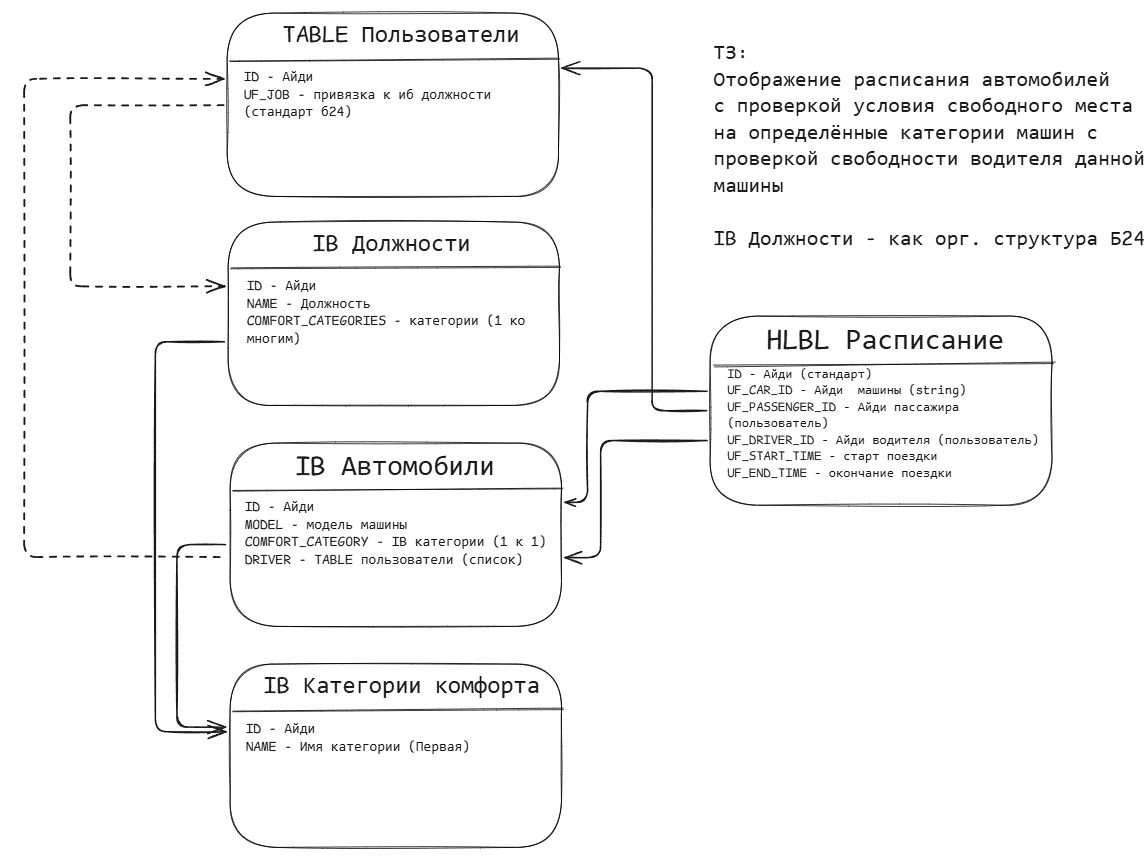
1. Схема



1. Код

Написать компонент, который выводит для текущего сотрудника список свободных автомобилей на запрошенное время (запрошенное время начала и время окончания поездки передается от клиента на сервер в get-параметрах) с указанием модели, категории комфорта, водителя

Фильтр по времени + проверка по должности на категорию и свободность водителя

Файл class.php

<?php

if (!defined('B\_PROLOG\_INCLUDED') || B\_PROLOG\_INCLUDED !== true) die();

use Bitrix\Main\Loader;

use Bitrix\Main\Context;

use Bitrix\Main\Type\DateTime;

use Bitrix\Main\UserTable;

use Bitrix\Main\Engine\CurrentUser;

use Bitrix\Iblock\IblockTable;

use Bitrix\Iblock\ElementTable;

use Bitrix\Iblock\PropertyTable;

use Bitrix\Highloadblock\HighloadBlockTable;

use Bitrix\Main\Web\Json;

class CompanyFreeCarsComponent extends CBitrixComponent

{

/\*\* @var CurrentUser \*/

private CurrentUser $currentUser;

/\*\* @var \Bitrix\Main\HttpRequest \*/

protected $request;

// Константы инфоблоков и HL-блоков

private const IBLOCK\_CODE\_JOBS = 'job\_positions';

private const IBLOCK\_CODE\_CARS = 'cars';

private const IBLOCK\_CODE\_COMFORT\_CATEGORIES = 'comfort\_categories';

private const HL\_SCHEDULE\_TABLE\_NAME = 'b\_schedule';

public function \_\_construct($component = null)

{

parent::\_\_construct($component);

if (!Loader::includeModule('iblock') || !Loader::includeModule('highloadblock')) {

throw new \Bitrix\Main\LoaderException('Не удалось подключить модули iblock или highloadblock');

}

$this->currentUser = CurrentUser::get();

$this->request = Context::getCurrent()->getRequest();

}

public function executeComponent()

{

$startTimeStr = $this->request->get('start\_time');

$endTimeStr = $this->request->get('end\_time');

if (!$startTimeStr || !$endTimeStr) {

ShowError('Укажите GET-параметры: start\_time и end\_time');

return;

}

try {

$startTime = new DateTime($startTimeStr);

$endTime = new DateTime($endTimeStr);

if ($startTime >= $endTime) {

ShowError('Время начала должно быть раньше окончания');

return;

}

} catch (\Exception $e) {

ShowError('Неверный формат времени');

return;

}

$userId = (int)$this->currentUser->getId();

if (!$userId) {

ShowError('Пользователь не авторизован');

return;

}

$this->arResult = $this->getFreeCars($userId, $startTime, $endTime);

echo Json::encode($this->arResult, JSON\_UNESCAPED\_UNICODE);

}

/\*\*

\* Основная логика поиска свободных машин

\*/

private function getFreeCars(int $userId, DateTime $startTime, DateTime $endTime): array

{

$jobId = UserTable::getList([

'filter' => ['=ID' => $userId],

'select' => ['UF\_JOB']

])->fetch()['UF\_JOB'];

if (!$jobId) {

return ['error' => 'У пользователя не указана должность'];

}

$comfortCategoryIds = $this->getComfortCategoryIdsByJob($jobId);

if (empty($comfortCategoryIds)) {

return ['error' => 'Для должности не заданы категории комфорта'];

}

$cars = $this->getCarsByComfortCategories($comfortCategoryIds);

if (empty($cars)) {

return ['error' => 'Нет автомобилей с подходящей категорией'];

}

$freeCars = $this->filterFreeCars($cars, $startTime, $endTime);

return array\_values($freeCars);

}

/\*\*

\* Получить ID категорий комфорта по должности (из множественного свойства PROPERTY\_COMFORT\_CATEGORIES)

\*/

private function getComfortCategoriesByJob(int $jobId): array

{

$iblockId = $this->getIblockId(self::IB\_JOBS);

if (!$iblockId) {

return [];

}

$rows = \Bitrix\Iblock\ElementPropertyTable::getList([

'filter' => [

'IBLOCK\_ELEMENT\_ID' => $jobId,

'=IBLOCK\_PROPERTY.CODE' => 'COMFORT\_CATEGORIES',

'=IBLOCK\_ELEMENT.IBLOCK\_ID' => $iblockId

],

'select' => ['VALUE'],

])->fetchAll();

return array\_column($rows, 'VALUE');

}

/\*\*

\* Получить автомобили по категориям комфорта с моделью, категорией и водителем

\*/

private function getCars(array $categoryIds): array

{

$iblockId = $this->getIblockId(self::IB\_CARS);

if (!$iblockId) {

return [];

}

$cars = [];

$props = \Bitrix\Iblock\ElementPropertyTable::getList([

'filter' => [

'=IBLOCK\_PROPERTY.CODE' => ['MODEL', 'COMFORT\_CATEGORY', 'DRIVER'],

'=IBLOCK\_ELEMENT.IBLOCK\_ID' => $iblockId,

],

'select' => ['IBLOCK\_ELEMENT\_ID', 'VALUE', 'IBLOCK\_PROPERTY.CODE']

]);

while ($prop = $props->fetch()) {

$id = (int)$prop['IBLOCK\_ELEMENT\_ID'];

$cars[$id]['ID'] = $id;

$cars[$id][$prop['IBLOCK\_PROPERTY\_CODE']] = $prop['VALUE'];

}

// фильтруем по категории

return array\_filter($cars, fn($car) =>

isset($car['COMFORT\_CATEGORY']) &&

in\_array($car['COMFORT\_CATEGORY'], $categoryIds)

);

}

/\*\*

\* Фильтрация свободных машин по расписанию (проверка машины и водителя)

\*/

private function filterFreeCars(array $cars, DateTime $start, DateTime $end): array

{

$entityClass = $this->getHlEntity(self::HL\_SCHEDULE);

if (!$entityClass) {

return $cars;

}

$carIds = array\_column($cars, 'ID');

$driverIds = array\_column($cars, 'DRIVER');

$booked = $entityClass::getList([

'filter' => [

[

'LOGIC' => 'OR',

['UF\_CAR\_ID' => $carIds],

['UF\_DRIVER\_ID' => $driverIds],

],

[

'LOGIC' => 'AND',

['<=UF\_START\_TIME' => $end],

['>=UF\_END\_TIME' => $start],

],

],

'select' => ['UF\_CAR\_ID', 'UF\_DRIVER\_ID']

])->fetchAll();

$busyCars = array\_column($booked, 'UF\_CAR\_ID');

$busyDrivers = array\_column($booked, 'UF\_DRIVER\_ID');

$free = [];

foreach ($cars as $id => $car) {

if (!in\_array($id, $busyCars) && !in\_array($car['DRIVER'], $busyDrivers)) {

$driver = $car['DRIVER']

? UserTable::getById($car['DRIVER'])->fetch()

: null;

$free[$id] = [

'MODEL' => $car['MODEL'] ?? '—',

'CATEGORY\_ID' => $car['COMFORT\_CATEGORY'],

'DRIVER' => $driver ? trim($driver['NAME'] . ' ' . $driver['LAST\_NAME']) : '—',

];

}

}

return $free;

}

/\*\*

\* Получить ID инфоблока по коду

\*/

private function getIblockIdByCode(string $code): ?int

{

$iblock = IblockTable::getList([

'filter' => ['=CODE' => $code],

'select' => ['ID'],

])->fetch();

return $iblock ? (int)$iblock['ID'] : null;

}

/\*\*

\* Получить HLBL

\*/

private function getHlEntity(string $code): ?string

{

$hl = HighloadBlockTable::getList([

'filter' => ['=NAME' => $code],

'select' => ['\*']

])->fetch();

return $hl ? HighloadBlockTable::compileEntity($hl)->getDataClass() : null;

}

}