisible(responseJSON.results[0].right_package);
        Ext.getCmp('btंpartner').setVisible(responseJSON.results[0].right_partner);
        Ext.getCmp('btناuswertungen').setVisible(responseJSON.results[0].right_auswertungen);
        Ext.getCmp('btंkunden').setVisible(responseJSON.results[0].right_kunde);
        Ext.getCmp('btंjugendherbergen').setVisible(responseJSON.results[0].right_jugendherbergen);
        //doLayout ist notwendig, da wir bereits fertig gerenderte
        bl1_westpanel.doLayout();
        bl2_south.add({html:'<p>Welcome, '+responseJSON.results[0].username+'!</p><p>Thank you for using Smart
Booking. If you experience any inconvenience or miss-behavior, feel free to contact our Support Hotline: 0901/XXX XX XX</p>'});
        bl2_south.doLayout();

        //Je nach Berechtigung wird ein Tab als Standart vorgeladen.
        if(responseJSON.results[0].right_anfrage) addAnfrageTab();
        else if(responseJSON.results[0].right_buchung) addBuchungTab();
        else if(responseJSON.results[0].right_kunde) addKundenTab();
        else if(responseJSON.results[0].right_auswertungen) addAuswertungenTab();
        else if(responseJSON.results[0].right_leistung) addLeistungenTab();
        else if(responseJSON.results[0].right_partner) addPartnerTab();
        else if(responseJSON.results[0].right_package) addPackagesTab();
        else if(responseJSON.results[0].right_user) addUserTab();
        else if(responseJSON.results[0].right_jugendherbergen) addJugendherbergenTab();
    }
},
failure: function() { //Schwerer Fehler trat auf.
    Ext.Msg.alert('Fehler', 'Server wurde nicht erreicht! Bitte versuchen Sie es später nochmals.');
}
});

//-----WEST-Panel-----
//Der Navigations-Bereich der Applikation. Die Buttons passen sich dynamisch der Höhe des Browserfensters an.
//Es gibt einen Bottom-Toolbar, der den Logout-Link beinhaltet, sowie ein Toolbar-Menü mit den Optionen.
//Hier brauchen wir, abweichend zur Einleitung am Seitenbeginn, doch etwas Geschäftslogik:
//Die Buttons des Optionsmenüs sind hier ausprogrammiert.
//-----

//Hier wird ein neuer V-Typ abgeleitet, der für die Passwort-Validation
//zuständig ist (z.B. ob 2 Passwörter zusammenstimmen wie
//bei der Passwort-Änderung) --> Siehe Beispiel ADVANCED VALIDATION
Ext.apply(Ext.form.Validators, {
    password : function(val, field) {
        if (field.initialPasswordField) {
            var pwd = Ext.getCmp(field.initialPasswordField);
            return (val == pwd.getValue());
        }
        return true;
    },
    passwordText : 'Passwords do not match'
});

//Und hier die Anlage des West-Panels:
var bl1_westpanel= new Ext.Panel({
    region:'west',

```

```

title:'Navigation',
id:'bl1_westpanel',
collapsible:false,
//im bbar liegt ein Menü und ein Logout-Button
bbar:new Ext.Toolbar{
    items: [{
        text:'Logout',
        iconCls:'logout',
        width:80,
        id:'btnLogout',
        handler:function() {
            bl1_northbox.destroy();
            bl1_westpanel.destroy();
            bl2_south.destroy();
            tabcenter.destroy();
            bl1_centerpanel.destroy();
            bl2_center.destroy();
            Ext.Ajax.request({
                url: 'index.php',
                params:{},
                cmd:'Logout'
            },
            method: 'POST',
            success:function(response,options){
                //Wenn der Logout-Command ohne Probleme durchgelaufen ist, ist der Success-Code auf true.
                //danach wird die Seite neu geladen und es wird der Anmeldeschirm angezeigt.
                if(Ext.decode(response.responseText).success == true) {
                    self.location='index.php';
                } else Ext.Msg.alert('Logout Error', 'Sie wurden nicht vollständig ausgeloggt. Bitte Seite erneut laden und nochmals ausloggen!');
            },
            failure:function() {
                Ext.Msg.alert('Logout Error', 'Der Server wurde nicht erreicht. Bitte versuchen Sie später nochmals sich auszuloggen.');
            }
        });
    },
    text:'Options',
    iconCls: 'options',
    width:80,
    menu: new Ext.menu.Menu{
        items: [{
            iconCls:'email',
            text: 'eMail Adresse ändern',
            handler: function() {
                //Beim Click auf diesen Button wird ein Fenster geöffnet,
                //von wo aus die neue Mailadresse eingegeben werden kann
                //vorher muss noch das aktuelle Passwort angegeben werden...
                var wndChangeMail = new Ext.Window{
                    title:'eMail Adresse ändern',
                    closable:true,
                    width:420,
                    border:true,
                    resizable:false,
                    modal:true,
                    items: [
                        new Ext.form.FormPanel({
                            labelAlign: 'left',
                            monitorValid:true,
                            id:'changeEmail',
                            items: [
                                new Ext.form.FieldSet({
                                    labelAlign: 'left',
                                    labelWidth: 150,
                                    layout:'form',
                                    defaults: {width: 200},

```

```

        bodyStyle:'padding:10px',
        defaultType: 'textfield',
        height: 'auto',
        border: false,
        items:[{
            fieldLabel: 'Passwort',
            inputType:'password',
            name: 'pwd'
        },{
            fieldLabel: 'neue e-Mail Adresse',
            vtype:'email',
            name: 'value'
        }]
    },
    //Hier die Buttons des Formpanels "changeEmail"
    buttons: [{{
        //dieser Button macht den Form.submit();
        text:'save',
        iconCls:'save',
        formBind: true,
        handler: function() {
            //Diese Funktion setzt überittelt asynchron die Daten
            //des Formulares an die main.php, von wo aus die Daten in der DB
            //geändert werden.
            Ext.getCmp('changeEmail').getForm().submit({
                method: 'POST',
                url:'index.php',
                params: {
                    field: 'email',
                    cmd: 'UpdateUser'
                },
                success: function(form,action) {
                    wndChangeMail.close();
                    Ext.example.msg('Success', 'eMail erfolgreich geändert!');
                },
                failure: function(form,action) {
                    if(action.result.pwd==true)
                        Ext.example.msg('Stop', 'Keine Änderung möglich. Möglicherweise
existiert diese Adresse bereits?');
                    else
                        Ext.example.msg('Stop', 'Passwort leider falsch!');
                }
            });
        }
    }{
        //dieser Button schließt einfach nur das Window, ohne etwas zu bewirken.
        text: 'cancel',
        handler: function() {wndChangeMail.close();}
    }]
},
]);
});
```

wndChangeMail.show(**this**);

},

iconCls:'password',
text: 'Passwort ändern',
handler: **function()** {
//Beim Click auf diesen Button wird ein Fenster geöffnet,
//von wo aus das neue Passwort mit 2maliger Eingabe geändert wird.
//Vorher muss noch das aktuelle Passwort angegeben werden...
**var** wndChangePwd = **new** Ext.Window{

title:'Passwort ändern',
closable:true,
width:420,

```

border:true,
resizable:false,
modal:true,
items: [
    new Ext.form.FormPanel({
        labelAlign: 'left',
        monitorValid:true,
        id:'changePassword',
        items: [
            new Ext.form.FieldSet({
                labelAlign: 'left',
                labelWidth: 150,
                layout:'form',
                defaults: {width: 200},
                bodyStyle:'padding:10px',
                defaultType: 'textfield',
                height: 'auto',
                border: false,
                items:[{
                    fieldLabel: 'altes Passwort',
                    inputType:'password',
                    name: 'pwd'
                },{
                    fieldLabel: 'neues Passwort',
                    inputType:'password',
                    id:'newPwd',
                    name: 'newPwd'
                },{
                    fieldLabel: 'neues Passwort wiederh.',
                    vtype: 'password',
                    initialPassField: 'newPwd',
                    inputType:'password',
                    name: 'value'
                }]
            })
        ],
        //Hier die Buttons des Formpanels "changePassword"
        buttons: [{
            //dieser Button macht den Form.submit();
            text:'save',
            iconCls:'save',
            formBind: true,
            handler: function() {
                //Diese Funktion setzt überittelt asynchron die Daten
                //des Formulares an die main.php, von wo aus die Daten in der DB
                //geändert werden.
                Ext.getCmp('changePassword').getForm().submit({
                    method: 'POST',
                    url:'index.php',
                    params: {
                        field: 'passwort',
                        cmd: 'UpdateUser'
                    },
                    success: function(form,action) {
                        wndChangePwd.close();
                        Ext.example.msg('Success', 'Passwort erfolgreich geändert!');
                    },
                    failure: function(form,action) {
                        if(action.result.pwd==true)
                            Ext.example.msg('Stop', 'PWD gleich, nicht upgedatet!');
                        else
                            Ext.example.msg('Stop', 'Passwort leider falsch!');
                    }
                });
            }
        }],
        //dieser Button schließt einfach nur das Window, ohne etwas zu bewirken.
    });
}

```

```
        text: 'cancel',
        handler: function() {wndChangePwd.close();}
    }
]
})
});
 wndChangePwd.show(this);
}
}
}
},
border:true,
layout: {
    type:'vbox',
    padding:'20',
    align:'stretch'
},
defaults:{margins:'0 0 15 0'},
width:165,
items:[ new Ext.Button({
    text: 'Anfragen',
    iconCls: 'anfrage',
    id:'btinanfrage',      //dieses Objekt hat keinen Namen und muss über diese
                           //ID angesprochen werden. Ext.getCmp('btinanfrage')
    scale:'medium',
    hidden:true,
    //Beim Klick auf den Button wird eine Funktion ausgelöst, die
    //einen neuen Tab im Centertab-Panel generiert. Diese Funktion
    //liegt im ordner /javascript/addXXX.js
    handler: addAnfrageTab,
    flex:1
}), new Ext.Button({
    text: 'Buchungen',
    id:'btnbuchung',
    iconCls: 'buchungen',
    scale:'medium',
    hidden:true,
    handler: addBuchungTab,
    flex:1
}), new Ext.Button({
    text: 'Kunden',
    iconCls: 'kunden',
    id:'btnkunden',
    scale:'medium',
    hidden:true,
    handler: addKundenTab,
    flex:1
}), new Ext.Button({
    text: 'Statistik',
    id:'bttnauswertungen',
    iconCls: 'auswertungen',
    scale:'medium',
    hidden:true,
    handler: addAuswertungenTab,
    flex:1
}), new Ext.Button({
    text: 'Leistungen',
    id:'bttnleistungen',
    iconCls: 'leistungen',
    scale:'medium',
    hidden:true,
    handler: addLeistungenTab,
    flex:1
}), new Ext.Button({
    text: 'Partner'
```

```

        iconCls: 'partner',
        id:'btncitizen',
        scale:'medium',
        hidden:true,
        handler: addCitizenTab,
        flex:1
    }), new Ext.Button({
        text: 'Packages',
        id:'btncitizen',
        iconCls: 'packages',
        scale:'medium',
        hidden:true,
        handler: addPackagesTab,
        flex:1
    }), new Ext.Button({
        text: 'User',
        iconCls: 'user',
        id:'btncitizen',
        scale:'medium',
        hidden:true,
        handler: addUserTab,
        flex:1
    }), new Ext.Button({
        text: 'Junge Hotels',
        id:'btncitizen',
        iconCls: 'herbergen',
        scale:'medium',
        hidden:true,
        handler: addJugendherbergenTab,
        flex:1
    })
});
```

```

//-----CENTER-Panel-----
//Das Zentrum der Anwendung (unterhalb des Logos und rechts von der Navigation).
//Dieses Zentrum ist abermals unterteilt in ein Zentrum und in einen Südbereich.
//Im Süden steht ein Bereich für zusätzliche Informationen. Im Zentrum ist jetzt endgültig der
//tatsächliche Arbeitsbereich der Anwendung in Form des TabPanels. In dieses TabPanel werden
//während der Laufzeit Tabs hinein-erzeugt, wenn auf den entsprechenden Navigationsbutton gedrückt wird.
//-----
var bl2_south = new Ext.Panel({
    region:'south',
    border:true,
    title:'Additional Info',
    collapsible: true,
    margins: '0 2 0 0',
    collapsed: false,
    split: true,
    height: 70,
    layout:'fit'
});

var tabcenter = new Ext.TabPanel({
    border:false,
    resizeTabs:true,
    id:'tabcenter',
    minTabWidth: 115,
    tabWidth:135,
    enableTabScroll:true,
    defaults: {autoScroll:true},
    tabPosition:'top'
});

var bl2_center = new Ext.Panel({
    region:'center'.
```

```

        id:'bl2_center',
        layout:'fit',
        border:false,
        items:[tabcenter]
    });

    var bl1_centerpanel = new Ext.Panel({
        region:'center',
        margins: '0 0 0 2',
        border:false,
        layout:'border',
        items:[bl1_south, bl2_center]
    });

//-----DER VIEWPORT. DIE TRAEGERKOMPONENTE ANGEWENDET AUF document.body-----
var viewport=new Ext.Viewport({
    layout:'border',
    items:[bl1_northbox, bl1_centerpanel, bl1_westpanel]
});

//Quick-Tipps ermöglichen und den Singleton einige Einstellungen mitgeben :)
Ext.QuickTips.init();
Ext.apply(Ext.QuickTips.getQuickTip(), {
    maxWidth: 250,
    minWidth: 100,
    showDelay: 10,
    trackMouse: false
});

//-----Ausblenden des Loading-Prozessbar nachdem alles geladen wurde-----
Ext.get('loading').fadeOut({remove: false});

}

//*********************************************************************FEHLERBEHANDLUNG*****/
catch(err) {
    Ext.example.msg("Leider ist ein Fehler aufgetreten: " + err);
}

});

function handleException(err) {
    Ext.example.msg("Type: "+err.type, err.message); //weiter Unterscheidung mit Fehler-Objekt
}

```

## 11.2 CSS-Styles

```

/* Allgemein gültige CSS */

html, body {
    font:normal 11px tahoma;
    margin:0;
    padding:0;
    border:0 none;
    overflow:hidden;
    height:100%;
    scrollbar-base-color:#DCE7EE;
    scrollbar-3dlight-color:#e4e8ed;
    scrollbar-arrow-color:#000000;
    scrollbar-darkshadow-color:#666666;
    scrollbar-face-color:#C1D3EA;
}

```

```

scrollbar-highlight-color:#ffffff;
scrollbar-track-color:#DFE9F5;
}

p {
    margin:5px;
}

#northbox div{
    background-color: #a3a3a3;
    background-image: url(../../../../Resources/images/headerback.jpg);
    background-repeat:repeat-x;
    background-position: top right;
    text-align:right;
    text-align:right;
}

.x-grid3-row-alt {
    background:url(../../../../Resources/images/grey.png);
    background-color:none;
}

.x-grid3-row-over {
    /*border-color:#666;*/
    /*background-color:#efefef !important;*/
    background-image:url(../../../../Resources/images/yellow.png)!important;
}

.x-grid3-row td {
    line-height: 16px;
}

.x-grid3-row {
    border-color:#dddddd;
    border-style:solid;
    border-top-color:#ffff;
}

.x-grid3-row-selected {
    /*background-color: #ffffcf !important;*/
    background-image:url(../../../../Resources/images/black.png)!important;
    /*border-color:#DDD;*/
}

.lightred {
    background-image:url(../../../../Resources/images/red.png);
}

.lightgreen {
    background-image:url(../../../../Resources/images/green.png);
}

#bl2_center .x-grid3-scroller {
    background-image: url(../../../../Resources/images/hg.gif);
}

#bl1_westpanel td.x-btn-mc {
    text-align: left;
}

```

```

/* Icon-CLS - Also Pics für die Icons */
.anfrage {
    background-image:url(../../../../Resources/images/attachment.png) !important;
}
.buchungen {
    background-image:url(../../../../Resources/images/folder.png) !important;
}
.kunden {
    background-image:url(../../../../Resources/images/smiley.png) !important;
}

.auswertungen {
    background-image:url(../../../../Resources/images/bar_chart.png) !important;
}

.leistungen {
    background-image:url(../../../../Resources/images/shopping_cart.png) !important;
}

.partner {
    background-image:url(../../../../Resources/images/group.png) !important;
}

.packages {
    background-image:url(../../../../Resources/images/archive.png) !important;
}

.user {
    background-image:url(../../../../Resources/images/user.png) !important;
}

.herbergen {
    background-image:url(../../../../Resources/images/home.png) !important;
}

.logout {
    background-image:url(../../../../Resources/images/exit.png) !important;
}

.add {
    background-image:url(../../../../Resources/images/add.png) !important;
}
.delete {
    background-image:url(../../../../Resources/images/delete.png) !important;
}
.deny {
    background-image:url(../../../../Resources/images/deny.png) !important;
}
.grant {
    background-image:url(../../../../Resources/images/grant.png) !important;
}
.save {
    background-image:url(../../../../Resources/images/save.png) !important;
}
.archiv {
    background-image:url(../../../../Resources/images/archiv.png) !important;
}
.lesen {
    background-image:url(../../../../Resources/images/lesen.png) !important;
}
.schreiben {
    background-image:url(../../../../Resources/images/schreiben.png) !important;
}

```

```
.options {
    background-image:url(../../Resources/images/options.png) !important;
}
.email {
    background-image:url(../../Resources/images/email.png) !important;
}
.password {
    background-image:url(../../Resources/images/password.png) !important;
}

.print {
    background-image:url(../../Resources/images/print.gif) !important;
}
.redboldtext {
    color:red;
    font-weight: bold !important;
}
.redtext {
    color:red;
}
.greentext {
    color:green;
}

/* CSS für den Loading-Indicator - Also für die Lademaske */

#loading{
position:absolute;
left:40%;
top:40%;
padding:3px;
z-index:20001;
height:auto;
border:1px solid #aaa;
}

#loadingDiv {
margin-right:8px;
float:left;
vertical-align:top;
}

#loading .loading-indicator{
background:white;
color:#444;
font:bold 13px tahoma;
padding:10px;
margin:0;
height:auto;
}

#loading-msg {
font: normal 10px tahoma;
}
```

```
html,body,div,dl,dt,dd,ul,ol,li,h1,h2,h3,h4,h5,h6,pre,form,fieldset,input,p,blockquote,th,td{margin:0;padding:0;}
img,body,html{border:0;}
address,caption,cite,code,dfn,em,strong,th,var{font-style:normal;font-weight:normal;}
ol,ul {list-style:none;}caption,th {text-align:left;}h1,h2,h3,h4,h5,h6{font-size:100%}q:before,q:after{content:"";}

table {
    width: 100%;
```

```

text-align: left;
font-size: 11px;
font-family: arial;
border-collapse: collapse;
}

table th {
padding: 4px 3px 4px 5px;
border: 1px solid #d0d0d0;
border-left-color: #eee;
background-color: #ebeded;
}

table td {
padding: 4px 3px 4px 5px;
border-style: none solid solid;
border-width: 1px;
border-color: #ebeded;
}

```

### 11.3 Template

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" />
<title><?php print $this->title.'.'.$this->version; ?></title>

<link rel="stylesheet" type="text/css" href="Resources/ext-3.0.3/resources/css/ext-all-notheme.css" />
<link rel="stylesheet" type="text/css" href="Resources/ext-3.0.3/src/shared/examples.css" />

<!-- TEMPLATE AUSWÄHLEN (Overrides der ext-all.css) -->
<link rel="stylesheet" type="text/css" href="Resources/ext-3.0.3/resources/css/xtheme-tp.css" />

<!-- Einbinden der eigenen CSS-Klassen (und der Overrides) -->
<link rel="stylesheet" type="text/css" href="Templates/Application/css/application.css" />
</head>

<body>

<div id="loading">
<div class="loading-indicator">
    
    <?php print $this->title.'.'.$this->version; ?><br />
    <span id="loading-msg">Styles und Images werden geladen...</span>
</div>
</div>

<!-- Einbinden der Framework-Javascripts-->
<script type="text/javascript">document.getElementById('loading-msg').innerHTML = 'Laden der Grund-Bibliotheken...';</script>
<script type="text/javascript" src="Resources/ext-3.0.3/adapter/ext/ext-base.js"></script>
<script type="text/javascript">document.getElementById('loading-msg').innerHTML = 'Laden der UI Basis-Komponenten...';</script>
<script type="text/javascript" src="Resources/ext-3.0.3/ext-all.js"></script>

<!-- Einbinden der noch benötigten Framework-Extensions -->
<script type="text/javascript">document.getElementById('loading-msg').innerHTML = 'Laden der UI-Extensions...';</script>
<script type="text/javascript" src="Resources/ext-3.0.3/src/ux/SlidingPager.js"></script>
<script type="text/javascript" src="Resources/ext-3.0.3/src/ux/SliderTip.js"></script>
<script type="text/javascript" src="Resources/ext-3.0.3/src/ux/CheckColumn.js"></script>
<script type="text/javascript" src="Resources/ext-3.0.3/src/ux/ProgressBarPager.js"></script>
<script type="text/javascript" src="Resources/ext-3.0.3/src/ux/PagingMemoryProxy.js"></script>
<script type="text/javascript" src="Resources/ext-3.0.3/src/ux/xdatefield.js"></script>
<script type="text/javascript" src="Resources/ext-3.0.3/src/ux/GridPrinter.js"></script>
<script type="text/javascript" src="Resources/ext-3.0.3/src/ux/RowExpander.js"></script>
<script type="text/javascript" src="Resources/ext-3.0.3/src/shared/examples.js"></script>

```

```
<!-- Einbinden der eigenen Javascripts-->
<script type="text/javascript">document.getElementById('loading-msg').innerHTML = 'Laden der Applikation...';</script>
<script type="text/javascript" src="Templates/Application/javascript/Application.js"></script>
<script type="text/javascript" src="Templates/Application/javascript/pollForChanges.js"></script>
<script type="text/javascript" src="Templates/Application/javascript/addanfrage.js"></script>
<script type="text/javascript" src="Templates/Application/javascript/adbuchung.js"></script>
<script type="text/javascript" src="Templates/Application/javascript/addauswertungen.js"></script>
<script type="text/javascript" src="Templates/Application/javascript/addleistungen.js"></script>
<script type="text/javascript" src="Templates/Application/javascript/addpartner.js"></script>
<script type="text/javascript" src="Templates/Application/javascript/addpackage.js"></script>
<script type="text/javascript" src="Templates/Application/javascript/adduser.js"></script>
<script type="text/javascript" src="Templates/Application/javascript/addjugendherberge.js"></script>
<script type="text/javascript" src="Templates/Application/javascript/addkunden.js"></script>

</body>
</html>
```

## 11.4 Serverpolling for Changes

```
*****
* POLL FOR CHANGES IST FÜR DAS SERVER-POLLING VON NEUEN
* INFORMATIONEN ZUSTÄNDIG
*****
```

```
//diese Funktion schickt alle 15sek einen asynchronen Request an den Server und reloaded den store
//das bedeutet, ein neuer Record in der Datenbank wird mit max. 15sek Verzögerung in den store geladen und angezeigt.
//Für die Verwendung ist gedacht, diese Funktion statt der initialen store.load Function zu benützen.
function pollForChanges(grid,first) {
    //beim ersten Aufruf dieser Funktion, wird der store natürlich nicht reloaded, sondern es erfolgt der initiale LOAD-Aufruf.
    if (first == true) {
        grid.getStore().load({
            params:{start:0, limit:10},
            //Fehlerbehandlung für den ersten, initialisierenden Datenabruf in die Tabellen
            //Wenn es zu einer Exception kommt, wird die entsprechende handle-Exception Funktion aufgerufen
            callback: function(rec,opti,succ) {
                if (grid.getStore() != null) {
                    var JSON_Data = grid.getStore().reader.jsonData;
                    if(JSON_Data.error==true) { //FEHLERBEHANDLUNG
                        Ext.example.msg("Type: "+JSON_Data.type, JSON_Data.message); //weiter Unterscheidung mit Fehler-
Objekt
                    } else {
                        if(JSON_Data.total == 0) {
                            Ext.example.msg('Status', 'Noch keine Daten in diesem Bereich vorhanden.');
                        }
                    }
                }
            });
            setTimeout(function(){pollForChanges(grid,false);}, 15000);
        }
    } else {
        //wenn ein Datensatz bearbeitet wird, darf nicht gepolled werden, da ein reload des stores während einige Felder 'dirty' sind
        //kein Update in der Datenbank der schmutzigen Felder auslöst. Während des Bearbeitens eines Datensatzes habe ich die
selbst
        // eingebrauchte Eigenschaft _pollEnabled im Store auf false gesetzt, erst danach kann wieder gepolled werden.
        if (grid.getStore() != null) {
            if(grid.getStore()._pollEnabled == true) {
                grid.getStore().reload({
                    callback: function(rec,opti,succ) {
                        //unabhängig von success oder failure wird in 15sek wieder ein call abgesetzt.
                        //Bei den reloads im 15 Sekunden-Takt wollen wir keine Fehlerbehandlung.
                        //Auch wenn der Server nicht erreichbar war oder ein anderer Fehler auftrat
                        //wollen wir den User damit nicht behelligen. Das aktuell halten der Datensätze
                        //in den Stores soll unauffällig und im Hintergrund passieren.
                        setTimeout(function(){pollForChanges(grid,false);}, 15000);
                    }
                });
            }
        }
    }
}
```

```
        });
    });
}); //else setTimeout(function(){pollForChanges(grid,false);}, 15000);
}
```

## 12 Frontend

## 12.1 Javascript

```
*****
In dieser Javascript-Datei steht der Code zum Aufbau für das Frontend.
*****
```

```
Ext.onReady(function(){

//Für die Darstellung des Logos unterscheiden wird die User des IE6 zum Rest der Userschaft.
//Im IE6 können PNG-Grafiken mit Alphawert nicht richtig dargestellt werden. In diesem Fall
//verwenden wir einfach ein gif-Image.
var logo;
if(/MSIE 6/.test(navigator.userAgent)) {
    logo='';
} else {
    logo='';
}

Ext.BLANK_IMAGE_URL = 'Resources/ext-3.0.3/resources/images/default/s.gif';

var norden = new Ext.Panel({
    region:'north',
    height:60,
    border:false,
    id:'northbox',
    html:logo,
    layout:'fit',
    collapsible:false,
    collapsed:false
});

var randomValue1 = Math.round(Math.random() * 10);
var randomValue2 = Math.round(Math.random() * 10);
var targetValue = randomValue1 + randomValue2;

var anfrageFieldset1 = new Ext.form.FieldSet({
    labelWidth: 150,
    columnWidth:.5,
    layout: 'form',
    border:false,
    bodyStyle:'padding:20px 5px 5px 5px',
    items: [
        fieldLabel: 'Jugendherberge*',
        xtype:'combo',
        allowBlank:false,
        id:'JhbComboID',
        editable:false,
        forceSelection:true,
        triggerAction:'all',
        mode: 'local',
        store: new Ext.data.Store{}
```

```

proxy: new Ext.data.HttpProxy({
    url: 'index.php',
    method: 'POST'
}),
reader: new Ext.data.JsonReader({
    root: 'results',
    totalProperty: 'total',
    id: 'ID_Jhb'
},Ext.data.Record.create([
    {name: 'ID_Jhb', type: 'int'},
    {name: 'jhb', type: 'string'}
]),
baseParams:{
    cmd: "GetJhb"
},
autoLoad: true
}),
valueField: 'ID_Jhb',
displayField: 'jhb',
name: 'jhb',
anchor:'80%'
},
fieldLabel: 'Paket*',
xtype:'combo',
allowBlank:false,
editable:false,
id:'PaketComboID',
forceSelection:true,
triggerAction:'all',
mode: 'local',
store: new Ext.data.Store({
    proxy: new Ext.data.HttpProxy({
        url: 'index.php',
        method: 'POST'
    }),
    reader: new Ext.data.JsonReader({
        root: 'results',
        totalProperty: 'total',
        id: 'ID_Package'
    },Ext.data.Record.create([
        {name: 'ID_Package', type: 'int'},
        {name: 'packagename', type: 'string'}
    ]),
    baseParams:{
        cmd: "GetPackage"
    }
}),
valueField: 'ID_Package',
displayField: 'packagename',
name: 'packagename',
hiddenName: 'ID_Package',
hiddenId:'hiddenPackageIdCombo',
name:'ID_Package',
anchor:'80%'
},
fieldLabel: 'Termin Anreisetag*',
allowBlank:false,
format: 'd.m.Y',
id:'anreisedatum',
minValue: new Date(),
xtype:'datefield',
name: 'termin',
anchor:'80%'
},
fieldLabel: 'Kategorie*',
xtype:'combo',
allowBlank:false,

```

```

        editable:false,
        forceSelection:true,
        triggerAction:'all',
        mode: 'local',
        store: new Ext.data.ArrayStore({
            fields: ['KategorieText'],
            data: [['Privat'],['Schule'],['Firma']]
        }),
        valueField: 'KategorieText',
        displayField: 'KategorieText',
        name: 'kategorie',
        anchor:'80%'
    },
    xtype:'textfield',
    fieldLabel: 'Organisation',
    name: 'organisation',
    anchor:'95%'
},
{
    xtype:'textfield',
    fieldLabel: 'Vorname*',
    allowBlank:false,
    name: 'vorname',
    anchor:'95%'
},
{
    xtype:'textfield',
    fieldLabel: 'Nachname*',
    allowBlank:false,
    name: 'nachname',
    anchor:'95%'
},
{
    xtype:'textfield',
    fieldLabel: 'Adresse*',
    allowBlank:false,
    name: 'adresse',
    anchor:'95%'
},
{
    xtype:'textfield',
    fieldLabel: 'PLZ*',
    allowBlank:false,
    name: 'plz',
    anchor:'95%'
},
{
    xtype:'textfield',
    fieldLabel: 'Ort*',
    allowBlank:false,
    name: 'ort',
    anchor:'95%'
},
{
    xtype:'textfield',
    fieldLabel: 'Telefon',
    name: 'tel',
    anchor:'95%'
},
{
    xtype:'textfield',
    fieldLabel: 'Fax',
    name: 'fax',
    anchor:'95%'
},
{
    xtype:'textfield',
    fieldLabel: 'eMail',
    vtype:'email',
    name: 'email',
    anchor:'95%'
},
{
    xtype:'textfield',
    fieldLabel: 'Telefon Ansprechpartner*',
    allowBlank:false,
    name: 'telefonAnsprechpartner'
}
]

```

```

        name: 'telAP',
        anchor:'95%'
    }{
        xtype:'textfield',
        fieldLabel: 'eMail Ansprechpartner*',
        id:'mailAnsprechperson',
        allowBlank:false,
        vtype:'email',
        name: 'emailAP',
        anchor:'95%'
    }]
});

Ext.getCmp('JhbComboID').addListener('select',function() {
    Ext.getCmp('PaketComboID').getStore().setBaseParam('ID_Jhb', Ext.getCmp('JhbComboID').getValue())
    Ext.getCmp('PaketComboID').getStore().load();
    Ext.getCmp('PaketComboID').setValue("");
});
}

var anfrageFieldset2 = new Ext.form.FieldSet({
    labelWidth: 150,
    columnWidth:.5,
    layout: 'form',
    border:false,
    bodyStyle:'padding:20px 5px 5px 5px',
    items: [
        xtype:'textfield',
        fieldLabel: 'Anzahl Erwachsene',
        name: 'erwachsene',
        anchor:'50%'
    },{
        xtype:'textfield',
        fieldLabel: 'Anzahl Kinder',
        name: 'kinder',
        anchor:'50%'
    },{
        xtype:'textfield',
        fieldLabel: 'Anzahl Weiblich',
        name: 'female',
        anchor:'50%'
    },{
        xtype:'textfield',
        fieldLabel: 'Anzahl Männlich',
        name: 'male',
        anchor:'50%'
    },{
        xtype:'textfield',
        fieldLabel: 'Anzahl Vegetarier',
        name: 'vegetarier',
        anchor:'50%'
    },{
        fieldLabel: 'religiöse Speisevorschriften',
        xtype: 'textarea',
        name: 'relVorschriften',
        anchor:'95%'
    },{
        fieldLabel: 'Allergien',
        xtype: 'textarea',
        name: 'allergien',
        anchor:'95%'
    },{
        fieldLabel: 'Bemerkung',
        xtype: 'textarea',
        name: 'bemerkung',
        anchor:'95%'
    }
]);

```

```

xtype:'textfield',
fieldLabel:<b style="color:#ff0000">Bitte addieren Sie '+randomValue1+' + '+randomValue2+'</b>',
anchor:'60%',
id:'targetValue'
});

};

var anfrageForm = new Ext.form.FormPanel({
labelAlign:'center',
id:'anfrageForm',
region:'center',
frame:true,
border:true,
layout:'column',
url:'index.php',
buttons: [{{
text: 'Senden',
id:'sendButton',
formBind: true,
handler: function(){
if(Ext.getCmp('targetValue').getValue() != targetValue) {
Ext.example.msg('Sicherheitsfrage', 'Bitte tragen Sie das richtige Ergebnis der Rechnung ein!!');
}
else {
anfrageForm.getForm().submit({
method:'POST',
waitTitle:'Connecting',
waitMsg:'Sending data...',
params: {
cmd:'insertAnfrage',
ersatztermin: Ext.getCmp('anreisedatum').getValue()
},
success:function(form, action){
if(action.result.success == true) {
Ext.Msg.alert('Anfrage gesendet', 'Vielen Dank! Ihre Anfrage wurde erfolgreich übermittelt. Sie
werden in Kürze von uns kontaktiert.');
anfrageForm.getForm().reset();
} else {
Ext.example.msg('Fehler', 'Nicht gespeichert!!');
}
},
},

}),
};

Ext.Ajax.request({ //hier wird der AJAX-Call abgesetzt.
waitTitle:'Connecting',
waitMsg:'Sending data...',
url: 'index.php',
params: {
cmd:"Mail",
to: "roland@widmayer.at",
body: "Ein Konsument hat eine Anfrage an Sie gerichtet! Bitte prüfen Sie in der Applikation Smart
Booking die neuen Anfragen und ergreifen Sie eine geeignete Maßnahme.",
subject:"Neues Ticket eingelangt"
},
failure:function(response,options){
Ext.example.msg('Fehler', 'Server wurde nicht erreicht! Bitte versuchen Sie es später nochmals.');
},
success:function(response,options){
var responseJSON = Ext.decode(response.responseText);

if(responseJSON.error==true) { //FEHLERBEHANDLUNG
handleException(responseJSON);
} else {
}
}
}
);

```

```

        if(responseJSON.success == true) {
            Ext.example.msg('Status', 'Sachbearbeiter wurde informiert!');
        } else Ext.example.msg('Fehler!', 'Leider kein Zugriff');
    }
});

Ext.Ajax.request({ //hier wird der AJAX-Call abgesetzt.
    waitTitle:'Connecting',
    waitMsg:'Sending data...',
    url: 'index.php',
    params: {
        cmd:"Mail",
        to: Ext.getCmp('mailAnsprechperson').getValue(),
        body: "Vielen Dank, Ihre Anfrage für den "+Ext.getCmp('anreisedatum').value+" wurde erfolgreich übermittelt. \nUnsere Sachbearbeiter werden sich so bald wie möglich bei Ihnen melden. \n\nFreundliche Grüße, \nIhr NOEJHW Team",
        subject:"Anfrage erfolgreich übermittelt"
    },
    failure:function(response,options){
        Ext.example.msg('Fehler', 'Server wurde nicht erreicht! Bitte versuchen Sie es später nochmals.');
    },
    success:function(response,options){
        var responseJSON = Ext.decode(response.responseText);

        if(responseJSON.error==true) { //FEHLERBEHANDLUNG
            handleException(responseJSON);
        } else {
            if(responseJSON.success == true) {
                Ext.example.msg('Status', 'Sie haben ein Mail zur Bestätigung erhalten.');
            } else Ext.example.msg('Fehler!', 'Leider kein Zugriff');
        }
    });
}

text: 'Reset',
handler: function() {
    anfrageForm.getForm().reset();
}
},
items:[anfrageFieldset1,anfrageFieldset2]
});

var viewport=new Ext.Viewport ({
    layout:'border',
    items:[norden, anfrageForm]
});

//-----Ausblenden des Loading-Prozessbar nachdem alles geladen wurde-----
Ext.get('loading').fadeOut({remove: false});

};

function handleException(err) {
    Ext.example.msg("Type: "+err.type, err.message); //weiter Unterscheidung mit Fehler-Objekt
}

```

## 12.2 CSS-Styles

```
html, body {
```

```

font:normal 11px tahoma;
margin:0;
padding:0;
border:0 none;
overflow:hidden;
height:100%;

scrollbar-base-color:#DCE7EE;
scrollbar-3dlight-color:#e4e8ed;
scrollbar-arrow-color:#000000;
scrollbar-darkshadow-color:#666666;
scrollbar-face-color:#C1D3EA;
scrollbar-highlight-color:#ffffff;
scrollbar-track-color:#DFE9F5;

}

p {
margin:5px;
}

#northbox div {
background-color: #a3a3a3;
background-image: url(../../Resources/images/headerback.jpg);
background-repeat:repeat-x;
background-position: top right;
text-align:right;
text-align:right;
}

/* CSS für den Loading-Indicator - Also für die Lademaske */

#loading{
position:absolute;
left:40%;
top:40%;
padding:3px;
z-index:20001;
height:auto;
border:1px solid #aaa;
}

#loadingDiv {
margin-right:8px;
float:left;
vertical-align:top;
}

#loading .loading-indicator{
background:white;
color:#444;
font:bold 13px tahoma;
padding:10px;
margin:0;
height:auto;
}

#loading-msg {
font: normal 10px tahoma;
}

```

## 12.3 Template

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" />
<title><?php print $this->title.' '.$this->version; ?></title>
    <link rel="stylesheet" type="text/css" href="Resources/ext-3.0.3/resources/css/ext-all-notheme.css" />
    <link rel="stylesheet" type="text/css" href="Resources/ext-3.0.3/src/shared/examples.css" />
    <link rel="stylesheet" type="text/css" href="Resources/ext-3.0.3/resources/css/xtheme-tp.css" />
    <link rel="stylesheet" type="text/css" href="Templates/Frontend/css/frontend.css" />
</head>
<body>
<div id="loading">
    <div class="loading-indicator">
        
        <?php print $this->title.' '.$this->version; ?><br />
        <span id="loading-msg">Styles und Images werden geladen...</span>
    </div>
</div>

<!-- Einbinden der Framework-Javascripts-->
<script type="text/javascript">document.getElementById('loading-msg').innerHTML = 'Laden der Grund-Bibliotheken...';</script>
<script type="text/javascript" src="Resources/ext-3.0.3/adapter/ext/ext-base.js"></script>
<script type="text/javascript">document.getElementById('loading-msg').innerHTML = 'Laden der UI Basis-Komponenten...';</script>
<script type="text/javascript" src="Resources/ext-3.0.3/ext-all.js"></script>
<script type="text/javascript" src="Resources/ext-3.0.3/src/ux/xdatefield.js"></script>
<script type="text/javascript" src="Resources/ext-3.0.3/src/shared/examples.js"></script>
<script type="text/javascript" src="Templates/Frontend/javascript/Frontend.js"></script>
</body>
</html>
```

## 13 Login Form

### 13.1 Javascript

```
*****
In dieser Javascript-Datei steht der Code zum Aufbau für das Login-Fenster. Die Funktion Write_Log_In in der login.php ruft diese Javascript zu Hilfe um das Loginfenster darzustellen.
*****
```

```
Ext.onReady(function(){
    Ext.QuickTips.init();

    var login = new Ext.FormPanel({
        labelWidth:60,
        standardSubmit: true,
        bodyStyle:'padding:10px;',
        //monitorValid:true,
        width:307,
        autoHeight:true,
        defaultType:'textfield',
        items:[
            {
                xtype:'box', //create image
                autoEl:{
                    tag:'img',
                    src:'Resources/images/profile.png'
                }
            },
            {
                fieldLabel:'e-Mail',
                type:'email'
            }
        ]
    });

    login.render('loginForm');
});
```

```

        name:'user_email',
        anchor: '100%',
        allowBlank:false
    },
    fieldLabel:'Passwort',
    name:'password',
    anchor: '100%',
    inputType:'password',
    id: 'pass',
    allowBlank:false
}
],
buttons:[{
    text:'Login',
    //formBind: true,
    handler:function(){
        var fp = this.ownerCt.ownerCt,
        form = fp.getForm();
        form.submit();
    }
},
{
    text: 'Reset',
    handler: function(){
        login.getForm().reset();
    }
}]
});
};

var createwindow = new Ext.Window({
    frame:true,
    title:'Smart Booking Login...',
    width:320,
    resizable:false,
    height:170,
    draggable:false,
    closable: false,
    items: login
});
createwindow.show();
});

```

## 13.2 CSS-Styles

```

body {
    background-image: url(../../../../Resources/images/logo.jpg);
    font-family:\\"Arial\\", sans-serif;
    font-weight:normal;
    font-size:1.0em;
}

```

## 13.3 Template

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" />
<title><?php print $this->title.' '.$this->version; ?></title>
<link rel="stylesheet" type="text/css" href="Resources/ext-3.0.3/resources/css/ext-all-notheme.css" />
<link rel="stylesheet" type="text/css" href="Resources/ext-3.0.3/resources/css/xtheme-tp.css" />
<link rel="stylesheet" type="text/css" href="Resources/ext-3.0.3/src/shared/examples.css" />
<link rel="stylesheet" type="text/css" href="Templates/LoginForm/css/loginForm.css" />
</head>
<body>
<script type="text/javascript" src="Resources/ext-3.0.3/adapter/ext/ext-base.js"></script>
<script type="text/javascript" src="Resources/ext-3.0.3/ext-all.js"></script>

```

```
<script type="text/javascript" src="Resources/ext-3.0.3/src/shared/examples.js"></script>
<script type="text/javascript" src="Templates/LoginForm/javascript/LoginForm.js"></script>
<p><?php print $this->message; ?></p>
</body>
</html>
```