Sistem de gestionare a cererilor de despăgubire pentru o societate de asigurări

Rareș-Neagu Florian

June 14, 2017

Tabel de conținut

1 Introducere **4**
- 1.1 Condițiile preexistente . . . 4
- 1.2 Servicii prestate clienților . . . 5
- 1.3 Detalii despre relația dintre firma angajatoare și Altex / Media Galaxy . . . 5
- 1.4 Detalii despre dezvoltarea aplicației . . . 5
- 1.5 Despre aplicație . . . 6

2 Tehnologiile folosite **8**
- 2.1 PHP 5.6 . . . 8
- 2.2 Laravel 5.3 . . . 9
  - 2.2.1 Directive Blade . . . 9
  - 2.2.2 Laravel Forms / HTML . . . 10
  - 2.2.3 Artisan . . . 11
  - 2.2.4 Eloquent ORM . . . 11
- 2.3 Baza de date - MySQL . . . 11
- 2.4 GitHub . . . 11
- 2.5 CloudFlare . . . 12
- 2.6 jQuery & jQuery UI . . . 12
- 2.7 Google Analytics . . . 12
- 2.8 Knockout.js . . . 13
- 2.9 LaTeX . . . 13
- 2.10 Mentiuni speciale . . . 13
  - 2.10.1 Shared hosting . . . 13
  - 2.10.2 PhpStorm IDE . . . 13
  - 2.10.3 Organizarea cunostiintelor pe un kanban board - Trello . . . 13
  - 2.10.4 Heroku . . . 13
  - 2.10.5 Amazon Web Services . . . 13
  - 2.10.6 npm . . . 14
  - 2.10.7 Composer . . . 14

3 Structura Aplicației **15**
3.1 Request life-cycle . . . 15
3.2 O scurta istorie a aplicatiei . . . 15
3.2.1 Rapoarte - vizualizarea datelor => refactor . . . 15
3.2.2 adaugarea campurilor de undecided / resolved / repaired 15
3.2.3 legatura stransa dintre claim-sale . . . 15
3.2.4 decuplarea . . . 15
3.2.5 FileDataExcelAdaptor . . . 16
3.2.6 matchuirea pentru rapoartele jmeky . . . 16
3.2.7 todo: includerea campului de external_id . . . 16
3.2.8 todo: decuplarea in one-to-one pentru a arata mai dragut / scoate rapoarte mai complicate cand o sa se ceara . . . 16
3.3 Baza de date . . . 16
3.3.1 Assurance . . . 16
3.3.2 Chat Claims . . . 16
3.3.3 Claims . . . 16
3.3.4 Decisions + parent decision (many-to-many) . . . 16
3.3.5 Messages . . . 16
3.3.6 Migrations . . . 16
3.3.7 Password Resets . . . 16
3.3.8 Photos . . . 16
3.3.9 Products . . . 16
3.3.10 Sales . . . 16
3.3.11 Sessions . . . 16
3.3.12 User . . . 16
3.4 Apache & PHP . . . 16
3.5 MVC - Model View Controller . . . 17
3.5.1 Controllers . . . 17
3.5.2 Models - Eloquent ORM . . . 18
3.5.3 functii ajutatoare - Helpers . . . 20
3.5.4 Request-uri . . . 20
3.5.5 .env ironment-ul . . . 20
3.5.6 Logging . . . 21
3.5.7 Event - Notification - Mail system . . . 21
3.5.8 Excel . . . 21

4 Manual de utilizare **22**
4.1 Administrative interface . . . 22
4.1.1 Claims view . . . 22
4.1.2 Import . . . 22
4.1.3 Sales . . . 23
4.1.4 Decisions . . . 23
4.1.5 Reports . . . 23
4.2 Registering a claim . . . 23
4.3 Responding to messages . . . 23
4.4 Settings . . . 23

4.5 Uploading photos 23

5 Concluzii **24**
5.1 Business Continuity Plan 24
5.2 Teste automate 24
5.3 Modularizarea tabelelor 25
5.3.1 Modificări necesare asupra structurii bazei de date 25
5.4 Sistem modular de rapoarte 25

Capitolul 1

Introducere

1.1 Condițiile preexistente

În momentul în care am interacționat prima oară cu Fandu, doreau o redirectare simplă a paginii de internet `http://www.fandu.uk/claims`[10] către `http://wsgp.co.uk/claims/`[17].

Proiectul era administrat pe server-ul companiei angajate să scrie codul. Pagina principală încânta utilizatorul cu o avertizare în engleză ce spunea că sistemul ales de gestionare a bazei de date va fi în curând scos din limbajul de programare ales de ei. Nu aveau validări a datelor, nu scoteau rapoartele necesare. Sistemul de gestiune a cererilor de despăgubire era centrat în jurul unor fișiere Excel, ținute de angajați și distribuite între ei.

Astfel, Fandu și-a dorit să poată să scoată rapoartele dintr-o aplicație. Fără să treacă cineva prin fiecare daună înregistrată. Fără să copieze informațiile dintr-un mail trimis automat. Au dorit să automatizeze o mare parte din munca asiduă a angajaților Fandu. De a putea scoate oricând rapoartele necesare pe orice perioadă de timp, fie ea o lună, un an, sau de la prima cerere de despăgubire.

Soluția a venit din partea mea. Dorind să mă afirm și să am experiență în domeniu, m-am oferit ca în timpul liber să ajut Fandu. Să scot problema umană din ecuația rapoartelor, să trimit automat mail și să ofer o soluție completă, eficientă, construită de jos în sus pentru fiecare parte a dorințelor lor. Ei au fost sceptici că pot, așa că de menționat ar fi că metoda „rudimentară" de a păstra datele în fișierele Excel a fost activă pe perioada dezvoltării aplicației, pentru a se asigura exactitatea rapoartelor scoase de sistem cu cele „manuale". Și asta a ajutat enorm. Multe probleme mici au apărut în timpul dezvoltării, ce au fost reparate o dată ce au fost comparate datele.

1.2 Servicii prestate clienților

Utilizatorii primesc în regulamentul de „Mobile Protect", valabil online și la `https://mediagalaxy.ro/regulament-mobile-protect`[18], în momentul achiziției unui telefon sau unei tablete, indicațiile de a se uita la `www.fandu.uk/claims` pentru cererile de despăgubire. Pot să trimită la adresa respectivă și documentele relevante pentru a soluționa cererii de despăgubire.

Sunt rugați de a completa cât mai complet și a da cât de multe detalii posibile pentru soluționarea mai rapidă a cererii.

Regulamentul mai avertizează repercursiunile legale posibile în cazul falsificării informațiilor (și automat respingerea cererii). Important de menționat, pentru victimele furtului agravat, este necesitatea raportului poliției în cazul furtului.

1.3 Detalii despre relația dintre firma angajatoare și Altex / Media Galaxy

Societatea „Fandu Holdings Ltd." cu sediul în Zinonos Kitieos nr. 8, Kato Lakatamia P.C. 2322, Nicosia, Cyprus[10] este o societate ce se ocupă cu cererile de despăgubire pentru Altex și Media Galaxy[18].

Aceasta răspunde în numele lui „AmTrust International Underwriters Limited".[18].

1.4 Detalii despre dezvoltarea aplicației

Dezvoltarea aplicației a început cu prima întâlnire între fondatorii companiei, cei ce aveau grijă de sistemul de rapoarte în Excel și mine. Ne-am așezat la masă și astfel am început să discutăm despre cerințele aplicației.

Accentul s-a pus pe dorința de a scoate rapoarte mai ușor, fără a putea interveni eroarea umană. Să nu se mai verifice manual câte asigurări s-au vândut într-o perioadă de timp determinată. Eliminarea nevoii de a căuta în fiecare fișier trimis săptămânal de Altex / Media Galaxy.

Pe tot parcursul dezvoltării aplicației, am fost singurul ce s-a ocupat de aranjarea sistemului informatic, migrarea spre o soluție de găzduire partajată, construirea bazei de date, relațiilor, securitatea datelor și interacțiunii utilizatorului cu aplicația.

Structura bazei de date a fost gândită prima oară a fi rigidă, a forța utilizatorul și persoanele ce se ocupă de gestiunea lor să introducă datele corecte, a nu avea probleme ulterioare. Ideea a fost întâmpinată cu rezistență, datorită libertății oferite de Excel, dar într-o scurtă perioadă de timp, rigiditatea a dat roade. Rapoartele erau exacte și erorile ne-existente.

Interacțiunea cu utilizatorul de rând a fost gândită a fi cât mai rapidă. Atunci când intră pe pagina aplicației, el poate deja să completeze cererea de despăgubire, așa cum se poate observa în poza 1.1. Utilizatorul este îndrumat

Figure 1.1: Pagina principală a `https://fandu.info`

pentru a completa cât mai detaliat detaliile despre cererea de despăgubire și este avertizat atunci când apare o eroare sau a uitat să completeze un câmp necesar.

Am ales să salvez toate datele utilizatorului în baza proprie de date, la o companie de încredere ce oferă soluții de găzduire partajată `https://xServers.ro`[19]. Dar pozele și spațiul de stocare a pdf-urilor încărcate de utilizatori sunt găzduite de Amazon Web Services[1], pentru a beneficia de prețul redus și valabilitatea extinsă a fișierelor pe „cloud".

Revenind la soluția de găzduire partajată, am ales xServers.ro[19] pentru că aplicația se încadrează lejer în oferta lor. Cel mai important atu este traficul de internet necontorizat și spațiu de stocare suficient.

Soluția de backup, oferită de cei ce găzduiesc aplicația, salvează baza de date în fiecare zi.

De asemenea, mă ocup personal pentru a asigura un backup săptămânal a bazei de date pe un volum encriptat cu „VeraCrypt"[8], un utilitar gratuit ce construiește un disc virtual securizat într-un simplu fișier.

Cele două soluții complementare asigură securitatea datelor și rezolvarea instantanee a problemelor bazei de date, cheia aplicației.

Codul sursă este ținut într-un proiect „git", un sistem de gestionare și versionare a codului de programare, încărcat pe GitHub, un aliat de încredere oricărui programator.

1.5 Despre aplicație

Pentru a putea reduce costurile aplicației, am optat pentru o metodă tradițională de găzduire partajată, ce oferă PHP și o bază de date MySQL inclusă în pachet.

Chiar dacă experiența mea unde lucrez își spune cuvântul în alt limbaj de programare (C#), am descoperit plictisindu-mă Laravel[14], o platformă modernă de a scrie cod PHP, inspirată din arhitectura sistemului „Ruby on Rails".

Înainte să interacționez cu persoanele de la Fandu, aveam deja mai multe

proiecte ușoare terminate. Știam cât de important este să nu te avânți cu capul înainte, fără a ști un nou cadru în care să programezi, pentru că rezultatul ar fi deplorabil.

Nu folosisem până acum într-un mediu de producție Laravel, dar eram încrezător că nu mi-ar oferi surprize neplăcute. De asemenea, văzusem oportunitatea de a folosi cele mai ridicate standarde de programare pentru PHP, și până la urmă am ales să merg el.

Astfel, am învățat cum este să gestionezi de unul singur dezvoltarea unei aplicații de la concept la problemele de memorie când nu poți să scoți rapoarte cu peste 30000 de linii, cu baza de date de pe producție.

Alt avantaj este încercarea minimizării complexității ascunse[15], ce face codul mult mai ușor de întreținut și permite dezvoltarea mai rapidă, fără a petrece mult timp înțelegând arhitectura sau codul scris anterior.

Sintaxa ușor de înțeles și ambiguitatea redusă fac din Laravel, în opinia mea, un bun provocator chiar și pentru „ASP.NET MVC"[7], liderul pe piață în momentul actual. Ambele platforme oferă soluții de la modele, la gestionarea bazei de date, la rularea migrațiilor, la salvarea stării și chiar și integrarea automată a unui sistem de utilizatori.

Dar spre diferență de soluția Microsoft, toată platforma necesară construirii și distribuirii aplicației nu este blocată pe sistemul de operare Windows (cu .NET Core situația se schimbă, dar în acest moment nu a fost adoptată la scară largă), ci poate să funcționeze pe orice sistem de operare.

Capitolul 2

Tehnologiile folosite

2.1 PHP 5.6

Când a apărut internet-ul, toate paginile erau statice. Nu puteai să interacționezi cu conținutul paginii. Dar apoi Netscape a revoluționat paginile, construind limbajul de programare LiveScript (redenumit ulterior JavaScript). În puțini ani, Netscape a revoluționat și modul în care paginile sunt distribuite utilizatorilor prin conceptul de JavaScript pentru aplicațiile ce serveau paginile. Astfel, s-a construit primul scenariu de cod folosit de servere. [12]

În continuare, s-a definit „Common Gateway Interface" (CGI), ce oferă un protocol pentru serverele web de a executa programe pentru a genera pagini de web dinamic atunci când este nevoie de o pagină de internet. Serverul web permite administratorului să seteze ce URL-uri vor fi folosite de ce programe CGI prin setarea unui folder cu scripturi. În loc să trimită fișierul respectiv, serverul HTTP apelează script-ul și trimite orice ar fi afișat ca și rezultat pentru client. [11]

Un astfel de program a fost și invenția lui „Rasmus Lerdorf". Și-a extins programele ce gestionau pagina lui personală pentru a interacționa cu baze de date și formulare, dând numele implementării „Personal Home Page/Forms Interpreter" (sau pe scurt, PHP/FI). Nu a avut o viziune clară asupra limbajului de programare, a continuat să adauge ceea ce considera a fi următorul pas logic. [9]

Printre îmbunătățirile aduse limbajul se numără „superglobalele" (o metodă ușoară de a accesa parametrii trimiși de formularele paginilor HTML) în versiunea 4.1, metode de gestiune a codului orientat pe obiecte, o interfață standardizată pentru a accesa bazele de date. Ulterior, adăugându-se pe parcursul dezvoltării limbajului suport pentru „namespace"-uri, generatoare, funcții anonime, extinderi, pe 28 August 2014, a fost lansat PHP 5.6, ce va fi întreținut până pe 31 Decembrie 2018. [6]

2.2 Laravel 5.3

Laravel este o librărie gratuită de PHP. Pentru Laravel 5.3, minimul necesar librăriei este PHP 5.5.9 sau mai nou. Librăria a fost concepută de „Taylor Otwell", pentru a construi aplicații web bazate pe arhitectura „Model-View-Controller". Printre numeroasele avantaje de a folosi Laravel se află sistemul modular de pachete, cu propriul său sistem de gestionare, moduri diferite de a accesa bazele de date relaționale, utilitare pentru scrierea și menținerea aplicației și orientarea spre expresivitatea codului.[2]

Arhitectura „Model-View-Controller" se bazează pe cele trei componente inter-conectate slab ale unei aplicații: Modelul, Controller-ul, respectiv View-ul. Modelul este componenta centrală a șablonului, pentru că se ocupă de datele, logica și regulile aplicației. Acest model este gestionat de Controller, ce acceptă datele de la utilizator și le transformă în comenzi pentru model și/sau View. View-ul este orice metodă de vizualizare a informației unui Model.[3]

Avantajele arhitecturii, mulțumită separării conceptuale, ajută dezvoltarea simultană și ușurința modificărilor. Astfel, mai multe persoane pot lucra simultan la o parte a aplicației. Dar din păcate, trebuie studiată în detaliu, pentru că nu este ușoară de folosit pragmatic.

2.2.1 Directive Blade

Un mare avantaj al librăriei este un simplu, dar foarte puternic și eficient sistem de șablonare — Blade. Spre deosebire de alte sisteme de șablon, Blade nu inhibă dezvoltatorul a folosi cod PHP în View-uri. De fapt, toate View-urile Blade sunt traduse în cod simplu PHP și salvate până ce sunt modificate. Astfel reușește să adauge aproape zero logică în plus aplicației.

Pentru a afișa ceva pe ecran folosind PHP în mijlocul codului HTML, ar trebui să scrii:

```
<?= $variabila ?>
```

Dar asta te face vulnerabil la „XSS".

XSS (Cross Side Scripting) este o vulnerabilitate a securității unei aplicații, des întâlnită în aplicațiile web, ce permite adăugarea oricărui cod, fie el malițios sau inocent, paginilor vizualizate de clienții normali.

Astfel, pentru a preveni exploatarea datelor trimise de la Controller, pentru a afișa valorile variabilelor trimise, directivele Blade indică folosirea sintagmei:

```
{{$variabila}}
```

Desigur, nu ești limitat la doar nume de variabile, ci la orice s-ar putea considera parametrul unei funcții, deoarece sunt trimise prin metoda `htmlspecialchars` a PHP-ului.

Printre cele mai folositoare beneficii folosind Blade se află și construirea fișierelor ce definesc structura HTML-ului aplicației. Nu se definește practic o pagină principală, doar se specifică pentru paginile existente deja a se folosi structura găsită în pagina specificată prin comanda:

```
@extends('layouts.app')
```

unde în acest caz în folderul „layouts” găsim fișierul „app.blade.php”.

Pentru a introduce conținutul paginii ce extinde șablonul, Blade folosește secțiuni. Declarate în pagina principală, acestea indică locul unde se așteaptă conținut, putând fi denumite după nevoile șablonului. De asemenea, în cazul definirii unei secțiuni, se poate folosi și o valoare implicită. Cel mai des e folosit un șablon în definirea titlului, cu o valore implicită a numelui proiectului. În cazul acestui proiect, am folosit ca sufix numele configurat în fișierul mediu, lăsând la dispoziția fiecărei pagini a seta titlul.

Folosind secțiuni, am descoperit o problemă la refolosirea unor componente vizuale. Blade adaugă precum PHP conținutul fișierului ce trebuie să existe unde ai scris instrucțiunea respectivă folosind comanda:

```
@include('decisions.includes.search')
// ce adaugă fișierul
// resources/views/decisions/includes/search.blade.php
```

Continuând șirul dezvoltării, descopăr o altă problemă la introducerea a mai multor segmente de cod JavaScript în aplicație, din fișiere Blade separate. Pentru a rezolva și această problemă, Blade se folosește de:

```
@push('scripts')
```

pentru a adăuga informații într-o stivă, ce va fi afișată în pagina ce definește structura HTML prin simpla comandă:

```
@stack('scripts')
```

Printre marile îmbunătățiri aduse de Blade, folosite în acest proiect, se numără și directivele. Acestea sunt propriile comenzi, definite de programator, ce extind limbajul. Sunt traduse în simple instrucțiuni de cod PHP. Am scris astfel propria directivă de a construi automat codul necesar afișării, folosind HTML și librăria vizuală „Bootstrap”, a unui câmp dintr-un formular, cu nume, tip și validare proprie:

```
@input(['eveniment_strada', ['Strada', false]])
// unde 'eveniment_strada' - id-ul câmpului.
// unde 'Strada' - denumirea câmpului.
// unde 'false' - necesitatea completării câmpului.
```

2.2.2 Laravel Forms / HTML

O mare parte a oricărei aplicații web este interacțiunea cu clientul. Pentru asta, există formulare ce sunt trimise către calculatorul aplicației. O fostă librărie inclusă în Laravel, dar acum extinsă și desprinsă este librăria ajutătoare pentru formulare și HTML. Este întreținută de „Laravel Collective”, o comunitate ce se ocupă de menținerea componentelor scoase din Laravel de „Taylor Otwell”.

Astfel, el are prilejul de a se concentra pe munca sa (Laravel) și comunitatea susține componentele îndrăgite de programatori în viață, independent.[4]

Un astfel de proiect este „Forms & Html" [5], ce simplifică codul ce trebuie scris de un programator Laravel. În loc de a scris de fiecare dată ce câmp are ca referință ce obiect din model, se definește în schimb în momentul construirii formularului modelul de referință. Când se construiește un câmp, trebuie doar să fie bine specificat identificatorul câmpului, iar când va fi trimis către client, va fi completat automat cu valoarea din model. De asemenea, un al treilea parametru lasă loc programatorului să completeze celelalte atribute ale nodului HTML printr-un vector unde cheia reprezintă numele atributului și valoarea reprezintă conținutul.

Nu doar metode ajutătoare, „Forms & Html" acceptă și componente specializate, definite de programator, ca șabloane Blade. Prin definirea lor într-un folder special, acestea pot fi folosite precum cele deja definite, pentru a ușura clasele și atributele necesare pentru dezvoltatorii ce folosesc „Bootstrap". Cum acest proiect se folosește de Bootstrap și „Forms & Html", vă pot da un exemplu de două componente des folosite în aplicație:

```
{{Html::panel_begin('Delete Assurance', 'in')}}
  // aici scriu cod...
{{Html::panel_end()}}
// unde 'Delete Assurance' - numele panoului
// unde 'in' - starea panoului: de a fi deja deschis
```

2.2.3 Artisan

migrations - artisan - database migrations artisan make:commands artisan routes:list artisan make:auth

2.2.4 Eloquent ORM

models - Eloquent ORM

2.3 Baza de date - MySQL

relatii dbms. export usor shared hosting - phpmyadmin export / import usor testare. interfata speciala de solutie de backup pentru a fi rulata automat.

2.4 GitHub

Trebuie să ne asigurăm că păstrăm undeva codul sursă al aplicației, pentru a fi siguri că nu se vor corupe datele. De asta am ales să folosesc un program de gestionare a codului.[13]

Sistemul de gestiune a codului se ocupă cu organizarea și aflarea modificărilor fișierelor cu cod, construind revizii pentru fiecare modificare în parte, ușoare de

identificat. De asemenea, comunică cu un server ce păstrează codul sursă și lasă programatorii să aibă o variantă ce funcționează pe mașina locală.

Funcționând pe un modelul tranzacțional, ajută enorm lucrului în echipă pentru că păstrează pentru cele mai des folosite operații (salvare, vizualizare istoric, revenind la modificări anterioare) sunt optimizare și deci rapide. Modelul tranzacțional asigură și că nu vor apărea probleme la modificări concurente de două sau mai multe persoane asupra aceluiași fișier.

Un astfel de sistem de gestiune a versiunii (VCS), pentru a urmări schimbările fișierelor și coordonarea lucrului în echipă, este „Git".[16] A fost conceput de „Linus Torvalds" în 2005 împreună cu alți dezvoltatori ai kernel-ului LInux pentru dezvoltarea kernel-ului. Se concentrează asupra vitezei, integrității datelor și susținerea unui mod de lucru distribuit și neliniar. Astfel, fiecare folder „Git" pe fiecare calculator conține istoricul complet al proiectului, independent de accesul la rețea. Este cel mai folosit sistem, mulțumită multiplelor garanții împotriva coruperii datelor, iar conform unui studiu făcut în 2015, 69.3% dezvoltatori folosesc activ „Git".

Pentru că un sistem de gestiune a versiunii se folosește de un server, am ales varianta online și bazată pe web „GitHub", ce folosește „Git" și adaugă și funcționalitate pentru lucrul în echipă, precum [20] :

- „bug tracking" - gestionarea problemelor în cod.
- „feature requests" - gestionarea propunerilor pentru funcționalitate nouă.
- „task management" - cine, ce, când are de modificat în cadrul proiectului.
- „wiki" - un loc comun despre tot ce ține de documentația proiectului.

2.5 CloudFlare

solutia de https dinainte de ce n-a mai functionat - Chrome / Firefox blacklisting solutie - folosirea unui certificat valid (semnat), dar ce trece prin Cloudflare pentru a asigura ca totusi e HTTPS si green icon.

2.6 jQuery & jQuery UI

orice modificare rapida de butoane logica de administrare / incarcare rapoarte / export validare

2.7 Google Analytics

unde se blocheaza / ce partea a interfetei e cea mai des folosita.

2.8 Knockout.js

pentru a putea structura front-end-ul cu logica pe baza modelelor logica asemanatoare pentru modele, ce au valori diferite folosit mai ales la exportul backup-ului adaugarea pozelor ca utilizator

2.9 LaTeX

relatii dbms. export usor shared hosting - phpmyadmin export / import usor testare. interfata speciala de solutie de backup pentru a fi rulata automat.

2.10 Mentiuni speciale

2.10.1 Shared hosting

avantajul de protectie contra XSS prin disabling exec dezavantajul de a nu putea rula migrarile pe serverul de productia solutia de urcare pe productie solutia de a separa app de public_html pentru a preveni code exploit-ul.

2.10.2 PhpStorm IDE

avantajul de a putea vedea usor ce se intampla cu modelele dezvantajul magic method-urilor laravel rezolvat cu proiectul de github.

2.10.3 Organizarea cunostiintelor pe un kanban board - Trello

construirea unui workflow de a incarca fiecare versiune noua pe serverul de productie enable debug / maintenance mode aplicare migrari incarcare fisiere incarcare in public_html disable debug

2.10.4 Heroku

rolling release testare free

2.10.5 Amazon Web Services

de ce am ales AWS integrarea cu Laravel mai intai a fost thumbnail + actual image - imagemagick apoi ajungeam in probleme de push commands asa ca pana la urma am ales sa mergem pe salvare in chior (fara compression / ceva) pentru ca nu depaseam quota-ul de 5GB / luna estimat

2.10.6 npm

2.10.6.1 integrarea cu laravel - laravel elixir

2.10.6.2 ce asigura un up-to-date css / js file cu versioning

2.10.6.3 agregarea css / scss / js -> webpack (minifier) -> versioning ->

2.10.7 Composer

Capitolul 3

Structura Aplicației

3.1 Request life-cycle

user-ul introduce adresa www.fandu.uk/claims este redirectat catre https://fandu.info aplicatia laravel incarca environment-ul se conteaza la baza de date gaseste ruta pentru controller ruleaza guard-urile pentru controller-ul respectiv. ruleaza controller-ul pentru a primi un view ruleaza view-ul (si-l compileaza daca e necesar - blade) afiseaza pe ecran. user-ul incepe sa completeze daca este sa iasa din aplicatie, il intreaba daca vrea sa piarda tot ce a scris pana in acel moment. drip - mentine sesiunea activa - token csrf se face validarea client-side se re-incepe request-ul, unde se valideaza back-end side si dupa ruleaza controller-ul respectiv rutei s.a.m.d.

3.2 O scurta istorie a aplicatiei

3.2.1 Rapoarte - vizualizarea datelor => refactor

FileData - export to text / csv

3.2.2 adaugarea campurilor de undecided / resolved / repaired

3.2.3 legatura stransa dintre claim-sale

problemele aduse cum se match-uiau inainte

3.2.4 decuplarea

migrarea in pasi a datelor separarea conceptuala pastrarea in UI a generic sale-urilor. stergerea lor si refactoring-ul spre solutia actuala - de a modifica iframe-ul.

3.2.5 FileDataExcelAdaptor

& split-uirea codului pentru Facade in doua trait-uri (multumita PHP 5.4)

3.2.6 matchuirea pentru rapoartele jmeky

3.2.7 todo: includerea campului de external_id

3.2.8 todo: decuplarea in one-to-one pentru a arata mai dragut / scoate rapoarte mai complicate cand o sa se ceara

3.3 Baza de date

tehnologia folosita: MySQL cum sunt salvate naming scheme Laravel de a numi tabelele cu pluralul lower_case.

3.3.1 Assurance

3.3.2 Chat Claims

3.3.3 Claims

3.3.4 Decisions + parent decision (many-to-many)

3.3.5 Messages

3.3.6 Migrations

3.3.7 Password Resets

3.3.8 Photos

3.3.9 Products

3.3.10 Sales

3.3.11 Sessions

3.3.12 User

3.4 Apache & PHP

interactiunea intre el si php

3.5 MVC - Model View Controller

detalii despre abordare de ce consider eu ca e cea mai buna - principiul de separare a ”puterii” cum ma ajuta Laravel sa obtin asta

3.5.1 Controllers

web.php routing naming scheme router REST-ful.

3.5.1.1 Namespace-ul default

3.5.1.1.1 Claims

3.5.1.1.2 Helper

3.5.1.1.3 Json - Images

3.5.1.1.4 Messages

3.5.1.2 Admin

3.5.1.2.1 Claims

3.5.1.2.2 Decision

3.5.1.2.3 Home

3.5.1.2.4 Import

3.5.1.2.5 Json - Search

3.5.1.2.6 Reports

3.5.1.2.7 Sales

3.5.1.2.8 Products

3.5.1.2.9 Assurances

3.5.1.2.10 Export

3.5.1.2.11 Backup

3.5.1.2.12 Settings

3.5.1.3 Auth

faptul ca am modificat minimal ceea ce mi-a dat Laravel

3.5.1.3.1 ForgotPassword

3.5.1.3.2 Register

3.5.1.3.3 ResetPassword

3.5.1.3.4 Login

3.5.2 Models - Eloquent ORM

ce aduce in plus ca functionalitate Laravel jmekeria cu One-To-Many / Many-To-Many detalii despre getXxxAttribute() protected $hidden detalii despre ce am adaugat si cum functioneaza pentru urmatoarele modele getReadable - pentru a genera automat ceea ce poate sa vada administratorul aplicatiei - campuri.

3.5.2.1 Assurance

3.5.2.2 Claim

3.5.2.3 Decision

factory manual resolution to text updateToPay remaining cum se calculeaza chestiile

3.5.2.4 Message

3.5.2.5 Photo

despre Factory method - in ce categorie se incadreaza.

3.5.2.6 Product

faptul ca rotunjim la 2 valori price-ul

3.5.2.7 Sale

generic sale factory totale assurance / price

3.5.2.8 Views

blade chemat de Controller o idee generala a structurii

3.5.2.8.1 vendor pagination bootstrap

3.5.2.8.2 settings

backup

index

3.5.2.8.3 sales

includes

3.5.2.8.4 reports

includes

3.5.2.8.5 newclaim

includes

3.5.2.8.6 messages

3.5.2.8.7 mail - layout

3.5.2.8.8 claim

3.5.2.8.9 photos

3.5.2.8.10 sales

3.5.2.8.11 app layout

3.5.2.8.12 import

cu tot js-ul de rigoare

- steps
- types

3.5.2.8.13 errors - custom error handling

3.5.2.8.14 decisions

3.5.2.8.15 import

3.5.2.8.16 components - html panels

3.5.2.8.17 reports

includes

3.5.2.8.18 claims

auth

3.5.2.9 Resources - js / scss

3.5.2.9.1 bootstrap

3.5.2.9.2 timepicker

3.5.2.9.3 typeahead

3.5.2.9.4 chart.js

3.5.2.9.5 extensions

- postData
- postSimpleJson
- postJson

3.5.2.9.6 custom app.scss stylesheet.

3.5.3 functii ajutatoare - Helpers

3.5.3.1 FormHelper

3.5.3.2 Helpers

3.5.3.3 TemporaryFiles

mai ales folositoare pentru rapoarte

3.5.4 Request-uri

ClaimRegisteredReques ReportRequest StoreClaimRequest

3.5.5 .env ironment-ul

necesitatea unui app_key conexiunea .env - db conectiunea .env - mail system

3.5.6 Logging

3.5.7 Event - Notification - Mail system

3.5.8 Excel

3.5.8.1 Facade

3.5.8.2 FileData

3.5.8.3 FileDataExcelAdaptor

3.5.8.4 Adaptors

3.5.8.4.1 DailyAdaptor

3.5.8.4.2 ExcelAdaptor

3.5.8.4.3 ExcelHelper

3.5.8.4.4 AltexGalaxyAdaptor

3.5.8.4.5 SalesImportAdaptor

3.5.8.4.6 OldAdaptor

3.5.8.5 Outputers

3.5.8.5.1 StatisticsOutputer

3.5.8.5.2 DecisionsOutputer

3.5.8.5.3 ProductsOutputer

3.5.8.5.4 SalesOutputer

3.5.8.5.5 AbstractDateOutputer

3.5.8.5.6 ExcelOutputer

3.5.8.5.7 YearlyOutputer

NegativeSale

3.5.8.5.8 MonthlyOutputer

Capitolul 4

Manual de utilizare

4.1 Administrative interface

4.1.1 Claims view

4.1.1.1 Color code

4.1.1.2 Search

4.1.1.2.1 Navbar ID

4.1.1.3 View claim

invoice automatic search duplicate imei view data

4.1.1.4 Edit Claim

short notes reminders status comments decisions mandatory - cum / de ce s-a intamplat asta

4.1.1.5 Reminders

today past due

4.1.2 Import

4.1.2.1 Weekly Sale import

de la altex

4.1.2.2 Custom imports

Daily import - format-ul vechi de daily tinut de Ramona pentru fiecare an Sales Import - import de date de la ”decision” output-ul Fandu (excel stuff) -

autocompletare campuri mai mult Old - old metadata import from Etonia etc. - ce momentan foloseste doar IMEI

4.1.3 Sales

search coloane, access rapid buton de remove / remove all logic

4.1.3.1 View

poti sa adaugi product poti sa stergi product poti sa adaugi assurance poti sa stergi assurance

4.1.4 Decisions

quick search detailed search de ce poti sa stergi - pentru ca daca nu ai asociat sale-ul corect, poti sa dai undo fara sa sufere nimic baza de date.

4.1.4.1 View

claim / messages / add old decision edit claim in urma analytics campurile explicate calculatorul

4.1.5 Reports

type of export (deprecated csv / excel) date start / end progress bar metadata when loading - especially yearly reports chunking

4.2 Registering a claim

4.3 Responding to messages

4.4 Settings

hidden DBA feature - register client - https://fandu.info/register backup - chunking -> sql -> .tar.gz

4.5 Uploading photos

Capitolul 5

Concluzii

5.1 Business Continuity Plan

5.2 Teste automate

În lunile care vin, înainte de mă ocupa de sistemul modular de rapoarte, o să doresc să petrec o bună perioadă scriind teste automate.

Testele automate, mai ales „testele unitare", sunt o metodă de verificare a modulelor individuale, asocierilor datelor, procedurilor de modificare a datelor. Kolawa, Adam; Huizinga, Dorota (2007). Automated Defect Prevention: Best Practices in Software Management. Wiley-IEEE Computer Society Press. p. 75. ISBN 0-470-04212-5.

Acestea pornesc de la premiza că un modul detașat de modificările externe, odată testat, se va comporta corect și atunci când va fi folosit în conjuncție cu alte module, ce la rândul lor vor fi testate. Astfel, de la cele mai mici module, se asigură comportamentul corect al aplicației. De la nivelul microscopic de funcție, până la nivelul macroscopic de funcționalitate.

Doresc să extind aplicația să se folosească de codul gata scris de cadrul Laravel și de a obliga aplicația găzduită pe mediul de testare să se asigure că fiecare nouă linie de cod nu strică funcționalitatea deja existentă. Acest principiu se mai numește și „integrare continuă", propusă prima oară de Booch, Grady (1991). Object Oriented Design: With Applications. Benjamin Cummings. p. 209. ISBN 9780805300918. Retrieved 18 August 2014. și folosită de majoritatea dezvoltatorilor moderni.

5.3 Modularizarea tabelelor

5.3.1 Modificări necesare asupra structurii bazei de date

5.4 Sistem modular de rapoarte

Bibliografie

[1] Amazon. *Amazon Web Services S3*. URL: https://aws.amazon.com/s3/ (visited on 06/14/2017).

[2] Martin Bean. *Laravel 5 Essentials*. books.google.com, 2015. ISBN: 978-1785283017.

[3] Frank Buschmann. *Pattern-Oriented Software Architecture*. Volume 1. Wiley, 1996. ISBN: 978-0471958697.

[4] Adam Engebretson. *About The Collective*. URL: https://laravelcollective.com/about (visited on 06/14/2017).

[5] Adam Engebretson. *Forms & HTML*. URL: https://laravelcollective.com/docs/master/html (visited on 06/14/2017).

[6] The PHP Group. *PHP: Supported Versions*. URL: http://php.net/supported-versions.php.

[7] HotFrameworks. *Web framework rankings*. URL: https://hotframeworks.com/ (visited on 06/14/2017).

[8] IDRIX. *VeraCrypt Free Open source disk encryption with strong security for the Paranoid*. URL: https://www.veracrypt.fr/en/Home.html (visited on 06/14/2017).

[9] Rasmus Lerdorf. *Rasmus Lerdorf, Senior Technical Yahoo: PHP, Behind the Mic*. URL: https://web.archive.org/web/20130728125152/http://itc.conversationsnetwork.org/shows/detail58.html (visited on 11/19/2003).

[10] Fandu Holdings Ltd. *Fandu.UK*. 2015. URL: http://www.fandu.uk/claims (visited on 06/14/2017).

[11] Anne Nelson and William Nelson. *Building Electronic Commerce with Web-Driven Databases with Cdrom*. 1st. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2001. ISBN: 020174130X.

[12] J. R. Okin. *The Information Revolution*. Ed. Ironbound Press, 2005. ISBN: 0-9763857-4-0.

[13] Bryan O'Sullivan. *Mercurial: the Definitive Guide*. Sebastopol: O'Reilly Media, Inc., 2009. ISBN: 9780596555474.

[14] TAYLOR OTWELL. *Laravel - The PHP Framework For Web Artisans*. URL: https://laravel.com/ (visited on 06/14/2017).

[15] Taylor Otwell. *Taylor Otwell on Twitter: "Average cyclomatic complexity per method*. URL: https://twitter.com/taylorotwell/status/817494232541839367 (visited on 01/07/2017).

[16] Kathryn D. Scopatz Anthony; Huff. *Effective Computation in Physics*. O'Reilly Media, Inc., 2015. ISBN: 9781491901595.

[17] Warranty Solutions. *Raportul avariilor*. 2015. URL: http://wsgp.co.uk/claims/ (visited on 06/14/2017).

[18] Altex Romania S.R.L. *Regulament Mobile Protect*. URL: https://mediagalaxy.ro/regulament-mobile-protect (visited on 06/14/2017).

[19] S.C. CLAX TELECOM SRL. *Hosting | Servere dedicate | VPS - xServers*. URL: https://www.xservers.ro/ (visited on 06/14/2017).

[20] Alex William. *GitHub Pours Energies into Enterprise. Raises $100 Million From Power VC Andreessen Horowitz*. URL: http://techcrunch.com/2012/07/09/github-pours-energies-into-enterprise-raises-100-million-from-power-vc-andreesen-horowitz/ (visited on 07/09/2012).