



Факультет программной инженерии и компьютерной техники
Информационные системы и базы данных

Курсовая работа: информационная система для частной военной компании

Преподаватель: Харитоновна Анастасия Евгеньевна

Выполнил: Кульбако Артемий Юрьевич Р33112

Предметная область

База данных для частной военной компании

Частная военная компания имеет штаб **СОТРУДНИКОВ** (у работников необходимо узнать **ИМЯ**, **ФАМИЛИЮ**, **ДАТУ_РОЖДЕНИЯ**, **ОБРАЗОВАНИЕ** и текущий **СЕМЕЙНЫЙ_СТАТУС**, а также хранить его **ДАТУ_ЗАЧИСЛЕНИЯ** на службу), которые могут занимать различные военные и невоенные **ДОЛЖНОСТИ** (**НАЗВАНИЕ_ДОЛЖНОСТИ**, **ЗАРПЛАТУ**, **ВОИНСКОЕ_ЗВАНИЕ** если есть, номер комплекта **ЭКИПИРОВКИ**, и тип **ВООРУЖЕННЫХ_СИЛ** (также работники могут занимать гражданские должности)).

Типы **ВС**: СВ, ВМФ, ВКС.

Каждый сотрудник имеет **МЕДКАРТУ** (с информацией о **РОСТЕ** в см, **ВЕСЕ** в кг, **ГРУППЕ_КРОВИ** (по системе АВО), **ПЕРЕНЕСЁННЫХ_ТРАВМАХ/ЗАБОЛЕВАНИЯХ**, **БИОЛОГИЧЕСКОМ_ПОЛЕ**) и закреплённую за ним **БАЗУ**, являющуюся его основным местом пребывания (с информацией о **МЕСТОПОЛОЖЕНИИ** базы и её **СТАТУСЕ**).

Сотрудники могут отправляться на **МИССИИ** (нужно хранить **НАЗВАНИЕ**, **ДАТУ_И_ВРЕМЯ_СТАРТА** и **ЗАВЕРШЕНИЯ**, **ЮРИДИЧЕСКИЙ_СТАТУС**, **МЕСТО_ОТПРАВЛЕНИЯ** и **ПРИВЫТИЯ**, **ВРАГОВ**), а также историю миссий сотрудников.

Миссия является большой частью военной **КАМПАНИИ** (должно содержать **НАЗВАНИЕ**, **ЗАКАЗЧИКА**, **ПРИВЫЛЬ**, **ЗАТРАТЫ** и **СТАТУС_ВЫПОЛНЕНИЯ**) на **ТРАНСПОРТЕ** (**НАЗВАНИЕ**, **ТИП**, необходимо также знать, когда **СОСТОЯНИЕ** находится), принадлежащем чвк (транспорт, естественно не утилизируется после миссии и может быть использован повторно).

Для безопасности, стоит хранить историю всех **ТЕХОСМОТРОВ** (с номерами **ТРАНСПОРТА**, **ОБСЛУЖИВАЮЩЕГО**, **ДАТЫ_ОСМОТРА**).

Людям с военной должностью должен выдаваться комплект **ЭКИПИРОВКИ** (где может быть (но необязательно) **КАМУФЛЯЖ**, **СРЕДСТВА_КОММУНИКАЦИИ**, **РАЗВЕДКИ**, **МЕДИКАМЕНТЫ** и **ПРОЧЕЕ**).

Экипировка в обязательном порядке должна включать один из **ИРП** (с описанием о **БЕЛКАХ**, **ЖИРАХ**, **УГЛЕВОДАХ** и **КАЛОРИЙНОСТИ**, **блюдах ЗАВТРАКА**, **ОБЕДА**, **УЖИНА**, **ПИЩЕВЫХ_ДОБАВКАХ**).

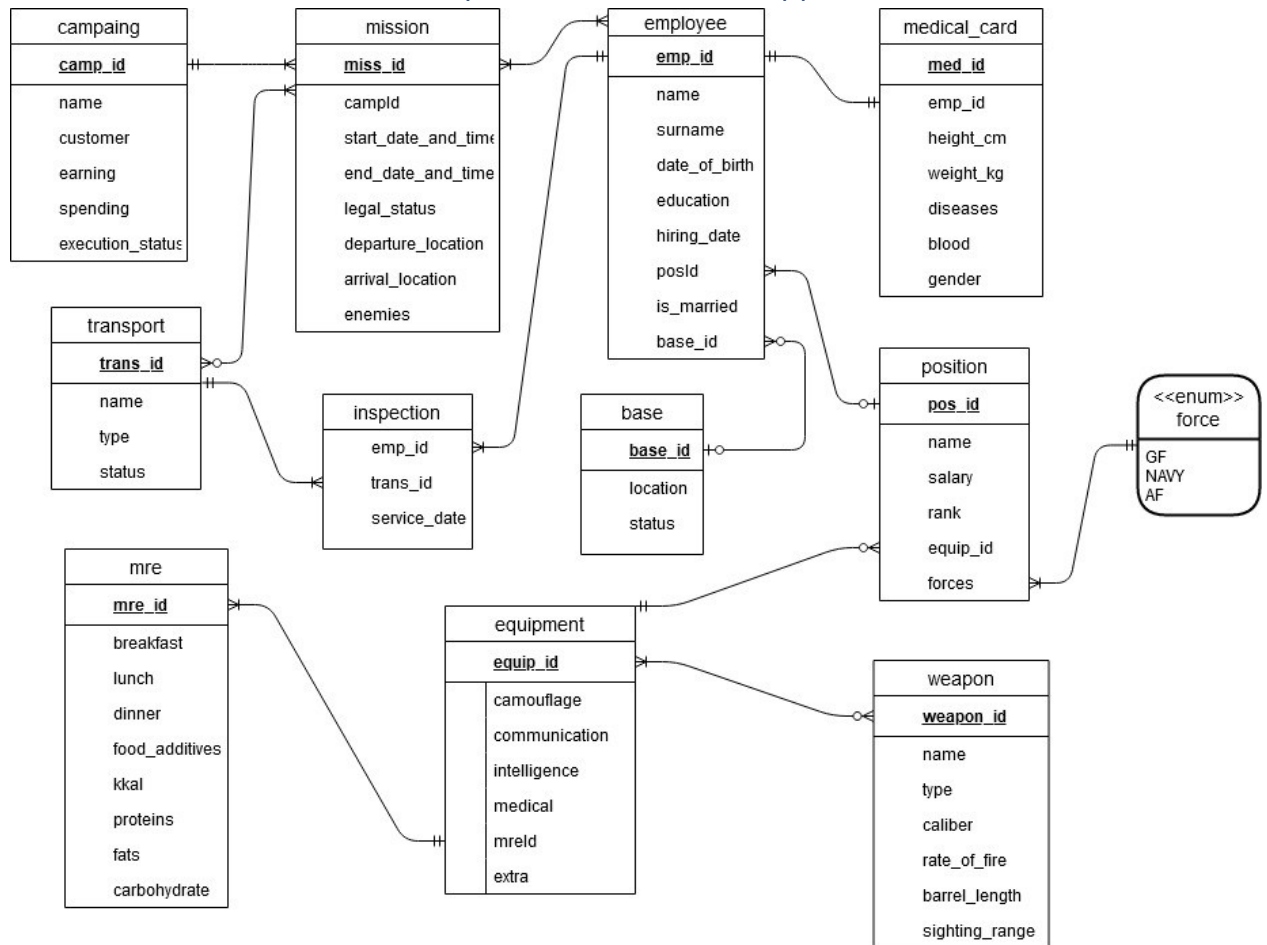
Экипировка может содержать одно или несколько **ОРУЖИЙ** (с подробными техническими характеристиками, будь то **НАЗВАНИЕ**, **ТИП**, **КАЛИБР**, **СКОРОСТРЕЛЬНОСТЬ**, **ДЛИНА_СТВОЛА**, **ПРИЦЕЛЬНАЯ_ДАЛЬНОСТЬ**).

Процессы

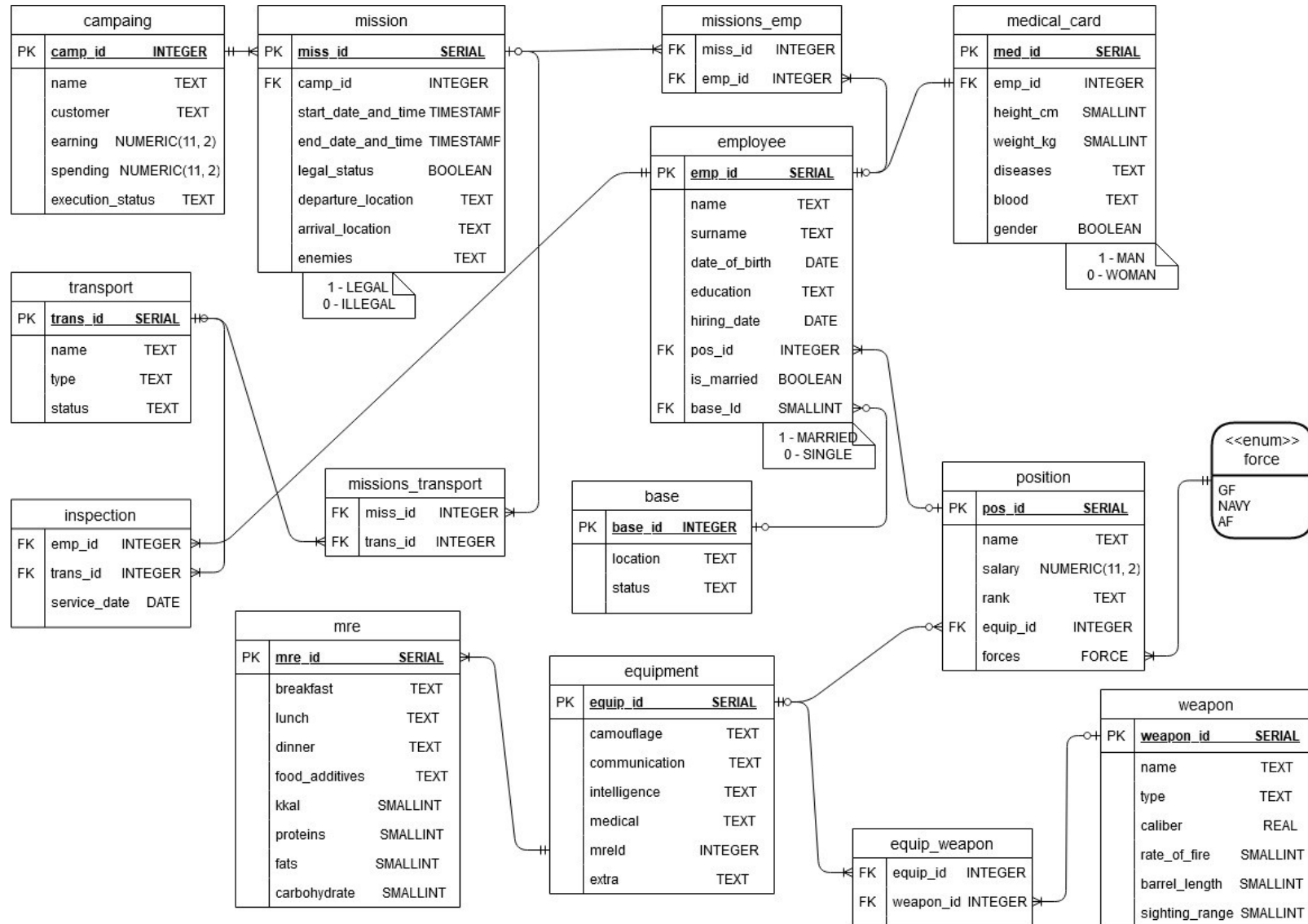
1. Тех, кто не имеет воинских званий, нельзя отправлять на боевые миссии.
2. Информационная система должна учитывать какие сотрудники отправились на миссии (один и тот же сотрудник не может находиться на двух миссиях одновременно).

3. Работников неподходящих по физическим данным запрещено устраивать как военных сотрудников.
4. Необходимо хранить историю инспекций транспорта, а транспорт со статусами «сломан» или «в ремонте» нельзя использовать в операциях.
5. Если за базой не закреплён ни один сотрудник, стоит закрыть её.
6. Стараться отправлять на боевые операции при прочих равных в первую очередь неженатых военных, давно не участвовавших в миссиях, имеющих большой опыт работы.

Инфологическая модель



Даталогическая модель



Индексы

Естественно, большая часть обращений будут происходить к таблицам, связанным с бизнес-процессами информационной системы, поэтому при оптимизации нужно делать ставку на них.

1-ую функцию оптимизировать смысла нет, т.к. при обращении к 2 из 3 трёх таблиц поиск и так происходит через индексы, ибо условие отбора работает с первичными ключами таблиц.

Во 2-ой функции, для избежание пересечения временных промежутков добавляемой миссии и существующих, нам необходимо получить этот самый интервал для каждой записи, и индекс здесь действительно будет к месту.

```
CREATE INDEX mission_period ON mission USING btree(start_date_and_time, end_date_and_time);
```

Planning Time: 0.275 ms	->	Planning Time: 0.197 ms
Execution Time: 0.179 ms		Execution Time: 0.127 ms

Выигрыш в треть.

В 3-ем необходимо сопоставлять работников и их некоторые параметры из их медицинской карты. Т.к. выборка на «нижнем уровне» дерева запроса осуществляет по ключу таблицы, индекс не нужен.

4-ый: аналогично 3-ему.

В 5-ой функции стоит сделать материализованное представление, которое будет содержать базы и количество их сотрудников, а логику обновления этой таблицы запускать по запросу на изменение данных Employee.

```
CREATE MATERIALIZED VIEW base_count_emp AS
  (SELECT base_id FROM base JOIN employee USING (base_id) GROUP BY base_id
  HAVING COUNT(emp_id) = 0);

CREATE FUNCTION update_base_count_emp() RETURNS trigger AS $$
  BEGIN
    REFRESH MATERIALIZED VIEW base_count_emp;
  END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER update_base_count_emp BEFORE INSERT OR UPDATE OR DELETE ON
employee
  FOR EACH ROW EXECUTE FUNCTION update_base_count_emp();
```

В последней функции присутствует условие отбора записей, не по ключу, и в таблице (правда записей в таблице не так уж и много). Создадим индекс:

```
CREATE INDEX pos_rank ON position USING btree(rank);
```

Planning Time: 0.069 ms
Execution Time: 0.053 ms

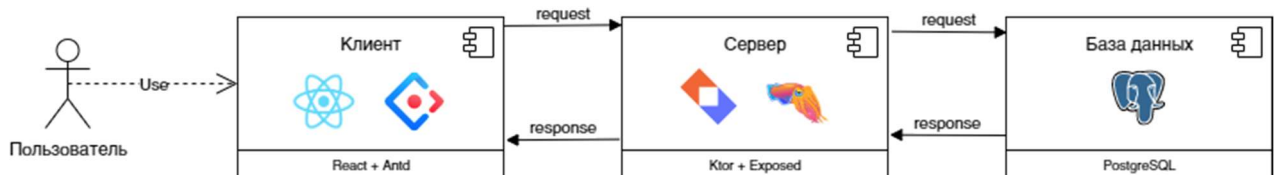
->

Planning Time: 0.070 ms
Execution Time: 0.039 ms

Время незначительно сократилось.

Информационная система

Компонентная модель



api

Аpi до безобразия банален. Для каждой сущности заведён свой url, а http-метод определяет операцию. Все данные передаётся в json.

GET (получение)	<p>Отправляется ключ "selectedIds" для получения записей, значением является массив (если пустой, то вернуть все записи, иначе с запрошенными id).</p> <p>Примеры:</p> <pre>GET http://localhost:9090/base?ids={ "selectedIds": [] } GET http://localhost:9090/base?ids={ "selectedIds": [3, 5] }</pre>
POST (добавление)	<p>Нужно передать всю сущность, вместо уникального ключа необходимо передавать null.</p> <p>Пример:</p> <pre>POST http://localhost:9090/base Content-Type: application/json { "baseId" : null, "location" : "LUNA", "status" : "TEST-BASE" }</pre>
PUT (модификация)	<p>Обязательно передаётся id сущности, а также вся сущности или только те поля, которые будут отличаться.</p> <p>Пример:</p> <pre>PUT http://localhost:9090/base Content-Type: application/json { "baseId": 419, "location": "MARS" }</pre>
DELETE (удаление)	<p>Передаётся массив с id удаляемых записей.</p> <p>Пример:</p> <pre>DELETE http://localhost:9090/base Content-Type: application/json</pre>

```
{
  "droppedIds": [414, 415]
}
```

Также есть уникальный путь **god**, для целей разработки:

```
### получить страницу godmode
GET http://localhost:9090/god

### создать таблицы
PUT http://localhost:9090/god

### заполнить таблицы случайными данными
POST http://localhost:9090/god

### уничтожить таблицы и все записи
DELETE http://localhost:9090/god
```

Реализация

Стек технологий выбирал по двум параметрам: возможность развернуть на сервере BT, при этом это должно было быть что-то, с чем я не работал. Бэкенд первоначально планировался на [Micronaut](#) (отказался из-за не самой подробной документации, возможно в будущем попробую ещё раз), потом [Node.js](#) (на Гелиосе запустить его не удалось). Тогда было решено вернуться к чему-то на JVM. Я знаю Kotlin, поэтому сначала решил попробовать Spring на Kotlin, но после перешёл на связку [Ktor](#) и [Exposed](#) – с нуля созданных на Kotlin для него фреймворка и ORM. Ktor чудесен – написать REST-API приложение можно быстро, и в отличие от Java + Spring количество бойлерплейт кода невелико. С Exposed оказалось сложнее (для взаимодействия с бд можно использовать DSL или DAO подход, и каждый из них имеет свои плюсы и минусы. В целом, Exposed мне понравился, но работе с ним всё же принесла немного боли.

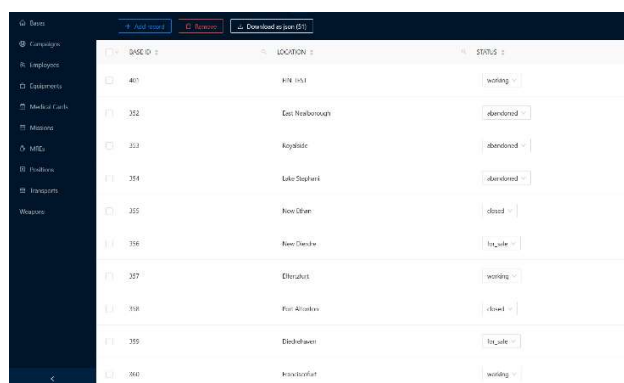
Фронт решил попробовать делать на [React](#) (до этого имел опыт с Vue, поэтому думал будет несложно). Несмотря на общую компонентную концепцию обоих, а также общие методы создания этих самых компонентов, JSX (язык React), показался мне чуть сложнее, ввиду не такого явного как у Vue разделения на разметку, логику и стиль. Также мне не удалось избежать вечной проблемы JavaScript – «ада коллбеков», и даже рефакторинг кода, с заменой части коллбеков на промисы не сильно спас ситуацию. Понравилось работать с библиотекой компонентов [Antd](#), которая сильно сократила время работы над фронтендом. Подводя итог, я всё же в будущем предпочту Vue, хотя уже после разработки вижу, что некоторых проблем можно было избежать, если бы на стадии проектирования я уже знал React.

Пробела была и с REST-API. Во время разработки бэкенда, я не вспомнил, что в http-GET запросе в теле метода не рекомендуется передавать данные, но явно это не запрещается. В время тестирования, я удачно посылал GET с json-контентом в теле, а позже обнаружил, что js-овский метод `fetch`, предназначенный для ajax-запросов, не позволяет этого делать, поэтому пришлось отправлять json-данные как параметр, прикреплённый к url-у.

С слоем базы данных проблем не возникало – [PostgreSQL](#) – проверенная временем база данных, информацию о которой достаточно легко найти и понять.

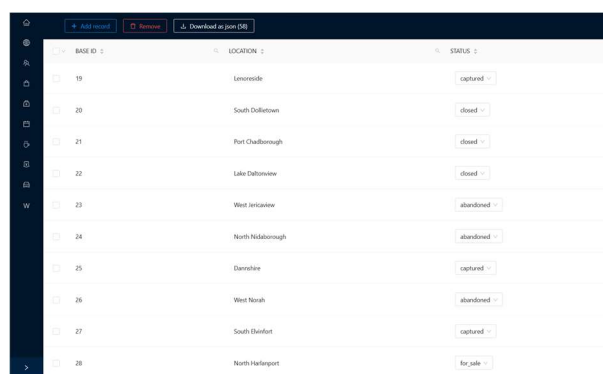
Руководство пользователя

При запуске программы вас встретит экран с таблицей первой по алфавиту сущности. Слева располагается меню-«гамбургер», где вы можете выбрать для просмотра записи необходимой сущности. Кнопка со стрелкой вниз меню позволяет свернуть его, чтобы освободить больше места для таблицы.



BASE ID	LOCATION	STATUS
400	Ark Ark	working
382	East Newborough	abandoned
383	Kaydick	abandoned
384	Lake Stephen	abandoned
385	Rocky Glen	closed
386	Rocky Glen	for sale
387	Elterabuck	working
388	Fort Alton	closed
389	Elterabuck	for sale
390	Ironstone	working

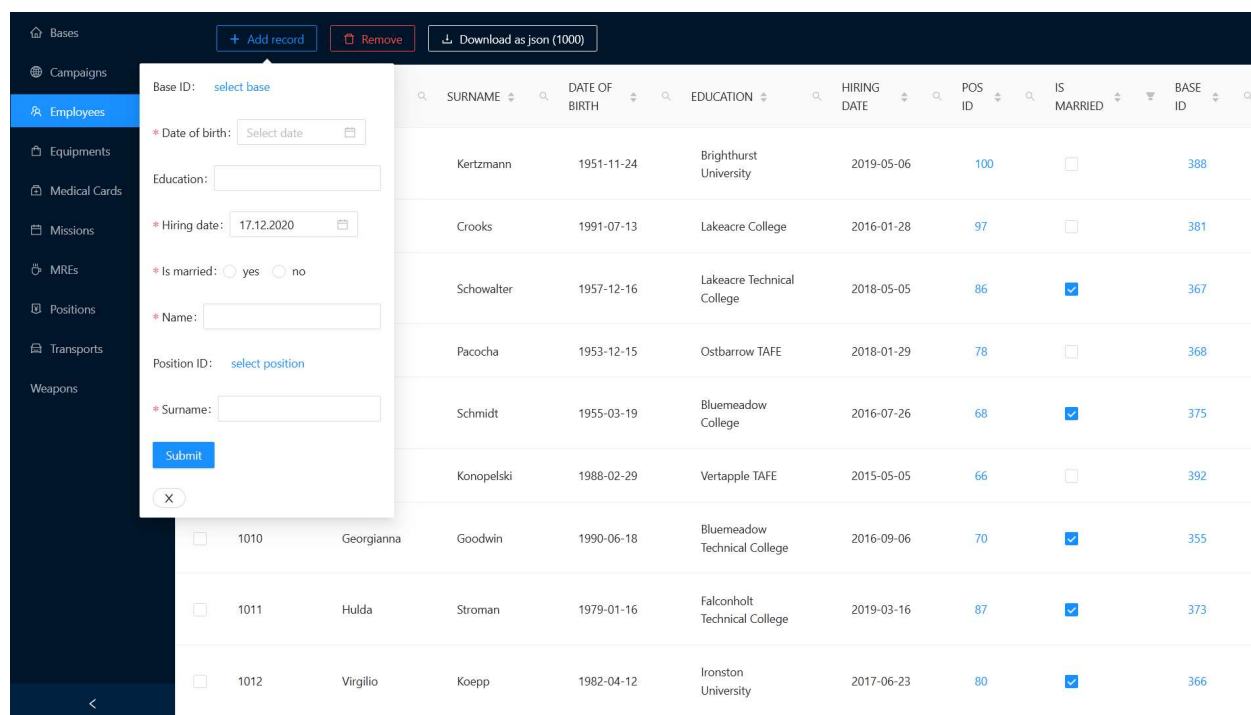
меню отображается



BASE ID	LOCATION	STATUS
19	Limestone	captured
20	South Deltown	closed
21	Fort Chalkborough	closed
22	Lake Deltown	closed
23	West Limestone	abandoned
24	North Deltown	abandoned
25	Damshire	captured
26	West North	abandoned
27	South Deltown	captured
28	North Harpport	for sale

меню скрыто

Для добавления новой записи в выбранную таблицу нажмите на кнопку «Add record»: появится окошко, которое необходимо заполнить, и нажать “Submit”. Если введённые данные корректы, появится уведомление о успехе операции, и новая запись отобразится в конце таблицы, иначе о возникших ошибках.

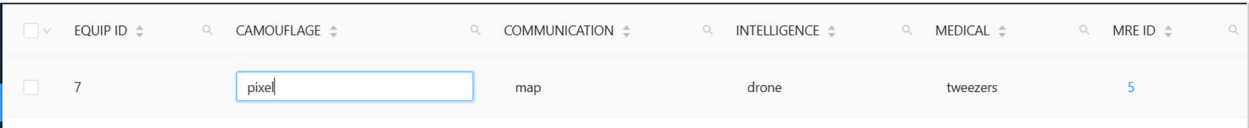


SURNAME	DATE OF BIRTH	EDUCATION	HIRING DATE	POS ID	IS MARRIED	BASE ID
Kertzmenn	1951-11-24	Brighthurst University	2019-05-06	100	<input type="checkbox"/>	388
Crooks	1991-07-13	Lakeacre College	2016-01-28	97	<input type="checkbox"/>	381
Schowalter	1957-12-16	Lakeacre Technical College	2018-05-05	86	<input checked="" type="checkbox"/>	367
Pacocha	1953-12-15	Ostbarrow TAFE	2018-01-29	78	<input type="checkbox"/>	368
Schmidt	1955-03-19	Bluemeadow College	2016-07-26	68	<input checked="" type="checkbox"/>	375
Konopelski	1988-02-29	Vertapple TAFE	2015-05-05	66	<input type="checkbox"/>	392
Goodwin	1990-06-18	Bluemeadow Technical College	2016-09-06	70	<input checked="" type="checkbox"/>	355
Hulda	1979-01-16	Falconholt Technical College	2019-03-16	87	<input checked="" type="checkbox"/>	373
Koepp	1982-04-12	Ironston University	2017-06-23	80	<input checked="" type="checkbox"/>	366

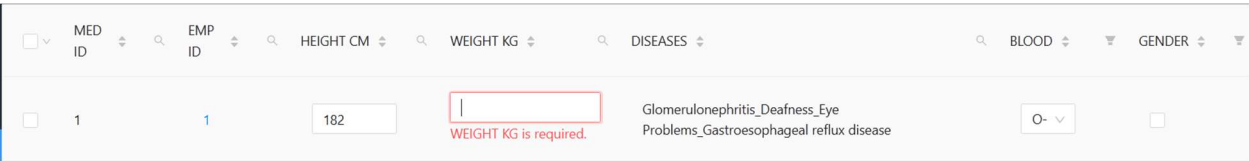
Кнопка «Remove» удаляет одну или несколько записей, предварительно выбранных.

«Download as json» загрузит json-дамп всех выбранных записей таблицы или всех по умолчанию.

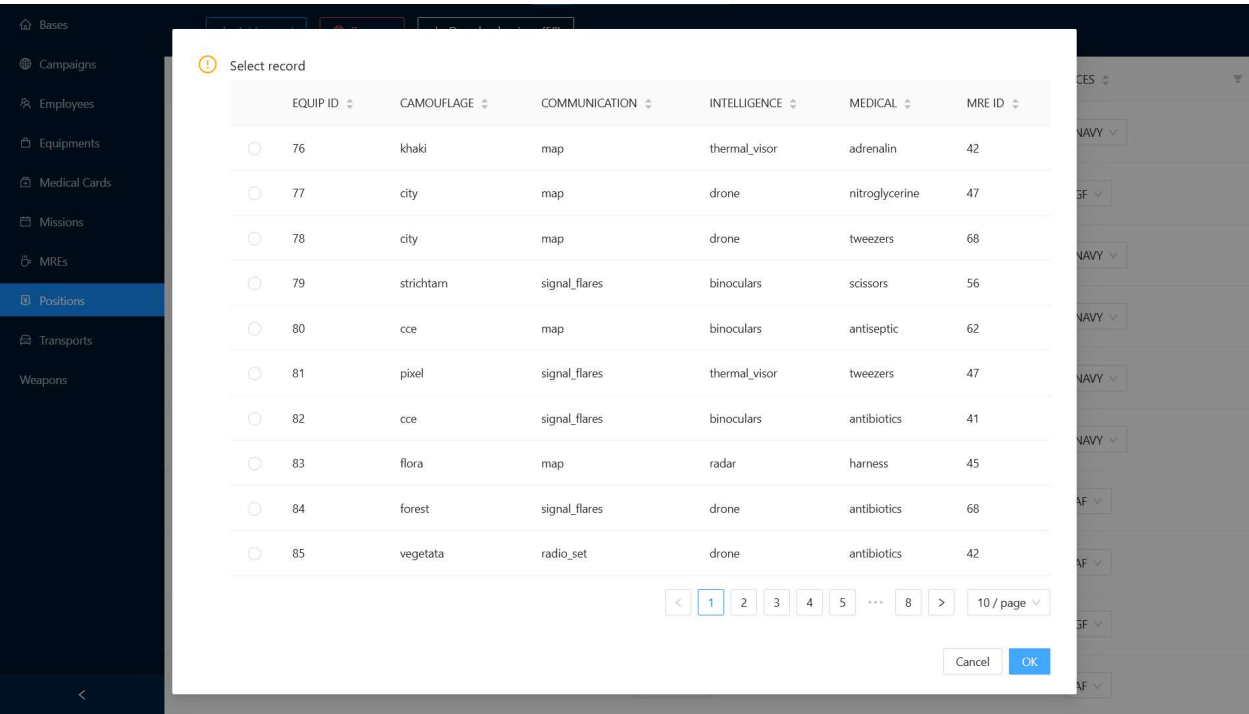
Все поля (включая ссылки на записи других сущностей) являются изменяемыми за исключением уникального id редактируемой сущности.



Если поле необходимо заполнить – система выдаст предупреждение (тоже самое произойдёт, при попытке ввести в ячейку некорректные данные).



При редактировании или добавление ссылок на связанные сущности, откроется упрощённая таблица этой сущности, чтобы выбирать было удобнее.

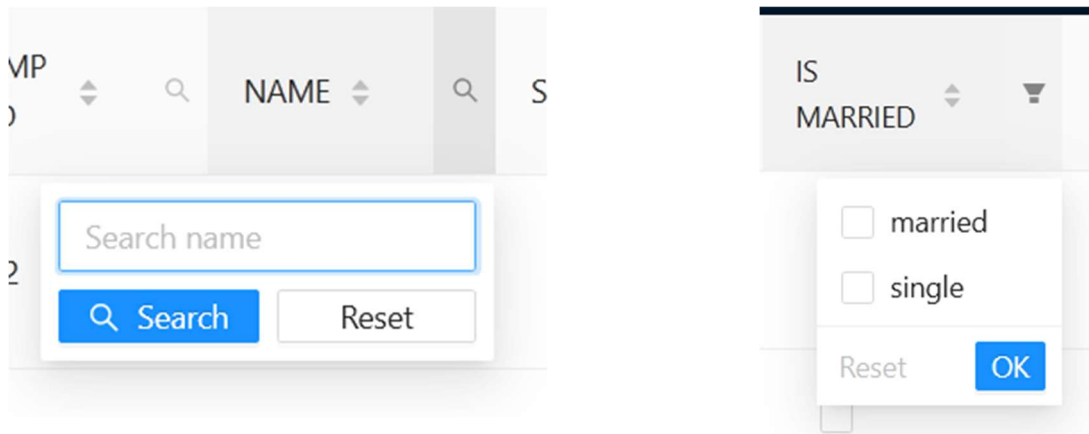


Если же вы хотите просто посмотреть подробную информацию о ней, достаточно навести мышку.

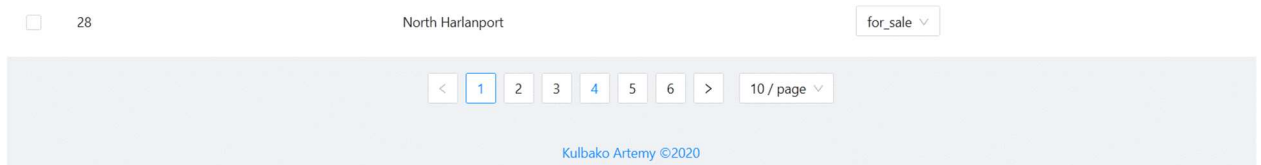
<input type="checkbox"/>	13	strichtarn	radio_set	thermal_visior	scissors	19
<input type="checkbox"/>	14	flora	signal_flares	binoculars	antiseptic	
<input type="checkbox"/>	15	desert	map	radar	ammonia	
<input type="checkbox"/>	16	strichtarn	signal_flares	radar	antibiotics	9
<input type="checkbox"/>	17	cce	radio_set	radar	ammonia	16

[[{"mreId":9,"breakfast":"French Fries with Sausages","lunch":"Peking Duck","dinner":"Som Tam","foodAdditives":"Feijoa_Bonza_Lychees","kcal":1982,"proteins":12,"fats":38,"carbohydrate":280}]]

Для удобства манипуляции с данными вы можете осуществлять поиск, сортировку и фильтрацию по одной или нескольким колонкам.



Внизу страницы есть кнопка для связи с разработчиком:



Вывод

В рамках курсовой работы я полностью создал информационную систему: от этапа формирования предметной области (где преподаватель выступал в качестве некоего заказчика), до этапа внедрения на оборудовании этого самого заказчика (Helios). За все два с половиной курса – это самое полезное и наиболее близкое к настоящей работе задание, где мне «повезло» столкнуться даже с типичными проблемами реальных проектов, в виде неправильного спроектированного API, к примеру, поверхностного знания технологий, а также неудачному проектированию, ввиду нехватки времени (по-моему мнению, это моя самая большая ошибка).

Полный код (включая конфигурационные файлы и ресурсы) доступен по ссылке:



Код бизнес-логики в прикреплённых файлах:



back_code.pdf



front_code.pdf



db_code.pdf

*файлы кликабельны