Міністерство освіти і науки України
Сумський державний університет

Навчально-науковий інститут бізнес-технологій «УАБС»

Кафедра економічної кібернетики

КВАЛІФІКАЦІЙНА МАГІСТЕРСЬКА РОБОТА

на тему «АВТОМАТИЗАЦІЯ ПОДАННЯ ЗАЯВОК НА ОТРИМАННЯ СУБСИДІЙ І ПІЛЬГОВИХ НАДБАВОК»

Виконав студент 2 курсу, групи ЕК.м-61а
(номер курсу) (шифр групи)

Спеціальності 051«Економіка («Економічна кібернетика»)»

Марченко Р.В.
(прізвище, ініціали студента)

Керівник:д.е.н., Кузьменко О.В.
(посада, науковий ступінь, прізвище, ініціали)

Суми – 2018 рік

ЗМІСТ

ВСТУП

Субсидія – це безготівкова допомога родині за адресом, яка забезпечує погашення витрат з оплати житлово-комунальних послуг.

Пільги – це переваги встановлені законодавством або іншими нормативними актами, що надається особі, або групі осіб порівняно з іншими громадянами.

Сьогодні в країні є актуальною роботою з оформлення соціальних допомог для населення. Кожна людина стикається із потрібністю легко, зручно та швидко отримати право на субсидію, або пільгові надбавки. Але, процес подання документів в даний момент є клопітким та не зручним.

Складність даної роботи несе певні наслідки. По-перше, людина витрачає свій час та зусилля. По-друге, в наслідок масової необхідності велике навантаження йде на працівників у місцевих органах соціального захисту. По-третє, багато питань та проблем виникло під час минулого опалювального сезону, що призвело негативного ставлення громадян до нашої країни.

Сьогодні в нашій країні система пільг спрямована для вирішення декількох принципово різних завдань. Перше – це привілеї які надаються окремим категоріям населення, влада хоче виділити із загальної сукупності, не дивлячись від рівня доходу. Існування таких пільг значною мірою є великим тягарем для нашої системи соціального захисту населення країни. Тоді, друге завдання несе в собі підтримка соціально уразливих верств населення, яку можна вважати специфічною формою адресної соціальної допомоги. Дана державна політика щодо соціального захисту населення має виступати

планомірною дією, безпосереднім чи опосередкованим впливом на постійне життя суспільства нашої країни, стати підсумком великої кількості програм, законодавчих намірів та організаційних взаємодій. І повинна вона розроблятися з метою досягнення поставлених цілей або певного результату, хоча не завжди реалізується на практиці.

Таким чином проблема автоматизації подання субсидії та пільгової надбавки набуває великого значення. Саме для вирішення даної проблеми, направлене дослідження у даній випускній роботі.

Значить, об'єктом дослідження є процес надання субсидії та пільгових надбавок.

Предмет дослідження – автоматизація процесу подання заявок на субсидію та пільгові надбавки.

Мета випускної роботи полягає в розробці автоматизованої системи подання заявок на субсидію та пільгові надбавки.

Для досягнення мети потрібно виконати такі задачі:

– охарактеризувати основні засади подання заявок на субсидію та пільгові надбавки;

– проаналізувати стан подання заявок на субсидію та пільгові надбавки;

– сформувати основні вимоги до автоматизованої системи подання заявок на субсидію та пільгові надбавки;

– дослідити нормативно-правове забезпечення, що формує правову основу процесу подання заявок на субсидію та пільгові надбавки;

– розробити алгоритм вирішення поставленої задачі;

– обрати архітектуру та технологію для розробки автоматизованої системи;

– визначити програмне та апаратне забезпечення вирішення задачі;

– розробити структуру та визначити особливості реалізації інформаційного забезпечення системи;

– спроектувати інтерфейс користувача;

– визначити очікуваний економічний ефект від впровадження автоматизованої системи подання заявок на субсидію та пільгові надбавки.

Методологічну основу даної роботи сформували концепції економічної кібернетики. В процесі виконання використані такі методи наукового дослідження: аналіз, синтез, індукція, дедукція, зіставлення, порівняння.

Інформаційно-фактологічну базу дослідження сформували законодавчі та нормативні акти, що стосуються питань ведення роботи з персональними даними, наприклад, Закон України "Про захист персональних даних" прийнятий 1 червня 2010 року [23]. Крім цього фактологічну базу було сформовано на основі дослідження, яке проводилося на базі переддипломної практики, а саме у e-Delux GMBH м. Суми

Автоматизація подання заявок на отримання субсидій і пільгових надбавок.

1 ЕЛЕКТРОННА РЕЄСТРАЦІЯ СУПСИДІЙ ТА ПІЛЬГОВИХ НАДБАВОК

1.1 Загальна характеристика електронної реєстрації субсидій та пільгових надбавок.

На даний момент для відшкодування родинам витрат на виплату житла і комунальних послуг в країні запроваджено програму надбавок та житлових субсидій, вона діє також з існуючими пільгами і в ніякому стані не скасовує їх. З проектом Житлового Кодексу (ст. 23) субсидії надаються також для будівництва або придбання власного житла. Також, передбачається, що черговість видачі субсидій надається з урахуванням часу коли взяли на облік для покращення житлових умов. З даного твердження ми можемо зробити висновок, що черга на поліпшення умов зберігатиметься і в майбутньому, це діє не тільки на отримання соціального житла, а також для отримання субсидій та пільгових кредитів, як це прогнозує у проекті Житлового Кодексу, які надаються підприємствам, установам, організаціям, комерційним банкам під заставу нерухомості, гарантії місцевих органів виконавчої влади, органів місцевого самоврядування чи гарантії підприємств, установ, організацій. Згідно з даним законодавством, субсидія надаються, якщо виплати за житлово-комунальні послуги перевищує встановлений відсоток середньомісячного сукупного доходу. Це така допомога держави, яка надається для відшкодування грошових витрат для оплати житлово-комунальних послуг у вигляді безготівкової допомоги малозабезпеченим сім’ям, із за підвищення плати за житло та комунальні послуги. На отримання субсидії мають право ті сім’ї, які проживають в житлово-будівельних кооперативах, гуртожитках, приватному та громадському житловому фонді. Після покупки квартири субсидії та пільги надаються в такому же порядку, як і до приватизації. Видані субсидії при приватизації не повертаються.

Субсидія завжди призначається лише на жилу площу, яка не може перевищувати санітарної норми на родину. Норма володіння або користування даною площею житла та користування комунальними послугами встановлюються з 21 м2 на наймача і прописаного у даному приміщені чи будинку та додатково 10,5 м2 на тих хто прописаних, а для людей, що проживають в однокімнатній квартирі, — на всю площу, що не залежить від розміру квартири. Субсидія надається тільки одному з членів сім'ї власнику житла, наймачу, на якого відкрито особовий рахунок, членові ЖБК. Розмір виплат визначається постановами Кабінету Міністрів України, наказами Міністерств: фінансів України, праці і соціальної політики, інших органів центральної виконавчої влади. Вони можуть також визначатися та надаватися органами місцевого самоврядування.

Прядок видачі субсидій визначається Постановою Кабінету Міністрів України "Про спрощення порядку надання населенню субсидій для відшкодування витрат на оплату житлово-комунальних послуг, придбання скрапленого газу, твердого та рідкого пічного побутового палива" від 21 жовтня 1995 р. за № 848 (в редакції Постанови Кабінету Міністрів України від 22 вересня 1997 р. за № 1050).

Дане положення визначає умови та порядок надання громадянам кожний місяць безготівкової субсидії для погашення витрат на оплату користування житлом або комунальними послугами водопостачанням, теплопостачанням, газопостачання, електроенергія, водовідведення,та інше. Отримання субсидії поширюється на громадян, які проживають у гуртожитках, — на оплату користування житлом. Відділами управліннями субсидій, а також контроль за їх використанням здійснюються субсидій районних. Вони призначаються за різниці між платою за житлово-комунальні послуги, скраплений газ, тверде та рідке пічне побутове паливо у межах норм споживання з урахуванням пільг, які надаються відповідно до чинного законодавства, і обсягом обов'язкового відсотка платежу, визначеного Кабінетом Міністрів України.

Субсидія не призначається в таких випадках:

– у житловому приміщені де є прописані працездатні громадяни працездатного віку, які не працюють а також не навчалися на денних відділеннях вищих і професійно-технічних навчальних закладів, де навчання зараховується до стажу, в період до трьох місяців, що передують звернення за призначенням субсидії крім громадян, які доглядають за дітьми до досягнення ними трирічного віку; також доглядають за дітьми і час догляду яких зараховується до трудового стажу; громадян, які доглядають за дітьми, що потребують догляду, визначеного у медичному висновку лікувально-консультаційної комісії, але не більше ніж до досягнення ними шестирічного віку; також громадян, що мають більше трьох дітей віком до16 років та доглядають за ними;люди які доглядають за інвалідами першої групи або дітьми-інвалідами віком до 16 років, за людьми які досягли 80-річного віку, та такі, що не стоять на біржі труда;

– власник житла, або наймач житлового фонда, власник приміщення, в якого відкрито особистий рахунок, також що прописана разом у приміщенні, яке здає за договором у найм або оренду жиле приміщення;

– власник або співвласник приміщення, наймач житла, власник приміщення де живуть, в якого відкрито особовий рахунок, а також ті прописані разом з ним у будинку, мають у своєму володінні або у володінні дружини разом більш ніж одне жиле приміщення, загальна площа яких у сумі перевищує встановлені нормами житла;

– також має у власності, або у власності дружини транспортний засіб, машину або механізм, крім засобів, що не є об’єктами оподаткування, також який є в експлуатації не більш як 10 років, починаючи з року випуску, та зареєстрований в установленому порядку.

Завдяки новому порядку на 2018 рік, призначення житлової субсидії відбувається без врахування майнового стану здобувача. Винятком є, здійснена протягом 1 року перед зверненням за субсидією, одноразова

покупка вартістю понад 50 тисяч гривень. Таким чином, право на отримання субсидії набувають і ті категорії громадян, які раніше були її позбавлені:

- орендодавці;
- власники кількох автомобілів;
- власники двох житлових приміщень;
- люди, які зробили покупку, а також, якщо оплатили послугу(навіть за навчання або лікування) більше ніж десять прожиткових мінімумів.

Але, люди що мають бажання отримати допомогу від держави для подолання комунальних потре, мають весь час стежити за змінами, адже “правила гри” не є постійними. Дивлячись на травень 2015 року, коли Кабмін нашої держави затвердив нову систему призначення житлової субсидії, не один раз ще змінювалися вимоги до громадян.

Великим збільшенням є кількість тих, хто розраховує на допомогу по виплаті комунальних послуг. Як наслідок, державі потрібно збільшувати “житловий” бюджет. субсидія є безповоротною державною допомогою, а її отримання не пов’язане і не тягне за собою зміни власника житла. Саме головне не потрібно приховувати інформацію про свої доходи. Деякі вважають, що надають субсидію лише тим людям які живуть у злиднях, наприклад в комунальній квартирі. Це зовсім не так. В загалом, можна мати недешеве майно, автомобіль і законно розраховувати на допомогу від держави в оплаті комунальних послуг.

Нові правила отримання житлової субсидії на 2018 рік. За законодавством України право на отримання державної субсидії мають сім’ї, чиї витрати на оплату житлово-комунальних послуг (в межах встановлених норм споживання та з урахуванням пільг) перевищують розмір обов’язкового відсотка платежу. Сума платежу для кожного домогосподарства визначається індивідуально і залежить від сукупного доходу сім’ї.

Постановою Кабінету Міністрів № 319 від 27.04.2016 внесено зміни до порядку надання пільг та субсидій на оплату житлово-комунальних послуг. Одним з нововведень уряду є пропозиція тимчасової відміни пільг на

квартиру. Таким чином, на період отримання субсидії громадянам, які мають пільги на оплату житлово-комунальних послуг, вони нараховуватися не будуть. Проте, розрахунок субсидії буде проводитися з урахуванням пільг на сплату послуг і обсяг обов'язкового платежу за послуги ЖКГ буде залежати виключно від доходів сім'ї.

Житлово-комунальна субсидія оформляється на одного з зареєстрованих членів домогосподарства (сім'ї), на ім'я якого відкрито особові рахунки на оплату комунальних послуг. Решта членів домогосподарства для призначення субсидії повинні підтвердити свою згоду на доступ і обробку персональних даних. Квартиронаймачі, які сплачують комунальні послуги відповідно до договору оренди, також мають право на надання субсидії.

Право на отримання субсидії готівкою на придбання скрапленого газу, твердого та рідкого пічного побутового палива надається тим громадянам, чиє житло не забезпечується електро, тепло або газопостачанням для обігріву. При цьому, якщо житлові приміщення опалюються одночасно декількома способами (теплова енергія та / або природний газ і / або електроенергія), субсидія призначається і виплачується тільки на один вид палива.

Відповідно до програми житлових субсидій, держава відшкодовує витрати на оплату користування житлом та комунальних послуг (тепло, газо, водопостачання, водовідведення, електроенергії, вивезення побутового сміття та рідких нечистот). Крім того, один раз на рік, для сімей, які використовують для опалення тверде паливо, а для приготування їжі скраплений газ, надається грошова субсидія на їх купівлю (Положення про порядок призначення та надання населенню субсидій для відшкодування витрат на оплату житлово-комунальних послуг, придбання скрапленого газу, твердого та рідкого пічного побутового палива).

Постановою Кабінету Міністрів України №409 від 06.08.2014 «Про встановлення державних соціальних стандартів у сфері житлово-

комунального обслуговування» прийняті єдині для всієї України норми житла та споживання комунальних послуг, на які може бути призначена субсидія. Ознайомитися з вищевказаними стандартами на 2018 рік можна нижче.

Таблиця 1.1 – Електроенергія

51 кВт г на 1 м2 площі	На опалення житла (в опалювальний період)
110 кВт на сім'ю з 1 особи + 30 кВт г на наступного мешканця	У будинках, обладнаних стаціонарними електроплитами (при наявності гарячої води)
130 кВт на сім'ю з 1 особи + 30 кВт г на наступного мешканця	У будинках, обладнаних стаціонарними електроплитами (при відсутності гарячої води)
70 кВт на сім'ю з 1 особи + 30 кВт г на наступного мешканця	У будинках, не обладнаних стаціонарними електроплитами (при наявності гарячої води)
100 кВт на сім'ю з 1 особи + 30 кВт г на наступного мешканця	У будинках, не обладнаних стаціонарними електроплитами (при відсутності гарячої води)

Таблиця 1.2 – Вода

Єдині соціальні нормативи	
2,4 м3 на людину	Централізоване постачання холодної води
1,6 м3 на людину	Централізоване постачання гарячої води
4,0 м3 на людину	Централізована холодна вода (за відсутності гарячої)
4,0 м3 на людину	Ценралізоване водовідведення

Таблиця 1.3 – Житло

Єдині соціальні нормативи	
48,87 м2	Норма на сім'ю з 1 особи
62,52 м2	Норма на сім'ю з 2 осіб
76,17 м2	Норма на сім'ю з 3 осіб
89,82 м2	Норма на сім'ю з 4 осіб

Таблиця 1.4 – Вода

Єдині соціальні нормативи	
5,0 м2 площі	На опалення житла (в опалювальний період)
3,3 м3 на людину	На приготування їжі (газова плита)
5,4 м3 на людину	Приготування їжі та підігрів води (якщо немає гарячої води і газового водонагрівання)
10,5 м3 на людину	Приготування їжі та підігрів води (газова плита при наявності газового водонагрівача)

Кабмін переглянув соціальні нормативи споживання житлово-комунальних послуг для надання пільг і субсидій. Зокрема, для сімей, що складаються з 1-2 непрацездатних осіб (в тому числі пенсіонерів), соціальна норма житла, на яку надається субсидія була збільшена з 49 кв. м. до 75 квадратних метрів на домогосподарство (Постанова Кабінету Міністрів України № 300 від 26.04.2017).

Що, на рахунок пільг, то в Податковому кодексі України сказано, що дане поняття – це часткове зменшення розміру податків на доходи. Всі, хто вчасно сплачує податки, може використати цю пільгу і податки сплачувати не в повному обсязі. Ця допомога від держави допоможе урівняти прибутки різних соціальних груп.

У 2018 році на пільги мають право усі громадяни нашої держави, які вчасно платять податки. Ті, хто отримує платню нелегально або вона платиться у конверті, не може претендувати на цю пільгу. Але є деякі обмеження, дохід не має бути більшим, за прожитковий мінімум працездатних громадян. Кількість цих коштів множать на 1,4, а потім округлюють до 10 гривень. Це і є граничний розмір доходу. За допомогою цієї формули і вираховується соціальна пільга 2018 року.

Зарплата у співвітчизників поступово збільшується, як прийняли відповідний закону про бюджет, а із-за цього й соціальна пільга збільшується. Так, при мінімальній зарплаті 3200, гранична сума, при якій

можна претендувати на пільгу, складала 2240 грн. Ця сума залежить від прожиткового мінімуму, і повинна відповідати нормам ПКУ.

Підвищена пільга на відрахування податків:

– 150% пільгу можуть оформити одинокі батьки, яким доводиться самостійно виховувати дитину чи дітей, це стосується також вдів та вдівців, опікунів прийомних дітей. Розрахунки проводять на всіх неповнолітніх дітей;

– батьки, які виховують дітей з інвалідністю. Розрахунок проводиться відповідно на кожну дитину;

– громадяни, які підпадають і під першу, і під другу категорію ті,що постраждали від аварії на ЧАЕС;

– громадяни що навчаються мають право на отримання пільги;

Податкова соціальна пільга 2018 не потребує збору і подачі якихось особливих документів. Потрібні документи зведено до мінімуму. Потрібно лише подати відповідну заяву за місцем роботи. Також, якщо пільга дається на дітей, то потрібні відповідні документальні підтвердження.

За чинним законодавством кожен громадянин має право використовувати лише одну соціальну пільгу, на скільки б інших він не мав права. Це не стосується сімей, які використовують одночасно пільгу на неповнолітніх дітей і дітей-інвалідів, вони просто додаються між собою. Роботодавці не мають права не приймати заяв від своїх працівників на отримання пільги.

Багато громадян України стикаються з серйозними фінансовими проблемами, обумовленими економічною кризою. Це призводить до зацікавленості отримання фінансової допомоги з боку державних властей. Насправді українська влада стикається з серйозними проблемами при наданні допомоги соціально незахищеним громадянам, так як економічна ситуація в країні є далеко неоднозначною. Незважаючи на подібні нюанси, багато громадян цікавляться, яка буде соц. пільга у 2018 році в Україні та на що в дійсності можна розраховувати. У 2018 році має відбутися перехід до повної монетизації пільг в Україні. Передбачається, що пільговики можуть

претендувати на отримання пільг в натуральному вигляді, так як державна влада зіткнулися зі значними заборгованостями і економічними проблемами.

Незважаючи на серйозні проблеми в українській економіці і нестачу коштів для надання пільг, соціальна допомога в Україні в 2018 році як і раніше буде доступна і багато громадян зможуть розраховувати на державну підтримку в фінансових питаннях. Видатки на соціальні допомоги, пільги і субсидії загалом складуть 122,7 млрд. грн. проти 117,6 млрд. грн.. у цьому році. Це передбачено проектом Державного бюджету України на 2018 рік. Зокрема, 55,1 млрд. грн. з державного бюджету будуть спрямовані на адресні субсидії на оплату житлово-комунальних послуг, 7 млрд. грн. з місцевих бюджетів – на пільги на оплату житлово-комунальних послуг. 2,7 млрд. грн. передбачено у бюджеті на надання пільг та субсидій на придбання твердого палива і скрапленого газу. 57,9 млрд. грн. закладено в проекті Державного бюджету на 2018 р. на виплату малозабезпеченим сім'ям та іншим отримувачам соціальної допомоги. При цьому соціальні стандарти зростатимуть швидше за інфляцію.

Розрахунок граничного розміру доходу, який дає право на застосування пільги у 2018 року може бути застосована до заробітної плати, розмір якої не перевищує суму прожиткового мінімуму на одну особу в розрахунку на місяць станом на 1 січня, помножену на 1,4 та округлену до найближчих 10 гривень Категорії працівників, до заробітної плати яких можуть бути застосовані пільги – не змінилися, але розміри ставок , відповідно, зросли.

З 1 січня 2018 р. встановлено мінімальну заробітну плату в місячному розмірі — 3723 грн. і 22,41 грн. — у погодинному розмірі (ст. 8. Закону про бюджет 2018). Як і в минулому році, у 2018 межа застосування ПСП нижча від мінімальної зарплати. Таким чином, у 2018 р. механізм застосування ПСП зберігається, і якщо, наприклад, працівник працює на 0,5 ставки мінімальної зарплати, він може скористатися правом на ПСП.

Категорії платників податку, які мають право на податкову соціальну пільгу	Розмір ПСП, %	Сума у 2017 р., грн	Сума у 2018 р., грн
Будь-які платники податку, чий оподатковуваний дохід за місяць не перевищує 2240 грн (п. 169.1.1 ПКУ)	100	800	881
Платник податку, який утримує двох чи більше дітей віком до 18 років (у розрахунку на кожну таку дитину); (п. 169.1.2 ПКУ)	100	800	881
Одинокі батьки, вдови (вдівці), опікуни, утримувачі дитини-інваліда — на кожну дитину віком до 18 років. Чорнобильці першої або другої категорій. Інваліди I або II групи, у тому числі з дитинства. Інші. Повний перелік дивіться у п. 169.1.3 ПКУ.	150	1200	1321,5
Герої України, СРСР, Герої Соціалістичної Праці. Учасники бойових дій, інваліди I і II групи, колишні в'язні концтаборів. Повний перелік наведено у п. 169.1.4 ПКУ.	200	1600	1762

Рисунок 1.1 – Категорії працівників і розміри ставок, до яких вони застосовуються

1.2 Аналіз стану впровадження електронної реєстрації субсидій та пільгових надбавок

На даний момент попередня реєстрація документів на субсидії та пільги вже існує для всіх. Міністерство соціальної політики впровадило он-лайн послугу що, дає змогу підготувати до початкової реєстрації документів які в свою чергу призначають надання субсидії. Завітавши на портал subsidii.mlsp.gov.ua це все можна зработи. Для того щоб подати он-лайн реєстрацію документів в електронній формі потрібно лише підтвердити емейл адрес своєї пошти. Даний вид подання документів для подання в цьому вигляді дає людині можливість правильно заповнити, затвердити, зберегти заяву і декларацію для звернення у подальшому до управління соціального захисту населення. Є, звичайно, підказки які користувач отримую від он-лайн сервісу для заповнення документів . А навіть якщо

людина зробить помилку, то система їй повідомить які поля необхідно виправити.

До основних функцій електронної послуги міністерства соціальної політики про призначення житлової субсидії належить:

– графічний інтерфейс користувача;

– направити документи в електронній формі для попереднього розгляду;

– після опрацювання електронних документів та підтвердження права на призначення субсидії необхідно подати готові документи особисто;

– заповнити та роздрукувати документи для особистої подачі в орган соціального захисту населення;

– реєстрація за адресою електронної пошти;

– вхід за адресою електронної почти;

При переході до заповнення електронної форми декларації про доходи і витрати осіб, які звернулися за призначенням житлової субсидії, відкриється форма, що зображена на рис. 1.2

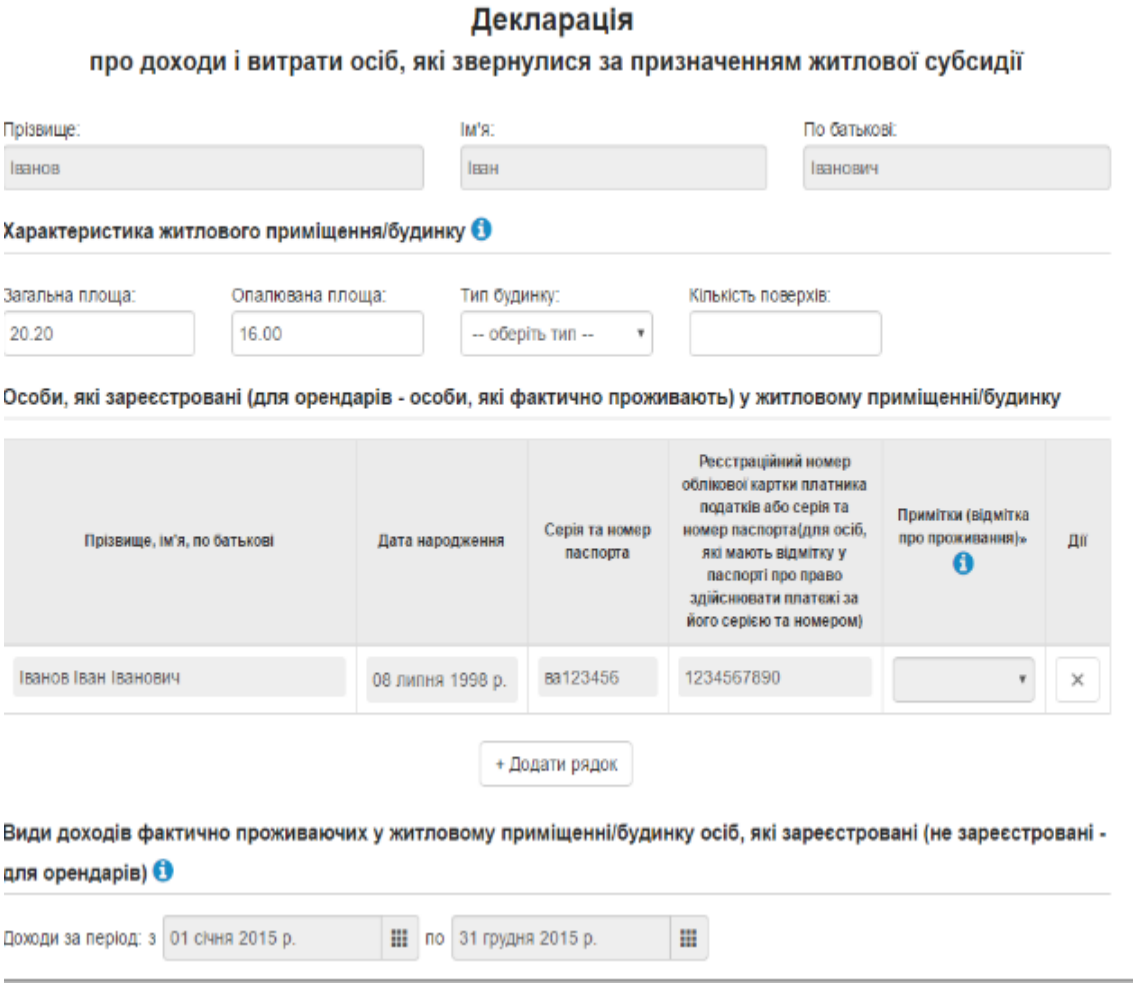


Рисунок 1.2 – Електронна форма

ПІБ особи автоматично створюється на підставі вже введених вами даних у заяві та не може бути відредагованим.

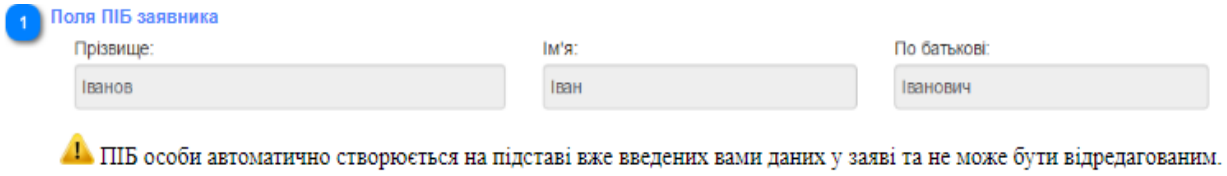


Рисунок 1.3 – Редагування даних форми

Декларація, складається з наступних блоків:

– характеристика житлового приміщення/будинку;

– особи, які зпреєстровані (для орендаторів – особи, які фактично проживають) у житловому приміщенні.будинку;

– види доходів фактично проживають у житловому приміщенні/будинку;

– інформація про витрати на придбання майна, товарів або оплату послуг на суму, що перевищує 50 тис. гривень, які здійснені протягом 12 місяців перед зверненням за призначення житлової субсидіїї.

Також на даному сайті є призначення субсидії в електронному виді. Дана функція працює тільки в деяких областях, а саме у Вінницької, Волинської, Донецької. В загалом, заявник може подати документи для призначення субсидії в електронній формі (в даному випадку відсутня необхідність візиту до органу соціального захисту населення та надання документів у паперовій формі). У випадку входу до Системи за допомогою електронної пошти, поля ПІБ заповнюються користувачем вручну. Поля "Прізвище" та "Ім'я" є обов'язковими для заповнення. У разі входу до Системи за допомогою ЕЦП чи BankID, дані поля заповнюються автоматично без можливості їх зміни.

Не всі ВНЗ про це інформують, але студенти, які проживають в гуртожитку, мають право на оформлення субсидії. Згідно постанови уряду про спрощення порядку надання населенню субсидії для відшкодування витрат на оплату житлово-комунальних послуг, придбання скрапленого газу, твердого та рідкого пічного побутового палива, на допомогу може розраховувати кожен, хто проживає в гуртожитку й сплачує за комунальні послуги більше 15% щомісячного сукупного доходу. Це Положення визначає умови призначення та порядок надання громадянам щомісячної адресної безготівкової субсидії для відшкодування витрат на оплату управління багатоквартирним будинком, користування житлом або його утримання, послуг з транспортування та розподілу природного газу та комунальних послуг (водо, тепло, газопостачання, водовідведення, електроенергія, вивезення побутового сміття та рідких нечистот), а також один раз на рік субсидії готівкою на придбання скрапленого газу, твердого та рідкого пічного побутового палива. Складністю розрахунку для студентів є те, що

субсидія нараховується не на всю вартість проживання, а лише на житлово-комунальні послуги. Їхня вартість уже включена в ціну проживання в гуртожитку. Точні дані можна дізнатися в бухгалтерії ВНЗ, але отримати субсидію можна й без них. У разі її оформлення органи соціального забезпечення самі зроблять запит у ВНЗ щодо вартості комунальних послуг. Даний сайт не в змозі реєструвати студентів, для отримання даної пільги.

1.3 Формування вимог до веб-орієнтованої інформаційної системи «Автоматизвції субсидій та пільгових надбавок»

Незважаючи на велике розмаїття програмних систем, що застосовуються для керування контентом, актуальним завданням залишається дослідження, розробка та вдосконалення засобів універсального керування інформаційними об’єктами Web-ресурсів, що дозволить спростити створення нових ресурсів різного призначення, а також забезпечить ефективні механізми їх супроводження та налаштування. У навчальному виданні розглянуто ключові засади створення сучасних Web-орієнтованих інформаційних систем.

Розглянемо дану задачу для вирішення автоматизованої системи, яка реалізує інформаційну технологію виконання встановлених функцій за допомогою персоналу і комплексу засобів а також, дасть людям змогу з реєстрації та оформлення он-лайн субсидій та пільгових надбавок.

Найбільшою проблемою, яку треба подолати у цій автоматизованій системі, є проблема реєстрації субсидій та пільгових надбавок.

В цьому разі, програмний веб-додаток має бути зробленим у простій і водночас зрозумілій формі яку будуть використовувати для користування, що дозволить швидко орієнтуватись та працювати з веб-додатком.

Також, дуже важливою особливістю є крос-платформне програмування яке дасть змогу користуватися системою на різних браузерах. Крос-платформний вигляд веб додатку можна досягли лише за

допомогою кросбраузерності css. Тим, хто знайомий з HTML, не потрібно розповідати, що це мова розмітки сторінки. Але його творці вирішили додати теги, що відповідають за зовнішній вигляд і дизайн. Ось тільки при такому підході код сторінки ставав занадто об'ємним і практично нечитабельним. Природно, цей шлях вів у нікуди, тому було прийнято комплексне рішення: HTML відповідає за розмітку сторінок, CSS – за візуальне оформлення. Блокова розмітка сторінки засобами CSS має кілька незаперечних переваг. По-перше, стиль об'єктів знаходиться окремо від HTML документа, що значно підвищує читабельність коду і дозволяє швидко проводити візуальні зміни. По-друге, є можливість накласти один шар на інший, а це в кілька разів полегшує процес позиціонування. Природно, такі сайти швидше завантажуються і індексуються пошуковими системами. Розмітка сторінки в CSS дозволяє легко задавати сучасні візуальні ефекти. Єдиний недолік такого підходу – різне «розуміння» браузерами. Деякі відображають ресурс в такому вигляді, в якому бачить його веб-майстер. Але є браузери, що викривляють зображення, тому при великій кількості класів та селекторів необхідно використовувати методи, що зроблять код кроссбраузерним.

Для збереження даних які користувач буде заповняти ми будемо використовувати MySQL.

Крім цього, завдяки використанню веб-додатку, користувач матиме змогу здійснювати реєстрацію субсидій та пільгових надбавок зі свого смартфона, планшета, персонального комп’ютера, або ноутбука швидко та зручно, оскільки дані пристрої завжди біля користувача.

Отже, вимоги – це можливості або умови, яким повинна відповідати система чи проект. Основне завдання етапу визначення вимог – знайти, обсудити і зафіксувати, що дійсно вимагається від системи у формі, яка буде зрозуміла і клієнтам і розробникам [20, с. 89].

Тобто, необхідно розробити перший пакет вимог до системи. Далі, у розробці та тестуванні веб-додатку потреби будуть уточнюватись і доповнюватись. Даний підхід допоможе максимально точно забезпечити

виконання вимог користувача, і є також є із основних засад уніфікованого підходу. У проекті ми повинні стежити за стрункістю і мінімалізмом архітектури використовувавши DRY, KISS. Повинні писати як на сучасній мові програмування, поділяючи на публічні ресурси і файли програми. Завжди валідувати будь які користувальницькі введення, екрануючи висновок. Потрібно стежити за відсутністю сміття в коді і називати методи і змінні так, щоб по одній назві було зрозуміло що до чого.

Більш детальну інформацію про дані підходи та розробку можна знайти у наступних розділах даної випускної роботи.

Отже, виходячи із бачення продукту, можна виділити ряд правил високого рівня. Ці правила забезпечують про якість програмного засобу. До них відносяться:

1. програмний веб- додаток має виконувати свою головну функцію, а саме реєстрацію субсидій та пільгових надбавок;

2. дана система має бути зручна для користувача і задовольняти його потреби, пов'язані зі сферою роботи веб-додатку;

3. веб-сайт має допомагати людям заощаджувати свій час, а також економічний стан;

Всі інші вимоги, які деталізують сам програмний продукт, можуть змінюватись у рамках уніфікованого підходу та ітеративної розробки. Отже, доцільним є розгляд вимог за певною системою. Однією із таких систем, яка гарно себе зарекомендувала у світовій практиці розробки програмних продуктів, є система FURPS+.

Суть її полягає в тому, що програмний додаток варто робити на функціональні вимоги, які пристосовуються до можливостей системи а також властивостей які у свою чергу поділяються на вимоги зручності, надійності, продуктивності, та можливості підтримки.

Отже, за даною моделлю було складено перший пакет вимог до програмного продукту:

4. вимоги що, до функціонування:

1) Ведення користувачем. Весь процес реєстрації має бути зручним для користувальницького введенням, щоб повною мірою контролювати користувача.

2) Персоналізація. Для кращої диференціації інформації потрібно створити систему персоналізації користувача.

3) Доступність і надійність особливо важливі для веб-додатків з великою кількістю користувачів. Додатки повинні бути здатні витримувати великі навантаження і зобов'язані працювати навіть в нестандартних ситуаціях.

4) Аналіз сайту. Адміністратор повинен мати можливість виконувати та переглядати проаналізовану інформацію у різних форматах.

5) Швидкість веб-додатка. Програма має швидко реагувати на різні запити.

6) Гнучкість системної архітектури. Вона має підтримувати можливість для змін у контенту структуру додатку без негативного впливу на другі частини даної структури.

5. зручність веб додатку:

1) Юзабіліті означає, що користувач може працювати ефективно, зручно і при цьому ще отримувати задоволення від роботи програми. Доступність є частиною юзабіліті.

2) Люди повинні легко розуміти, що їй потрібно зробити, щоб отримати результат без будь яких проблем.

3) Повна та зрозумілість.

6. вимоги надійності:

1) Частота збоїв. Частота збоїв має бути зведена до мінімуму, у разі виникнення помилки, користувачу має бути пред'явлена зрозуміла коротка інформація про проблему та інструкції щодо її подолання.

2) Стійкість бази даних. Процес взаємодії із базою даних має бути спроектований таким чином, щоб при жодних обставинах не відбувалася втрата інформації або блокування її.

7. вимоги продуктивності:

1) Час відгуку. Час відгуку має зведений до мінімуму, або не більше 100 мс. Для деяких операції припускається трохи збільшений час відгуку.

2) Точність. Програмна система має безпомилково записувати всі дані, які вказав користувач, та здійснювати їх аналіз.

3) Доступність. Робота з веб-додатком має бути доступна для користувача у будь-який час.

4) Використання ресурсів. веб-додаток повинен добре індексуватися гугл аналітикою.

8. вимоги можливості підтримки:

1) Конфігурування. Програмна система має підтримувати систему конфігурації, для досягнення кращої відповідності потребам користувача.

2) Он-лайн підтримка веб додатку. Для відповідей на питання, у програмі має бути зазначена електронна адреса розробників.

Основна мета забезпечення якості, а точніше, управління якістю програмного забезпечення, що - зробити витрати і вигоди прозорими для всіх сторін, що беруть участь у створенні програмного забезпечення. Низька якість програми в довгостроковій перспективі викликає додаткові витрати. Якщо ці витрати можуть бути визначені кількісно, то потрібно зробити висновок, що досягнення кращої якості призведе до зниження витрат в майбутньому. Це, мабуть, єдиний спосіб змусити керівництво або клієнта розглянути питання про виділення бюджету для рефакторинга коду.

Web-технологія повністю перевернула уявлення про роботу з інформацією, та й з комп'ютером взагалі. Виявилося, що традиційні параметри розвитку обчислювальної техніки - продуктивність, пропускна

здатність, ємність запам'ятовуючих пристроїв - не враховували головного "вузького місця" системи - інтерфейсу з людиною. Застарілий механізм взаємодії людини з інформаційною системою стримував впровадження нових технологій і зменшував вигоду від їх застосування. І тільки коли інтерфейс між людиною і комп'ютером був спрощений до природності сприйняття звичайною людиною, по слідував безпрецедентний вибух інтересу до можливостей обчислювальної техніки.

Інформація, доступна користувачам інтернет, розташовується на Web-серверах. Значна частина цієї інформації організована у вигляді веб-сайтів. Кожен з них має свою адресу. Веб-сайт - це інформація, представлена в певному виді. Для перегляду веб-сайтів на комп'ютері користувача використовуються спеціальні програми, які називаються браузерами. Найбільш поширеними браузерами в даний час є Google Chrome, Opera Mozilla Firefox. В залежності від того, яку адресу ми задамо в рядку браузер буде завантажувати в своє вікно відповідну інформацію.

РОЗДІЛ 2 РОЗРОБКА ВЕБ-ОРІЄНТОВАНОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ «ПОДАННЯ ЗАЯВОК НА ОТРИМАННЯ СУБСИДІЇ І ПІЛЬГОВИХ НАДБАВОК»

2.1 Архітектура автоматизованої системи

Дивлячись на основні правила розробки веб-додатку сформуємо наступну архітектуру автоматизованої системи. Прототип автоматизації подання заявок на отримання субсидій і пільгових надбавок будемо розробляти на мові програмування PHP.

PHP (Hypertext Preprocessor) — це мова програмування яка має велику популярність, має загальне призначення з відкритим кодом виходу. Він орієнтується для розробки веб-додатків, його код можливо інтегрувати в HTML. PHP додатки обробляються на сервері. Він простий для вивчення, але може задовольнити запроси професійних програмістів. Розглянемо як PHP працює на схемі:

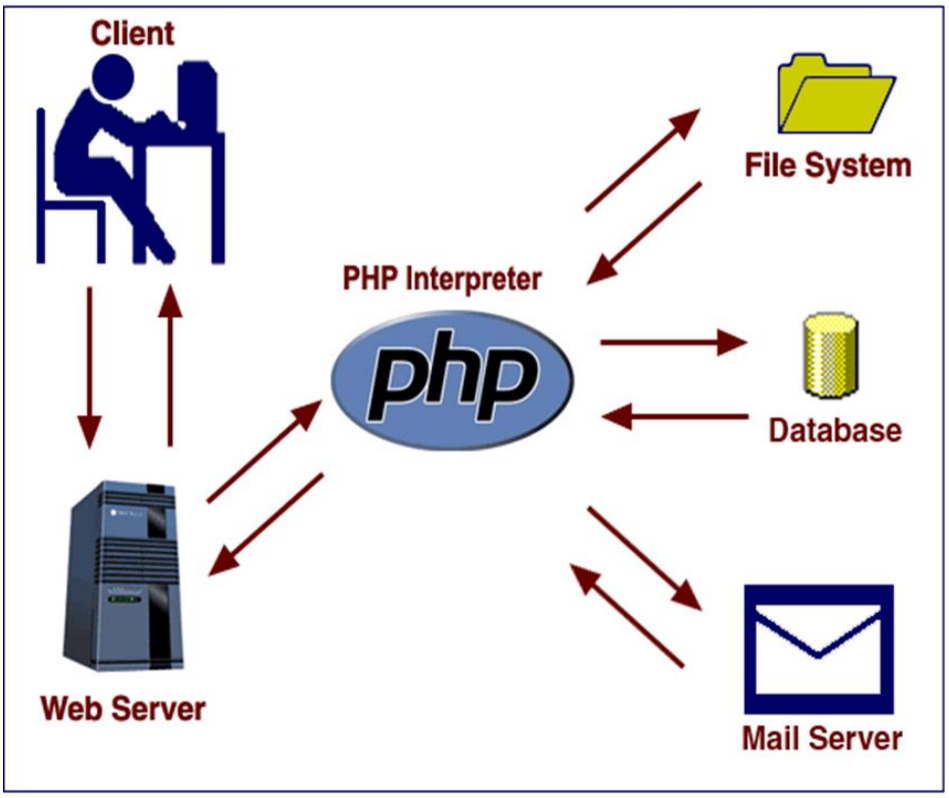


Рисунок 2.1 – Робота PHP

Опишемо (рис. 2.1) для повного розуміння:

- веб-додаток отримує запит;
- потім веб-сервер відправляє запит через PHP, який вирішує які файли будуть допустимі, або значення бази даних;
- далі мова PHP складає пакети, а потім створює HTML сторінку яка відається користувачу.

Для зручності та правильності написання програмного додатку ми будемо використовувати фреймворк Laravel. Для нашого прототипу подання заявок на отримання субсидій і пільгових надбавок, доступні маршрути, визначені в routes / web.php, ввівши URL-адресу визначеної сторінки можемо переглянути ті чи інші дані.

Наприклад, можемо отримати доступ до наступного посилання, перейшовши до http://your-app.dev/user у своєму веб-браузері. Це безкоштовний веб-фреймворк організований на базі поширеного патерну «модель-вид-контролер», який ми можемо побачити на рис. 2.2.

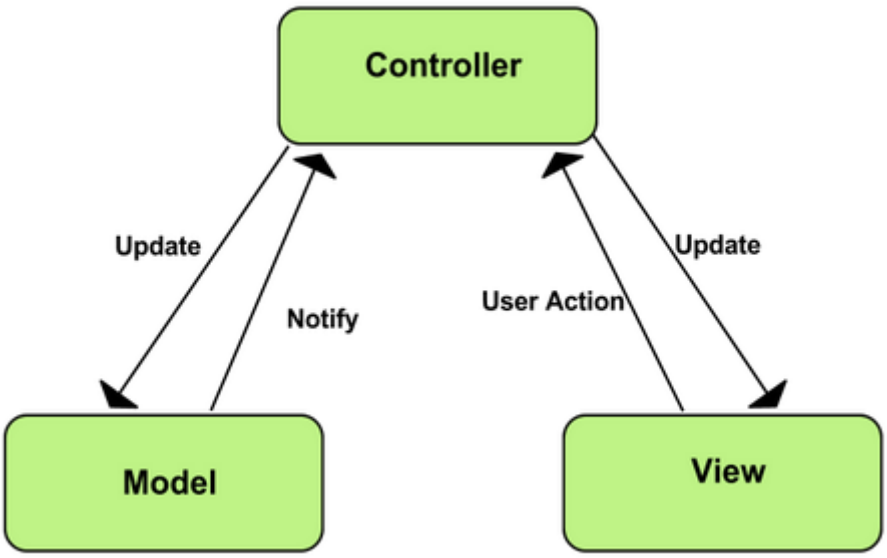


Рисунок 2.2 – Диаграма «модель-вид-контролер»

Модель це сутність збору даних та правил, яка використовується для обробки даних, які в свою чергу керують веб-додатком. Наприклад, модель може мати записи про реєстрації клієнта, записи на пристрої, які в свою чергу, обробляється певним образом. Як для нашої теми, також, це може бути ваш список контактів, адресів, телефонних номерів, електронної почти, та багато іншого.

Контроллер це в сою чергу управління HTTP запитами користувачів, коли він вибирає елемент інтерфейсу для виповнення різних дій. Кожна зовнішня дія представляється як унікальна подія в черзі. Його задача полягає в координації ресурсів та об’єктів, які потрібні для виконання забаганок користувачів. Як правило контроллер дає можливість визвати модель яка потрібна і передати дані для виду який підходе.

Вид - це візуалізація даних моделі. Він не спілкується з базаю данних. Види в загальному порядку розбивають на шаблон, який заповнюється даними. Може бути багато видів, але контроллер вибирає той вид, який найбільше підходе для даної ситуації. Це та частина, що користувач використовує в своїх цілях. Він відображає картинки, музику, відео, інформацію та інше.

Розберемо це на прикладі. Користувач може подивитися головну сторінку, пройти реєстрацію, записати свої дані та видалити їх. Припустимо, що користувач хоче подивитися свої записи. В свою чергу, він натисне на певний розділ, для того щоб відкрити інформацію. Контроллер отримає запит користувача, перевіре його а потім визве модель, запросив у неї дані які підходять до даної тематики. Використавши ці дані він верне інформацію до того шаблону який відповідає даній категорії.

По будуємо діаграми нашого прототипу IDEF0 для кожного значущого прецеденту, у якому потрібно детально розібратися як розробникам, так і замовникам.

На наступному рисунку 2.3 зображено IDEF0 діаграму для основного прецеденту "Облік подачі заявок".

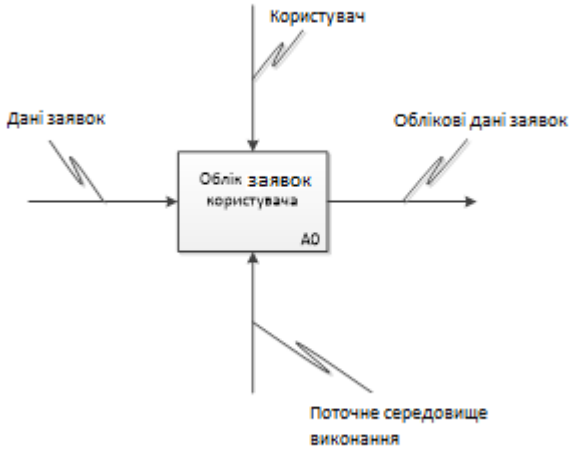


Рисунок 2.3 – Нульовий рівень декомпозиції процесу " Облік подачі заявок "

На даному рівні весь процес обліку та аналізу персональних заявок зображено у вигляді одного функціонального блоку. Вхідними даними для цього блоку є дані реєстрації користувача; зовнішньою системою контролю –

сам користувач; технологічною платформою – поточне середовище виконання; вихідними даними – облікові дані доходів.

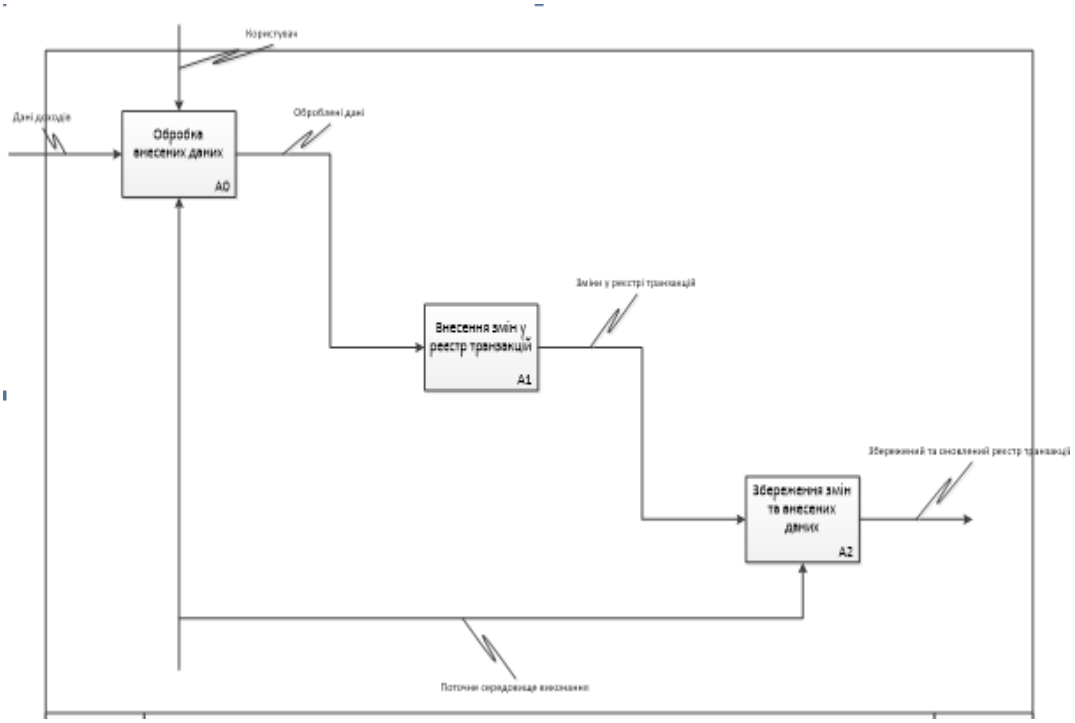


Рисунок 2.4 – Перший рівень декомпозиції процесу " Облік подачі заявок користувача"

На першому етапі здійснюється обробка введених даних. Дані перевіряються на відповідність можливим значенням та конвертуються у програмні об'єкти. На наступному етапі, оброблені дані вносяться у реєстр транзакцій. Далі, всі зміни у реєстрі транзакцій зберігаються для подальшого використання у системі. Перший та останній етапи виконуються з урахуванням поточного середовища виконання.

На рисунку 2.5 зображено другий рівень декомпозиції процесу "Збереження змін та внесених даних".

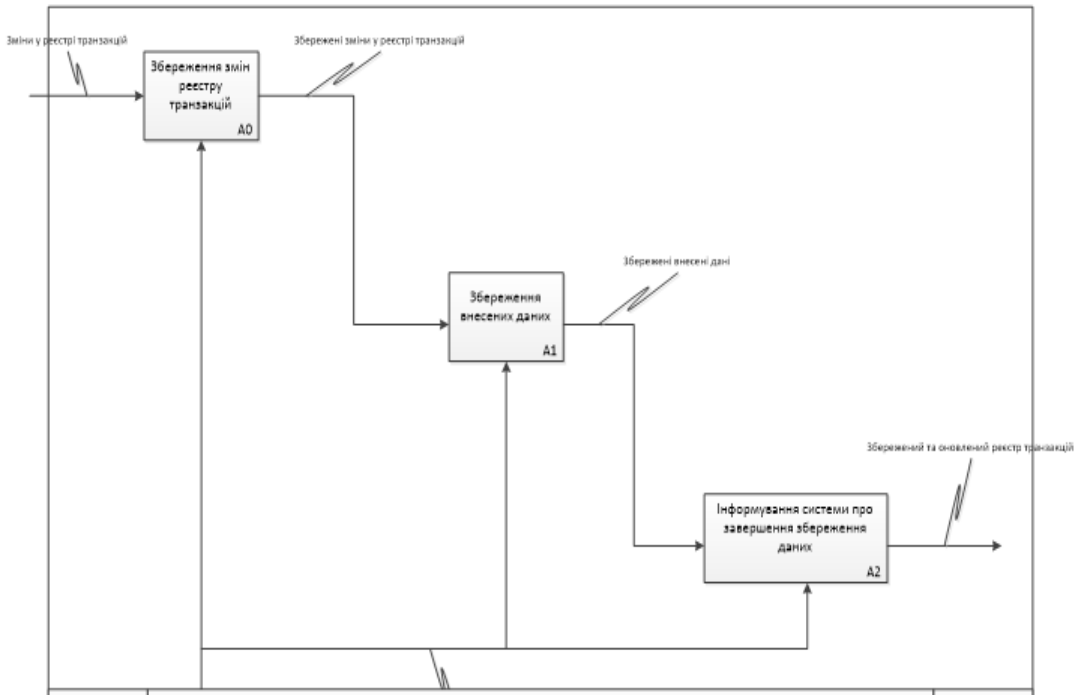


Рисунок 2.5 – Другий рівень декомпозиції процесу "Облік доходів користувача". Дія " Збереження змін та внесених даних "

Для розробки веб-додатку будемо вікористовувати операційну систему Linux Ubuntu. Це операційна система заснована Debian GNU/Linux. Дана система призначена для веб-серверів. На даній системі оточення є більш схожим обо таким же перед вигрузкой проекту на сервер.

Щоб почати розробку веб-орієнтованого прототипу потрібно встановити таке програмне забезпечення: Apache HTTP Server, MySql, PHP, PHPMyAdmin, PhpStorm, Composer.

В якості БД веб-програми ми будемо використовувати MySql. Це вільна та швидка реляційна база даних з відкритим кодом була створена як альтернатива комерційним системам. Дана база даних найкраще підходить до веб-додатків. Адже швидкість обробки даних в о обсязі до 500000 записів є найкращою. В неї відкрита та вільна та безкоштовна ліцензія, а також найбільше підходить до більшості хостингових компаній в Україні. ЇЇ можна використовувати на різних платформах така як Unix, Windows, інші. Кілька процесів можуть одночасно без жодних проблем читати дані з однієї бази.

MySQL має подвійне ліцензування, може розповсюджуватися відповідно до умов ліцензії GPL. Однак за умовами GPL, якщо яка-небудь програма включає вихідні коди MySQL, то вона теж повинна розповсюджуватися за ліцензією GPL. Це може розходитися з планами розробників, не бажаючих відкривати вихідні тексти своїх програм. Для таких випадків передбачена комерційна ліцензія, яка також забезпечує якісну сервісну підтримку.

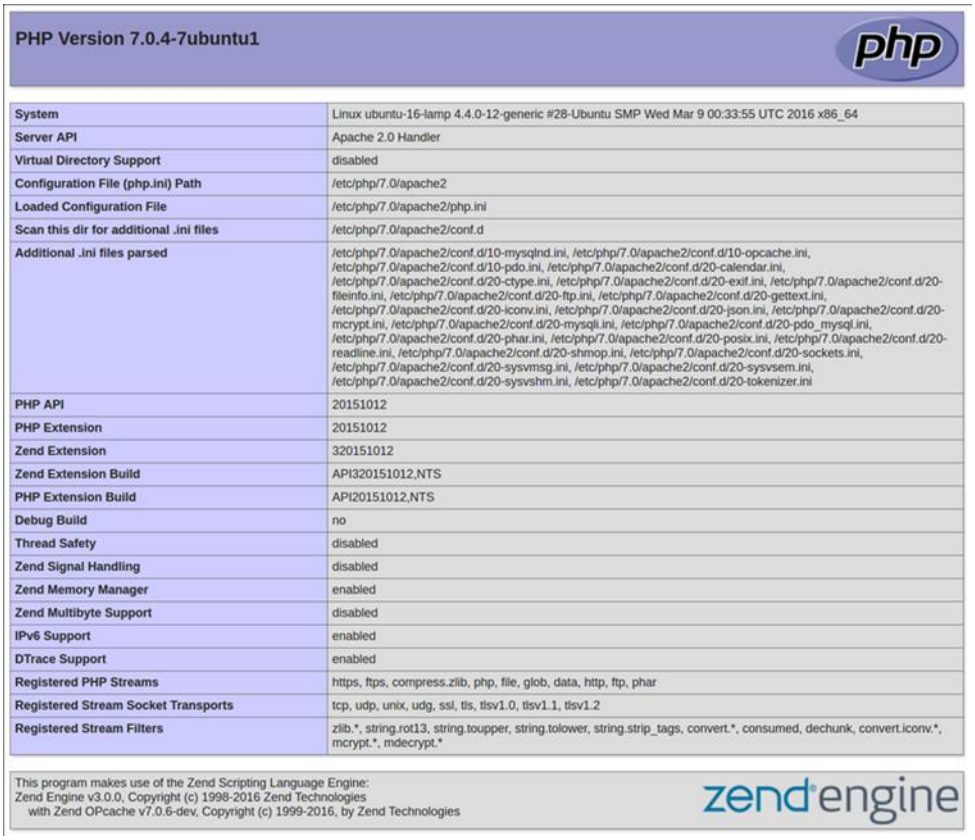
Apache HTTP Server - проект який розвивається The Apache Software Foundation, в результаті якого розробляється кроссплатформенний HTTP сервер з откритим вихідним кодом. Будучи лідером найбільших вагомих на ринку розробників Open Source, наштовхнула до прихильності компанії Apache Software Foundation, для того щоб створити постійне високоякісне програмне забезпечення, яке поліпшує майбутнє відкритої розробки. Apache піддержує половину веб-серверів в інтернеті, управляючи великою кількістю даних, виконує до 1000 мільярдів операцій в секунду, а також зберігає об’єкти в кожній галузі. Програмні проекти Apache є невід'ємною частиною майже будь-якого обчислювального пристрою кінцевого користувача, від ноутбуків до планшетів до телефонів. Код Apache для проектів складається більше ніж 6000 добровольців та співробітників корпорацій на шести континентах. Входить також в комплекс серверного програмного забезпечення як LAMP. Apache можна реалізувати чотирма різними варіантами:

- стандартні налаштування за замовчуванням в папці /var/www/html. Доступ буде здійснений по адресу http://localhost/;
- налаштування основного хостингу, прикладом буде http://localhost/phpmyadmin;
- за допомогою віртуальних хостів в папці, приклад http://mysite/
- модуль userdir в папці користувача public_html, приклад http://localhost/~username;

PHP це інтерпретатор мови який обробляє код, для виводу контенту. Він в загалом, може використовувати скріпти, підключатися до нашої бази

даних MySQL, для того щоб ми змогли відобразити і передасть контент в наш веб-сервіс. Потрібно встановити допоміжні менеджери пакетів, для того щоб PHP міг працювати з нашим сервером Apache і для того щоб MySQL без проблем спілкувалося.

Щоб Apache краще шукав файли PHP при зприті директорії, а не фали HTML (на даний час в першу чергу він буде шукати файл з назвою index.html), потрібно зробити наступне, в папці /etc/apache2/mods-enabled/ відкрити файл dir.conf і замінити містами index.html з index.php. Після того як ми це зробимо, потрібно зберегти файл, та перезапустити сервер Apache. Щоб перевірити, що наша система PHP працює та конфігурується, зробимо простий скріпт. Назвемо його info.php, та збережемо його в папці /var/www/html. Для перевірки потрібно відкрити лише дану сторінку з веб-браузера. Сторінку яку ми побачимо приведена на рис 2.6 . Ця сторінка несе в собі інформацію про наш сервер з точки PHP. Вона є важливою для отладки та коректності прийнятих налаштувань



PHP Version 7.0.4-7ubuntu1

System	Linux ubuntu-16-lamp 4.4.0-12-generic #28-Ubuntu SMP Wed Mar 9 00:33:55 UTC 2016 x86_64
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php/7.0/apache2
Loaded Configuration File	/etc/php/7.0/apache2/php.ini
Scan this dir for additional .ini files	/etc/php/7.0/apache2/conf.d
Additional .ini files parsed	/etc/php/7.0/apache2/conf.d/10-mysqlnd.ini, /etc/php/7.0/apache2/conf.d/10-opcache.ini, /etc/php/7.0/apache2/conf.d/10-pdo.ini, /etc/php/7.0/apache2/conf.d/20-calendar.ini, /etc/php/7.0/apache2/conf.d/20-ctype.ini, /etc/php/7.0/apache2/conf.d/20-exif.ini, /etc/php/7.0/apache2/conf.d/20-fileinfo.ini, /etc/php/7.0/apache2/conf.d/20-ftp.ini, /etc/php/7.0/apache2/conf.d/20-gettext.ini, /etc/php/7.0/apache2/conf.d/20-iconv.ini, /etc/php/7.0/apache2/conf.d/20-json.ini, /etc/php/7.0/apache2/conf.d/20-mcrypt.ini, /etc/php/7.0/apache2/conf.d/20-mysqli.ini, /etc/php/7.0/apache2/conf.d/20-pdo_mysql.ini, /etc/php/7.0/apache2/conf.d/20-phar.ini, /etc/php/7.0/apache2/conf.d/20-posix.ini, /etc/php/7.0/apache2/conf.d/20-readline.ini, /etc/php/7.0/apache2/conf.d/20-shmop.ini, /etc/php/7.0/apache2/conf.d/20-sockets.ini, /etc/php/7.0/apache2/conf.d/20-sysvmsg.ini, /etc/php/7.0/apache2/conf.d/20-sysvsem.ini, /etc/php/7.0/apache2/conf.d/20-sysvshm.ini, /etc/php/7.0/apache2/conf.d/20-tokenizer.ini
PHP API	20151012
PHP Extension	20151012
Zend Extension	320151012
Zend Extension Build	API320151012,NTS
PHP Extension Build	API20151012,NTS
Debug Build	no
Thread Safety	disabled
Zend Signal Handling	disabled
Zend Memory Manager	enabled
Zend Multibyte Support	disabled
IPv6 Support	enabled
DTrace Support	enabled
Registered PHP Streams	https, ftps, compress.zlib, php, file, glob, data, http, ftp, phar
Registered Stream Socket Transports	tcp, udp, unix, udg, ssl, tls, tlsv1.0, tlsv1.1, tlsv1.2
Registered Stream Filters	zlib.*, string.rot13, string.toupper, string.tolower, string.strip_tags, convert.*, consumed, dechunk, convert.iconv.*, mcrypt.*, mdecrypt.*

This program makes use of the Zend Scripting Language Engine:
Zend Engine v3.0.0, Copyright (c) 1998-2016 Zend Technologies
with Zend OPcache v7.0.6-dev, Copyright (c) 1999-2016, by Zend Technologies

Рисунок 2.6 – PHP інформація

PhpMyAdmin - це безкоштовний програмний інструмент, написаний на PHP, призначений для адміністрування MySQL через Інтернет. phpMyAdmin підтримує широкий спектр операцій на MySQL та MariaDB. Часто використовувані операції (керування базами даних, таблицями, стовпцями, відносинами, індексами, користувачами, дозволами тощо) можна виконувати за допомогою інтерфейсу користувача, тоді як ви все ще маєте можливість безпосередньо виконувати будь-яке SQL-команду.

phpMyAdmin пропонує широкий діапазон документації, і користувачі можуть оновлювати сторінки на вікіпедії, щоб поділитися ідеями та розробкою різних операцій. Команда phpMyAdmin намагатиметься допомогти, якщо ми зіткнетеся з будь-якою проблемою. Ми можете скористатися різними каналами підтримки, щоб отримати допомогу. Також phpMyAdmin дуже глибоко задокументована в книзі, написаній одним із розробників - Mastering phpMyAdmin для ефективного управління MySQL, який доступний англійською та іспанською мовами. Щоб полегшити використання широкого кола людей, phpMyAdmin транслюється на 72 мови та підтримує як LTR, так і RTL мови.

phpMyAdmin - це зрілий проект зі стабільною та гнучкою кодовою базою. Ми можете дізнатись більше про проект та його історію та нагороди, які вона заробила.

Проект phpMyAdmin є членом програми Conservancy Software Freedom. SFC є неприбутковою організацією, яка допомагає просувати, вдосконалювати, розвивати та захищати проекти Free, Libre та Open Source Software (FLOSS).

Головні переваги даної системи полягають в тому, що вона просто в використанні і в більшості випадків дозволяє не використовувати команди SQL, тому робота з базой даних э більш простою і підходе для людей які поверхнево знають MySql.Так же дана система має велику популярність у веб- програмуванні та є актуальною на сьогоднішній день. PhpMyAdmin може керувати цілим сервером MySQL (потребує супер-користувача), а

також єдину базу даних. Для виконання останньої вам буде потрібно належним чином налаштований користувач MySQL, який може читати або записувати лише бажану базу даних. Багато людей ускладнюють розуміння поняття керування користувачами щодо phpMyAdmin. Коли користувач входить до phpMyAdmin, це ім'я користувача та пароль передаються безпосередньо в MySQL. phpMyAdmin не здійснює керування обліковими записами самостійно (крім того, що дозволяють маніпулювати даними про обліковий запис користувача MySQL). Всі користувачі повинні бути дійсними користувачами MySQL. Оскільки інтерфейс phpMyAdmin повністю заснований у нашому браузері, для встановлення файлів phpMyAdmin нам потрібен веб-сервер Apache.

Команда розробника phpMyAdmin робить багато зусиль, щоб зробити phpMyAdmin максимально безпечним. Проте веб-додатки, такі як phpMyAdmin, можуть бути вразливими до ряду атак, і все ще вивчаються нові способи експлуатації. Коли phpMyAdmin показує фрагмент даних користувача, наприклад, щось всередині бази даних користувача, всі спеціальні символи html повинні бути вичерпані. Коли ці зникнення відсутні, шкідливий користувач може заповнити базу даних спеціально зібраним вмістом, щоб змусити іншого користувача цієї бази даних виконувати щось. Це, наприклад, може бути фрагментом коду JavaScript, який би робив будь-яку кількість неприємних речей.

PhpStorm - інтегрована середа розробки для PHP. Це інтелектуальна середа розробки з аналізатором кода, запобігання помилок в коді і автоматизованими засобами рефакторинга для PHP і JavaScript. PhpStorm — розроблений на платформі IntelliJ IDEA, написана на мові програмування Java. Розробники можуть додати функціонал середи розробки за допомогою встановлення плагінів, розроблених спеціально для платформи IntelliJ, або можна написати свої плагіни.

В ньому спостерігається спеціальна та зручна підсвітка коду. PhpStorm ідеально підходить для роботи з Symfony, Drupal, WordPress, Zend

Framework, Laravel, Magento, Joomla !, CakePHP, Yii та іншими фреймворками. Редактор фактично "отримує" ваш код і глибоко розуміє свою структуру, підтримуючи всі можливості мови PHP для сучасних та застарілих проектів. Вона забезпечує найкраще завершення коду, рефакціонування, попередження помилок на скріптах тощо.Сотні перевірок піклуються про перевірку даного коду під час введення, аналізуючи весь проект. Підтримка PHPDoc, кодовий аранжувальник і форматер, швидкі виправлення та інші функції допоможуть вам написати акуратний код, який легко підтримувати. Рефікціонування свого коду надійно з безпечним перейменування, переміщення, видалення, метод витяжки, вбудована змінна, натискання елементів вгору або витягування елементів вниз, змінити підпис та багато інших рефакторингів. Реакціонування за специфікою мови допоможе вам виконати загальні зміни впродовж декількох кліків, і їх можна безпечно скасувати.

PhpStorm відома своєю нульовою конфігурацією Visual Debugger, що забезпечує надзвичайне розуміння того, що відбувається у вашій програмі на кожному кроці. Він працює з Xdebug і Zend Debugger, і може використовуватися як локально, так і віддалено. Також доступні тестування модулів з PHPUnit, BDD з Behat та інтеграцією профілів.

Composer — це менеджер залежностей для PHP. є менеджером пакетів в тому ж самому значенні, що і Yum або Apt. Так, він стосується "пакунків" або бібліотек, але керує ними за принципом проекту, встановлюючи їх у каталозі (наприклад, постачальника) у вашому проекті. За замовчуванням він нічого не встановлює глобально. Таким чином, це менеджер залежності. Тим не менше, він підтримує "глобальний" проект для зручності через команду.

2.2 Склад функціональної частини

Важка система яка може складатися з елементів та забезпечувати організацію між ними є функціональною частиною. Автоматизовану інформаційну систему (АІС) розрізняють на функціональну частину та частину забезпечення, які дальше поділяються на менші елементи підсистеми.

Рисунок 2.7 – Автоматизована інформаційна система

Частина яка відповідає за фукцію АІС є переважною. Пов’язана вона з сервером та е обьєктом управління контентом . До неї належать:

1. призначення;
2. функції які виконуються в управліні;
3. функції які оброблять інформацію.

Елементи які полягають в основу функціональної частини АІС є: блоки, елементи, комплекси задачі, окремі задачі.

Функціональна підсистеми – це частинно незалежна частина системи, за функціональних ознак управління. В АІС органів казначейства функціональні підсистеми, як правило, виокремлюють за такими ознаками:

- прогнозування, планування, облік, звітність, тощо;
- видами основної діяльності;
- організаційною структурою.

Функціональна структура АІС має дивитися на ті проблеми інформації кінцевих користувачів, які змінюються в умовах ринку, та відтворювати зміст зміст, а також функцій ті що керують конкретним економічним об'єктом. В АІС має бути гнучка структура і бути відкритою системою, але вона повина відокремити ті змін у нашу модель та зробити нарощування функціональних частин як потребуєтьсся.

Дана задача виповнюється за поняттям модульності АІС. Кожний прикладний модуль системи віводить інформаційну плітику. Головною задачею при розробці дпаного модулів має орієнтуватися за системою на автоматизовану діяльність об'єкта, а не на впровадження локальних функціональних задач. При цій функції, що робляться, та модулі маютьдивитися з точки потреб наших користувачів, а не програмної реалізації. Комплексність нашої функціональної системи забезпечується для інтеграції модулів в одну систему.

Розробка функціональної частини слід проводити згідно з вимогами висунутими у даних формування задач до системи яка автоматизується до вимог розробки субсидій та пільгових надбавок. Проведемо виділення функцій та виділення функціональних модулів:

1. Управління користувачами:
   1.1. Реєстрація нових користувачів;
   1.2. Видалення користувачів.
2. Управління записами:
   2.1. Створення записів про субсидії;
   2.2. Створення записів про даних користувачів;

2.3. Розрахунок субсидії по калькулятору;

2.4. Запис даних по калькулятору.

3. Сервіс інформування користувачів:

3.1. Інформування про незаповнені поля;

3.2. Документація з використання автоматизованої системи.

4. Сервіс обробки:

4.1. Створення звітності про реєстрацію;

4.2. Створення калькулятора

У той же час, як однорівневі, наприклад, так і в дворівневих системах децентралізованої обробки даних функціональними елементами автоматизованих систем є функціонально-спеціалізована ARM (FSAMM). Враховуючи вищезазначене, AWP слід розуміти як сукупність персональних комп'ютерних пристроїв, які за допомогою обчислювальної потужності великого комп'ютера дозволяють користувачеві виконувати служби інформації на робочому місці в обсязі та режимі, необхідних для виконання його функцій.

Усі функціонально-спеціалізовані АРМ мають базову функціональну частину органів АІС. Їх кількість та структура в АІС на різних рівнях субсидії системи залежить від обраної технології обробки даних та розподілу функцій між виконавцями. Основними факторами, що впливають на формування функціонально-спеціалізованої АРМ, є:

- структура організації органів;

- різна дія органів;

- використання по діїї окремих групи працюючих та підрозділів по структурі органів субсидії, розділення функціональних повноважень між ними.

Автоматизація діяльності субсидіїї починається з найбільш трудомістких операцій, пов'язаних з реєстрацією осіб, обслуговування для отримання субсидіїї. Що залежить від особливостей його організаційної структури, ARM провіряє доходи і видає субсидію.

Потім автоматизувати обмін інформацією з даного питання про рух коштів між окремими рівнями казначейської системи та банківськими установами, а також іншими учасниками економічного процесу. Для забезпечення різних рівнів системи та організації їх взаємодії використовуються інформаційні зв'язки з банківськими установами: АРМ-1 - для вирішення функціональних проблем на центральному рівні; АРМ-2 - на рівні інформаційних центрів обласних управлінь НБУ; АРМ-СЕП - на рівні учасників системи електронних платежів.

Загалом, виконання функціональних завдань в АІС може розглядатися як послідовність дій з інформацією, що міститься в базі даних, а також операції вхідних і вихідних даних. Виклик функцій AIS здійснюється за допомогою раціонів. Для кожного користувача система забезпечує індивідуальне харчування тієї, в якій додаються лише необхідні функції, що виключає можливість доступу до функцій системи інших користувачів інформації.

Поліпшення та розвиток функціональної частини АІС відбувається у напрямку розробки та впровадження нових підсистем, засобів масової інформації, завдань. У той же час зміна функціональної структури пов'язана з зміною ринкового середовища, регулювання та ін. Покращення функціональних даних АІС забезпечує повне та ефективне виконання функцій, що виконуються автоматичним методом в а, а також підвищує функціональну придатність АІС, що в кінцевому підсумку відображає покращення управління фінансовими ресурсами держави.

2.3 Склад підсистем забезпечення функціональної частини

2.3.1 Програмне забезпечення

Програ́мне забезпе́чення (програ́мні за́соби) (ПЗ; англ. software) — сукупність програм системи обробки інформації і програмних документів, необхідних для експлуатації цих програм [13].

PHP є одним з найбільш широко використовуваних і впізнаваних технологій, що використовуються в Інтернеті. Спочатку PHP виступав за «Персональну домашню сторінку», хоча останнім часом він був змінений на «PHP: Hypertext Preprocessor». Однак незалежно від того, що воно називається, PHP є основною частиною будь-якої динамічної веб-сторінки.

Розвиток PHP почався в 1994 році як особистий проект Расмуса Лердорфа, який створив серію Perl-скриптів, які він назвав "Індивідуальними інструментами домашньої сторінки" для підтримки його особистої веб-сторінки. У 1995 році ці інструменти були упаковані та випущені як CGI-файли як "Персональна домашня сторінка / Форми перекладача", яка включала підтримку веб-форм та спілкування з базами даних.

Після того, як він був випущений у світ в цілому, PHP пройшов швидку розробку та розробку, а друга версія PHP / FI була випущена через два роки пізніше в листопаді 1997 року. PHP 3 був випущений в 1998 році з PHP 4 та PHP 5 в 2000 та 2004 роках, відповідно.

PHP 5 - це версія, яка наразі використовується на більшості веб-сайтів, і включає в себе кілька нових функцій, таких як підтримка об'єктно-орієнтованого програмування, послідовний інтерфейс для доступу до бази даних та кілька основних удосконалень.

Хоча PHP залишається в стадії розробки, поточний процес розробки, який в кінцевому підсумку призведе до PHP 6, був повільним, ніж передбачалося через труднощі додавання підтримки Unicode. У 2010 році було прийнято рішення про перенесення підтримки Unicode до філії, перемістивши всі інші функції, що розробляються, до головного корпусу PHP-коду. Однак у версії 5.4 PHP нарешті додав підтримку Unicode без зміни основної версії.

PHP випущений під ліцензією PHP, яка схожа на загальну публічну ліцензію GNU, за винятком того, що будь-яке похідне програмне забезпечення може не називатися "PHP" і не мати назви "PHP".

В даний час PHP має безліч застосувань, що робить його чудовим інструментом для вирішення будь-якої кількості проектів. Багато основних програмних продуктів, таких як WordPress та phpBB, використовують PHP для виконання завдань, таких як запуск блогу чи форум. PHP також має унікальні можливості, такі як можливість динамічно генерувати зображення в безлічі форматів і доступ до баз даних у багатьох різних форматах.

PHP також має можливість бути вбудованим безпосередньо в веб-сторінку або використовується з командного рядка, що робить його потужним інструментом, який може обробляти що-небудь від відображення інформації, витягнутої з бази даних, до виконання системних завдань запланованим способом.

Важливо пам'ятати про PHP, що це попередній процесор, а це означає, що будь-які скрипти PHP на веб-сторінці виконуються перед тим, як сторінка відображається. Це означає, що будь-які скрипти PHP на сторінці не можуть змінити цю сторінку після її відображення. Існує кілька способів подолання цього обмеження, наприклад AJAX (Asynchronous Java Script and XML), що дозволить вам змінювати те, що знаходиться на сторінці, не оновлюючи всю сторінку, але ці технології та методи не підпадають під дію цієї статті. .

Найпоширенішим використанням PHP є доступ до бази даних, аналіз результатів з цієї бази даних та відображення результатів на веб-сторінці. Ось чому PHP - остання частина загального абревіатуру "LAMP", що означає "Linux, Apache, MySQL і PHP". Встановлення LAMP - це одна з найпоширеніших конфігурацій веб-сервера і поєднує в собі потужний веб-сервер Apache з PHP та MySQL, що дозволяє створювати вражаючі надійні веб-сторінки та керувати даними. Фактично ці інструменти часто настроюються, щоб працювати разом з малою або без додаткової конфігурації.

Важливо відзначити, що кожен, хто регулярно використовує PHP, полягає в тому, що PHP найчастіше виконує з тими самими правами, що і веб-серверне програмне забезпечення. З точки зору безпеки важливо

пам'ятати, що якщо щось знаходиться у вашому веб-каталозі на вашому сервері, неправильно написаний скрипт PHP може мати доступ до нього.

На закінчення, PHP є потужною мовою, яка стала однією з рушійних сил Інтернету, і кожен, хто розглядає кар'єру в веб-розробці, повинен зробити навчання PHP першочерговим.

2.3.2 Апаратне забезпечення

Апаратне забезпечення (англ. hardware) — комплекс технічних засобів, який включає ЕОМ: зовнішні пристрої, термінали, абонентські пункти тощо, які необхідні для функціонування тієї чи іншої системи; фізична частина ЕОМ [15].

Згідно обраних технологій, архітектури, платформи, структури функціональної частини та програмного забезпечення до апаратного забезпечення висуваються такі вимоги:

- PHP>= 7.0.0;
- OpenSSL PHP Extension;
- PDO PHP Exstension;
- Mbstring PHP Exstension;
- Tokenizer PHO Exstension;
- XML PHP Exstension;

Якщо ми встановили PHP локально, і ми хочете використовувати вбудований сервер PHP для роботи з нашою програмою, можемо використовувати команду serve artisan.

Багато нових вбудованих системних плат з високопродуктивними обчислювальними можливостями стали нормою в світі Інтернету (кожного) речей. Незалежно від того, чи використовуєте ви Малину Пі, Beaglebone чи багатьох інших, незвичайно знайти вбудоване в мережу устаткування, здатне підтримувати функціональність обласного або простого сервера.

Багато хто з цих плат, як можна було б очікувати, можуть інтерфейсувати через цифрові або аналогові штифти до датчиків і виконавчих механізмів. Але вони також мають роз'єми RJ45, з'єднані з контролерами інтернет. І їх операційна система здатна розміщувати серверне програмне забезпечення (наприклад, Apache), яке включає PHP.

Метод Extract витягує потрібний набір точок і обробляє повний цикл доступу через виконуваний файл. Фактично, виконуваний файл обробляє як інтерфейс сенсора, так і деталі переносу вихідних даних. У свою чергу, клас Sensor може використовуватися для виведення даних відформатованого датчика на мережевий запит і, необов'язково, для виконання додаткової обробки.

Одна з великих переваг програмного забезпечення Open Source полягає в тому, що вона забезпечує можливість адаптації до нових середовищ. Це стосується PHP. Незважаючи на те, що спочатку він був призначений як модуль для веб-сервера Apache, PHP з тих пір прийняв стандарт ISAPI, що дозволяє йому працювати однаково добре з інформаційним сервером Microsoft. Що стосується вимог до апаратних засобів, ми особисто бачили, як працює PHP на 100-МГц Pentium-машинах під управлінням Slackware Linux та Windows NT відповідно. Виконання було добре для використання в якості особистого середовища розробки. Звичайно, сайт, який очікував отримати тисячі запитів на день, потребує швидшого обладнання. Незважаючи на те, що для порівняння PHP-сайту з плоским HTML-сайтом потрібні додаткові ресурси, вимоги не різко відрізняються. Незважаючи на мій приклад, ви не обмежуєтеся обладнанням Intel. PHP працює однаково добре на процесорах PowerPC та Sparc.

Вибравши операційну систему, моємо загальний вибір між Windows і UNIX-подібною операційною системою. PHP буде працювати на Windows 95 і 98, хоча ці операційні системи не підходять для великих трафічних веб-серверів. Він також буде працювати в Windows NT та його наступнику Windows 2000. Для операційних систем UNIX, PHP добре працює з Linux і

Solaris, а також іншими. Якщо ви вибрали систему на основі PPC, наприклад Macintosh, ви можете вибрати LinuxPPC, версію Linux. Ви можете переслідувати комерційний WebTen веб-сервер, який працює в ОС Macintosh. Чад Каннінгем запропонував патчі для складання PHP в Apple OS X. У 1999 році Брайан Хавард додав підтримку IBM OS / 2.

PHP все ще найкраще працює з веб-сервером Apache. Але тепер він дуже добре працює з nginx. Він також складається як модуль для веб-сервера fhttpd. Ви можете виконувати роботу PHP практично з будь-яким веб-сервером за допомогою версії CGI, але я не рекомендую цю настройку для веб-сайтів виробництва. Якщо ви використовуєте UNIX, ми рекомендуємо скомпілювати PHP як модуль Apache. Якщо ви використовуєте Windows NT, перейдіть до IIS.

Якщо використовувати Linux, ми можете легко знайти RPM для Apache та PHP, але ця установка може не включати будь-яку бажану функцію PHP. Я рекомендую цей маршрут як дуже швидкий старт. Ви завжди можете перейти до складання Apache та PHP з нуля пізніше. PHP буде компілюватися на більшості версій операційних систем UNIX, включаючи Solaris і Linux. Сценарії PHP - це лише текстові файли, і можем редагувати і створювати їх, як і HTML-файли. Звичайно, ми можете по ssh потрапити на свій веб-сервер і почати створювати файли з vi. Або ми можете створювати файли з блокнотом і використовувати ftp для їх завантаження по черзі. Але це не ідеальні переживання. Одна зручна функція нових редакторів - це вбудований FTP. Ці редактори можуть відкривати файли на віддаленому веб-сервері як на локальному диску. Один натиск зберігає їх на віддаленому веб-сервері. Інша особливість, - виділення синтаксису. Це призводить до того, що ключові слова PHP будуть забарвлені, щоб допомогти вам швидше прочитати код.

Отже, як бачимо для роботи розроблюваного веб-додатку нам потрібні найкрвщі апаратні потужності, що дозволяє використовувати його на багатьох хостингах.

2.3.3 Інші види забезпечення

Інші види до забезпечення відноситься організаційне забезпечення, правове, лінгвістичне та ергономічне.

Організаційну підтримку слід розуміти як узгодження місця, часу та мети спільної роботи окремих виконавців, команд та технічних засобів. Вона повинна бути запроваджена та керована певними правилами взаємодії, які формують правовий та моральний кодекс і формують правову базу. Те, що організаційна підтримка, базується на нормативно-правових актах правової охорони, і юридичне положення втілюється в організаційне забезпечення.

Організаційна підтримка інформаційної системи включає набір інструментів, методів та відповідного персоналу. Вона повинна забезпечити:

- проведення техніко-економічного обґрунтування існуючої системи управління, відбору та встановлення завдань побудови інформаційної системи на етапі розробки та реалізації;

- регулювання взаємодії персоналу з ансамблем технічними засобами та між собою в процесі вирішення контрольних завдань, моніторингу ефективності системи управління на етапі функціонування інформаційної системи.

На етапі проектування організаційна підтримка виконує наступні завдання:

- аналіз існуючих систем управління та формулювання напрямів підвищення їх ефективності;

- вибір та виклад завдань управління;

- формулювання вимог до ансамблю технічних засобів;

- розробка організаційних рішень щодо складу, структури, організації та методології вирішення проблем управління в інформаційній системі, складу робочих процедур та пояснення їх реалізації.

На етапі функціонування інформаційної системи організаційна підтримка вирішує такі завдання:

- впровадження методів управлінських завдань;

- організація персоналу та ансамблю технічних засобів інформаційної системи;

- контроль та аналіз ефективності управління;

- формування пропозицій щодо вдосконалення та розвитку інформаційної системи.

Основою ефективного розвитку підприємств є регуляторна база, що базується на сукупності законів, необхідних для регулювання та забезпечення. Юридична підтримка - це сукупність загальновизнаних заходів, виражених у нормативних актах, які встановлюють та встановлюють організацію інформаційної системи, її мету, завдання, структуру та функції (правовий статус інформаційної системи та її підрозділів), призначені для регулювання створення та функціонування інформаційної системи. Юридична безпека базується на правовому підході, який розглядає соціальне управління як організацію в чомусь, включаючи виконавчу та адміністративну діяльність державних органів, спрямовані на реалізацію законів та інших нормативних актів, прийнятих владою та керівництвом. Юридичний підхід аналізує місце та роль закону в управлінні, визначає зміст законної виконавчої та регуляторної діяльності органів державної влади та органів управління, робить рекомендації щодо його вдосконалення.

Функції структурно-правового управління виконуються у формі нормативно-правових актів, планів, регламентів і способів, обов'язкові для всіх гостей, які визначають різні матеріали у сферах планування та управління. Нормативно-правові акти - це конфігурація виразу та встановлення правових загально визнаних заходів, сукупність яких формує правове Основи управління, яке є ієрархічним. Нормативно-правові акти поділяються на законодавчі, нормативні акти міністерств та відомств, акти місцевих органів влади та управління, місцеві акти, окремі акти.

На етапі функціонування інформаційної системи юридична підтримка включає:

- статус інформаційної системи у конкретних сферах державного управління;

- правова позиція на тему компетенції зв'язків інформаційної системи та організації їх діяльності;

- права, обов'язки та відповідальність персоналу інформаційної системи;

- правовий статус певних видів управлінських процесів в інформаційній системі;

- порядок отримання та використання інформації в інформаційній системі, процедури його збору, реєстрації, зберігання, передачі та обробки;

- порядок отримання та використання ансамблю технічних засобів, програмного забезпечення, інформації та інших видів підтримки.

Лінгвістичне забезпечення охоплює сукупність науково-технічних термінів ти інших мовних засобів, правил формалізації мов, методів стиску запису інформації, засобів діалогу людини і обчислювальної системи. Лінгвістичне забезпечення включає в себе:

- інформаційні мови для опису структурних одиниць баз даних інформаційної системи (документів, показників, реквізитів);

- мови управління, маніпулювання і обміну даними в банку даних інформаційної системи;

- мовні засоби інформаційно-пошукових систем;

- мовні засоби системи автоматизованого проектування;

- діалогові мови;

- словники термінів і визначень.

Ергономічне забезпечення охоплює сукупність методів і засобів, призначених для створення оптимальних умов для ефективної діяльності і навчання операторів з складу персоналу інформаційної системи. Ергономічне забезпечення включає в себе:

- комплекс документації, яка містить ергономічні вимоги до робочих місць і здійснює експертизу робочих місць;

- комплекс методів, учбово-методичних матеріалів і технічних засобів підготовки персоналу до роботи;

- комплекс методів і засобів, які забезпечують професійний відбір.

В плані ергономічного забезпечення на етапах проектування інформаційної системи визначається ступінь і рівень участі людини в системі управління, вимоги до форми представлення інформації, умови оточуючого середовища діяльності людини, порядок роботи і відпочинку персоналу, нормативи навантаження і надійності персоналу; вимоги до технічних засобів, способи взаємодії персоналу і технічних засобів.

В реальних інформаційних системах загальне число видів забезпечення, що підтримують достатнє та повне функціонування організації може бути і меншим. Обов’язковими складовими інформаційної системи є лише підсистеми програмного та апаратного забезпечення. Функції інших видів забезпечення менш значимі і можуть об’єднуватися і групуватися або входити в основні підсистеми.

РОЗДІЛ 3 РЕАЛІЗАЦІЯ ПРОТОТИПУ ВЕБ-ОРІЄНТОВАНОЇ АВТОМАТИЗОВАНОЇ СИСТЕМИ «ПОДАННЯ ЗАЯВОК НА ОТРИМАННЯ СУБСИДІЙ І ПІЛЬГОВИХ НАДБАВОК»

3.1 Особливості розробки серверної частини

У ході розробки прототипу автоматизованої системи потрібно встановити стек LAMP. Це програмне забезпечення з відкритим вхідним кодом для відображення веб-сайтів та веб-додатків. Абревіатура розкриває операційну систему Linux з встановленим веб-сервером Apache. Сховище данних є MySQL, а також PHP, який обробляє динамічний контент.

Першим кроком буде встановлення веб-сервера Apache. Без проблем можна встановити його використовуючи менеджер пакетів Ubuntu apt. Для того щоб почати встановлення виконаєм команди, які приведені на рисунку 3.1.

```
$ sudo apt-get update
$ sudo apt-get install apache2
```

Рисунок 3.1 – Встановлення Apache

Після запуска цих команд установник повідомить, що буде встановлено і скільки місця на жорсткому диску буде зайнято. Нажавши Y продовжимо встановлювати .

Другим кроком буде встановлення MySQL. Також, за допомогою apt для загрузки та встановлення програмного забезпечення використаємо команду на рисунку 3.2.

```
$ sudo apt-get install mysql-server
```

Рисунок 3.2 – Встановлення MySQL

В ході встановлення сервер сробить запит для встановлення та підтвердження пораля для користувача root. Це аккаунт адміністратора, який має розширені права.

Крок три – це встановлення PHP. В цьому пункті ми встановемо, деякі допоміжні пакети, щоб PHP міг працювати з сервером Apache, а також спілкуватися з MySQL.

```
$ sudo apt-get install php libapache2-mod-php php-mcrypt php-mysql
```

Рисунок 3.3 – Встановлення PHP та допоміжні пакети

Встановимо phpMyAdmin безкоштовний веб-додаток для роботи з MySQL. Зробимо це через apt-get.

```
sudo apt-get install phpmyadmin apache2-utils
```

Рисунок 3.4 – Встановлення phpMyAdmin

Під час встановлення ми пройдемо базову конфігурацію:

- спочатку потрібно вибрати Apache2 для сервера;
- потім нажмемо YES, коли буде питання, потрібно налаштувати базу данних для phpMyAdmin за допомогою dbconfig-common;
- введемо пароль MySQL;
- пароль для phpMyAdmin.

Після завершення встановлення, добавимо phpMyAdmin в конфігурацію Apache2

```
Include /etc/phpmyadmin/apache.conf
```

Рисунок 3.5 – конфігурацію Apache2

Наступним кроком встановемо Composer:

```
$ mv composer.phar /usr/local/bin/composer
$ chmod +x /usr/local/bin/composer
```

Рисунок 3.6 – Встановлення Composer

Далі ми встановимо Laravel. Він використовує Composer для управління своїми залежностями.Зробимо, выпустивши команду create-project Composer в своєму терміналі.

```
composer create-project laravel/laravel {directory} "5.0.*" --prefer-dist
```

Рисунок 3.7 – Встановлення Composer

Після встановлення Laravel скористаемся перевагами phpMyAdmin, та створимо нашу базу данних. Приклад створення рисунок 3.8.

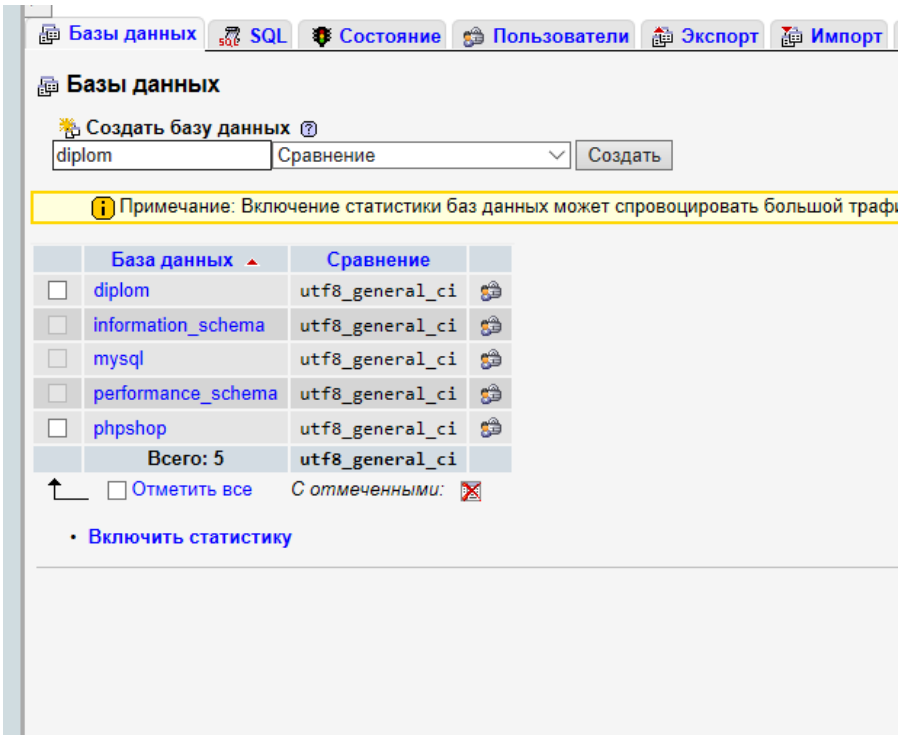


Рисунок 3.8 –Створення бази данних

Для створення таблиць в нашу базу данних скористаємся Eloquent ORM, який включений в Laravel та забезпечує прекрасну, просту реалізацію ActiveRecord для роботи з нашою базою даних. Створемо таблицю для користувачів нашого прототипу електронної реєстрації субсидії та пільгових надбавок використавши лістинг 3.1 .

Лістинг 3.1 – Приклад створення таблиці users

```
public function up()
{
    Schema::create('users', function (Blueprint $table) {
        $table->increments('id');
        $table->string('name');
        $table->string('email')->unique();
        $table->string('password');
        $table->rememberToken();
        $table->timestamps();
    });
}
```

Для кращого розуміння таблиці БД опишемо їх детальніше. Таблиця користувачів містить дані щодо імя людини та адреси елетронної почти. Її детальна структура представлена у таблиці 3.1.

Таблиця 3.1 – Структура та опис таблиці користувачів додатку

Назва атрибуту	Тип даних	Короткий опис
_ID	Integer	Первинний ключ, унікальний ідентифікатор рядків
name	String	Ім’я користувача
email	String	Емейл користувача
password	String	Пароль
remember_token	String	Токен
created_at	Timestamps	Дата створення
updated_at	Timestamps	Дата оновлення

Таблиця профіля забезпечую перхід до кабінета користувача, за його ідентифікатором.

Лістинг 3.2 – Приклад створення таблиці profiles

```
public function up()
{
    Schema::create('profiles', function (Blueprint $table) {
        $table->increments('id');
        $table->integer('user_id');
        $table->timestamps();
    });
}
```

Дана таблиця містить ідентифікатор користувача, та відомості про створення і оновлення даної таблиці. Зробимо опис.

Таблиця 3.2 – Структура та опис таблиці профіль прототипу

Назва атрибуту	Тип даних	Короткий опис
_ID	Integer	Первинний ключ, унікальний ідентифікатор рядків

user_id	Intager	Ідентифікатор користувача
created_at	Timestamps	Дата створення
updated_at	Timestamps	Дата оновлення

Великими перевагами структури бази даних є відсутність надмірності даних. Для цього потрібно розподілити інформацію з кількох окремих тематично організованим таблиць, щоб кожен факт був представлений один раз. Таблиця профілів та таблиця користувачів має зв'язок один до одного.

Лістинг 3.3 – Приклад зв’язку таблиці користувачів до профіля

```
public function profile()
{
    return $this->hasOne(Profile::class, 'user_id', 'id');
}
```

Лістинг 3.4 – Приклад зв’язку таблиці профіля до користувачів

```
public function user()
{
    return $this->belongsTo(User::class, 'user_id');
}
```

Подання заяви про реєстрації субсидії та пільгових надбавок розділено в три кроки. Щоб не було надмірності в базі данних розіб’ємо їх на чотири частини. В даних частинах зробемо йде обробка інформації яка забезпечує збереженя, редагування та видалення запису користувача. В даних таблицях є зв’язок один до одного з таблицею користувачів (приклад 3.3 та 3.4). Опишиме дані таблиці.

Таблиця 3.3 – Структура та опис таблиці data_users прототипу

Назва атрибуту	Тип даних	Короткий опис
_ID	Integer	Первинний ключ, унікальний ідентифікатор рядків

structure_unit	String	Структура підрозділу
name	String	Повне ім’я користувача
place_of_residence	String	Місце проживання
phone	String	Телефон
pasport_seria	String	Серія паспорта
issuance_pasport	String	Ким виданий паспорт
card_taxses	String	Номер платника податків
user_id	Intager	Ідентифікатор користувача
created_at	Timestamps	Дата створення
updated_at	Timestamps	Дата оновлення

Дана таблиця бере дані про користувача, для того щоб скласти заяву для призначення житлової субсидії, на придбання скрапленого газу, твердого та рідкого пічного побутового палива.

Таблиця 3.4 – Структура та опис таблиці housing_comunal прототипу

Назва атрибуту	Тип даних	Короткий опис
_ID	Integer	Первинний ключ, унікальний ідентифікатор рядків
house	String	Будинок
gas	String	Кількість використання газу
cold_water	String	Холодна вода
hot_water	String	Гаряча вода

drainage	String	Водовідведення
centralized_heating	String	Центалізоване опалення
electricity_supply	String	Електропостачання
household_waste_removal	String	Вивезення відходів
user_id	Intager	Ідентифікатор користувача
created_at	Timestamps	Дата створення
updated_at	Timestamps	Дата оновлення

В таблиці є відомості про житлово-комунальні послуги, які проживаю та використовують особи, що зреєстровані. А саме відноситься: утримання будинків та споруд, газопостачання, централізоване постачання холодної води, централізоване постачання гарячої води, водовідведення, електропостачання, центалізоване опалення, вивезення побутових відходів.

Таблиця 3.5 – Структура та опис таблиці revenues прототипу

Назва атрибуту	Тип даних	Короткий опис
_ID	Integer	Первинний ключ, унікальний ідентифікатор рядків
full_name	String	Повне ім’я
birth	String	Дата народження
card	String	Номер платника податків
type_of_income	String	Вид доходу
amount_income	String	Сумма доходу
source_income	String	Джерело доходу
user_id	Intager	Ідентифікатор користувача
created_at	Timestamps	Дата створення
updated_at	Timestamps	Дата оновлення

Таблиця 3.6 – Структура та опис таблиці revenues прототипу

Назва атрибуту	Тип даних	Короткий опис
_ID	Integer	Первинний ключ, унікальний ідентифікатор рядків
name	String	Ім’я та Ініціали
kind	String	Джерело доходу
user_id	Intager	Ідентифікатор користувача
created_at	Timestamps	Дата створення
updated_at	Timestamps	Дата оновлення

Кожна база даних містить відповідну "Модель", яка використовується для взаємодії з цією таблицею. Щоб розпочати, створимо модель. Вона, як правило, знаходиться у каталозі app, але можем розміщувати їх у будь-якому місці, яке можна автоматично завантажувати відповідно до нашого файла composer.json. Всі моделі розширюють клас Illuminate \ Database \ Eloquent \ Model.

Лістинг 3.5 – Скріпт моделі revenues

```
{
    protected $table = "revenues";
    protected $fillable = [
        'full_name',
        'birth',
        'card',
        'type_of_income',
        'amount_income',
        'source_income',
        'user_id',
    ];
}
```

Middleware це зручний механізм для фільтрації HTTP-запитів, що надходять у нашу програму. Наприклад, Laravel включає в себе проміжне програмне забезпечення, яке перевіряє користувача вашої заявки. Якщо користувач не автентифікується, проміжне програмне забезпечення переспрямовуватиме користувача на екран входу. Проте, якщо користувач автентифікується, проміжне програмне забезпечення дозволить запиту продовжувати входити в додаток.

Звичайно, додаткове проміжне програмне забезпечення може бути написано для виконання різних завдань, крім аутентифікації. Проміжне програмне забезпечення CORS може відповідати за додавання відповідних заголовків для всіх відповідей, які залишають вашу заявку. Логотип проміжного програмного забезпечення може записувати всі вхідні запити до нашої програми.

В рамках Laravel передбачено декілька проміжних програм, включаючи проміжне програмне забезпечення для автентифікації та захист CSRF. Всі ці проміжні версії містяться в каталозі додатків / Http / Middleware.

Лістинг 3.6 – Скріпт middleware до профіля

```
public function handle($request, Closure $next)
{
    $article = \App\Profile::find($request->parameter('id'));

    if ($article->user_id != Auth::user()->id)
    {
        abort(404);
    }
    return $next($request);
}
```

Щоб отримати екземпляр поточного HTTP-запиту за допомогою ін'єкції залежності, повинні навести клас Illuminate \ Http \ Request у своєму методі контролера. Екземпляр вхідного запиту буде автоматично введено службовим контейнером.

Лістинг 3.7 – Скріпт request до контроллераHousingComunal

```
public function rules()
{
    return [
        'house' => 'required',
        'gas' => 'required',
        'cold_water' => 'required',
        'hot_water' => 'required',
        'drainage' => 'required',
        'centralized_heating' => 'required',
        'electricity_supply' => 'required',
        'household_waste_removal' => 'required',
    ];
}
```

Замість того, щоб визначати всю нашу логіку обробки запитів у вигляді замків у файлах маршрутів, ми можемо налаштувати цю поведінку за допомогою класів контролерів. Контролери можуть групувати відповідну логіку обробки запитів у єдиний клас. Контролери зберігаються в каталозі програми / Http / Controllers.

Розглянимо методи контролера які виводять, добавляють, редагують та видаляють дані з бази даних.

Лістинг 3.8 – Скріпт який виводить дані по користувачю

```
public function index()
{
    $comunals = HousingCommunal::where('user_id','=', Auth::id())->get();
    return view('/interface.page.housing_communal.index',
compact('comunals'));
}
```

Лістинг 3.9 – Скріпт який зберігає дані по користувачю

```
public function store(HousingCommunalRequest $request)
{
    $comunal = new HousingCommunal($request->all());
    $comunal->house = $request->house;
    $comunal->gas = $request->gas;
    $comunal->cold_water = $request->cold_water;
    $comunal->hot_water = $request->hot_water;
    $comunal->drainage = $request->drainage;
    $comunal->centralized_heating = $request->centralized_heating;
    $comunal->electricity_supply = $request->electricity_supply;
    $comunal->household_waste_removal = $request->household_waste_removal;
    $comunal->user_id = Auth::id();
    $comunal->save();
    return redirect('/revenues/create');
}
```

Лістинг 3.10 – Скріпт який зберігає дані по користувачю

```
public function update(HousingCommunalRequest $request, $id)
{
    HousingCommunal::find($id)->update($request->all());
    return redirect('/housing_communals');
}
```

Всі маршрути якими можна користуватися на прототипі, Laravel визначає у файлах, які знаходяться в каталозі routes. Ці файли автоматично завантажуються системою. Файл routes/web.php визначає маршрути, які використовуються для веб-інтерфейсу прототипу. Ці маршрути призначаються групі веб-проміжного програмного забезпечення, що надає такі функції, як стан сеансу та захист CSRF. Маршрути в routes/api.php призначені для групи проміжного програмного забезпечення.

Лістинг 3.11 – Скріпт який забезпечує маршрути по сторінках

```
Route::get('/', 'AppController@home');

Auth::routes();

Route::get('/home', 'HomeController@index')->name('home');
Route::get('/sencs', 'ProfilesController@sencs')->name('sencs');
Route::get('/profile/{id}', 'ProfilesController@show')->middleware('auth');
//Route::resource('data users', 'DataUsersController')->middleware('auth');
Route::resource('data_users', 'DataUsersController', ['except' => [
```

3.2 Особливості розробки клієнтської частини

Клієнська частина полягає в написанні HTML розмітки сторінки та складання стилів CSS. Blade - це простий, але потужний шаблонний движок, забезпечений Laravel. На відміну від інших популярних двигунів шаблонів PHP, Blade не обмежує використанням простих PHP-кодів у поглядах. Насправді, всі думки Blade складаються в простий PHP-код і кешуються до модифікації, а це означає, що Blade додає до програми суттєво нульові накладні витрати. Файли перегляду Blade використовують розширення файлу .blade.php і зазвичай зберігаються в каталозі ресурсів / переглядів (дивись лістинг 3.12).

Лістинг 3.12 –Приклад файлу .blade.php

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

@section('content')
<div class="container">
    <div class="row">
        <div class="col-md-8 col-md-offset-2">
            <div class="panel panel-default">
                <div class="panel-heading">Dashboard</div>

                <div class="panel-body">
                    @if (session('status'))
                        <div class="alert alert-success">
                            {{ session('status') }}
                        </div>
                    @endif

                    You are logged in!
                </div>
            </div>
        </div>
    </div>
</div>
@endsection
```

Дві основні переваги використання Blade - це спадщина шаблону та розділи. По-перше, ми розглянемо макет сторінки "майстер". Оскільки більшість веб-програм підтримують один і той же загальний макет на різних сторінках, зручно визначити цей макет як єдиний перегляд (дивись лістинг 3.13).

Лістинг 3.13 –Скріпт файлу master.blade.php

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <meta name="description" content="">
    <meta name="author" content="">
    {{--<title>@yield('title')</title>--}}
    <title>DIPLOM</title>
    @include('interface.css')
    @yield('css')
</head>
<body>
@include('interface.nav')

@yield('content')
```

```
@include('interface.footer')
@include('interface.scripts')
@yield('scripts')
</body>
</html>
```

Стилі css в Laravel находяться в папці public. Для приємного відображення даних ми зробимо свої стилі (лістинг 3.14).

Лістинг 3.14 –Стилі style.css

```
header.main-header .header-container {
    margin: 0 auto;
    padding: 60px 0px;
    background-color: rgba(8, 134, 196, 0.96);
    font-family: "PTSansCaptionBold";
    background-image: url(../image/crest.png);
    background-repeat: no-repeat;
    background-position: 75px 82px;
}

header.main-header .text {
    padding-left: 190px;
    text-shadow: 0 -2px 3px rgba(0, 3, 1, 0.64);
    color: #fff;
}

.show-title{
    text-align: center;
}
```

Також для відображення стилів та положення блоків на різних пристроях, адптивності в мобільному вигляді, планшеті використаємо Bootstrap.

Лістинг 3.15 – Приклад підключення Bootstrap

```
<!doctype html>
<html lang="en">
  <head>
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-
fit=no">

    <!-- Bootstrap CSS -->
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-
beta.3/css/bootstrap.min.css" integrity="sha384-
Zug+QiDoJOrZ5t4lssLdxGhVrurbmBWopoEl+M6BdEfwnCJZtKxi1KgxUyJq13dy"
crossorigin="anonymous">
```

```
    <title>Hello, world!</title>
  </head>
  <body>
    <h1>Hello, world!</h1>

    <!-- Optional JavaScript -->
    <!-- jQuery first, then Popper.js, then Bootstrap JS -->
    <script src="https://code.jquery.com/jquery-3.2.1.slim.min.js" integrity="sha384-
KJ3o2DKtIkvYIK3UENzmM7KCkRr/rE9/Qpg6aAZGJwFDMVNA/GpGFF93hXpG5KkN"
crossorigin="anonymous"></script>
    <script
src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/popper.min.js"
integrity="sha384-ApNbgh9B+Y1QKtv3Rn7W3mgPxhU9K/ScQsAP7hUibX39j7fakFPskvXusvfa0b4Q"
crossorigin="anonymous"></script>
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-
beta.3/js/bootstrap.min.js" integrity="sha384-
a5N7Y/aK3qNeh15eJKGWxsqtnX/wWdSZSKp+81YjTmS15nvnvxKHuzaWwXHDli+4"
crossorigin="anonymous"></script>
  </body>
</html>
```

Для асинхронності запитів використаєм JavaScript, а саме підключеме бібліотеку JQuery (лістинг 3.16).

Лістинг 3.16 – Приклад асинхронного запиту JQuery

```
function getSliders() {
        $.get("/api/sliders", function (data) {
            addSliders(data.data);
        })

}

function addSliders(data) {
        let elemens = '';
        $.each(data, function (key, value) {
            elemens = elemens +
                '<div class="item">' +
                '<img src="../' + value.image + '" alt="..."></div>'
        });
        $(".myslider").html(elemens);
        $(".item:first").addClass('active');
}
```

Представлені об'єктами jqXHR $.ajax(), починаючи з jQuery 1.5, реалізують інтерфейс Promise, надаючи їм всі властивості, методи та поведінку Promise Ці методи використовують один або декілька аргументів функції, які викликаються при завершенні запиту $ ajax(). Це дозволяє призначити декілька зворотних викликів на одному запиті та навіть призначати зворотні повідомлення після завершення запиту. (Якщо запит вже завершено, зворотний виклик буде звільнений негайно.) .

Також, Bootstrap вимагає використання функції JavaScript. Зокрема, вимагають jQuery, Popper.js та наших власних плагінів JavaScript.

Лістинг 3.17 – Bootstrap jQuery

```
jQuery.fn = jQuery.prototype = {

    // The current version of jQuery being used
    jquery: version,

    constructor: jQuery,

    // The default length of a jQuery object is 0
    length: 0,

    toArray: function () {
        return slice.call(this);
    },

    // Get the Nth element in the matched element set OR
    // Get the whole matched element set as a clean array
    get: function (num) {

        // Return all the elements in a clean array
        if (num == null) {
            return slice.call(this);
        }

        // Return just the one element from the set
        return num < 0 ? this[num + this.length] : this[num];
    },
}
```

Laravel Mix забезпечує безперешкодний API для визначення кроків по створенню Webpack для прототипу, використовуючи декілька стандартних передпроцесорів CSS та JavaScript. Через простий метод ланцюжка, вільно визначається контейнер для активів. Laravel Mix посилається на попередньо сконфігурований файл webpack.config.js, щоб працювати як можна швидше. Інколи може виникнути потреба вручну змінити цей файл. Може бути спеціальний завантажувач або плагін, на який потрібно посилатися, або, можливо, краще використовувати Stylus замість Sass.

Компонентна система є ще одним важливим поняттям у Laravel Mix, оскільки це абстракція, яка дозволяє нам створювати великомасштабні програми, що складаються з невеликих, автономних і часто багаторазових

компонентів. будь-який тип інтерфейсу програми може бути абстрагований у дерево компонентів (рисунок 3.9).

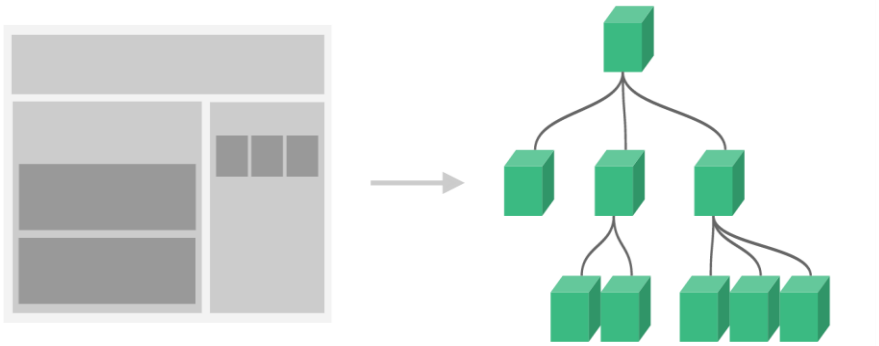


Рисунок 3.9 – Дерево компонентів

Це виглядає дуже схожим на виготовлення шаблону, але Laravel Mix зробив велику роботу. Дані та DOM тепер пов'язані, і все зараз реактивно.

Детальніше, частина Laravel Mix представлений у лістингу 3.18.

Лістинг 3.18 – Laravel Mix реактивність

```
<div id="app-2">
  <span v-bind:title="message">
    Hover your mouse over me for a few seconds
    to see my dynamically bound title!
  </span>
</div>
var app2 = new Vue({
  el: '#app-2',
  data: {
    message: 'You loaded this page on ' + new Date().toLocaleString()}
```

Атрибут v-bind, називається директивою. Директиви мають префікс v, щоб вказати, що вони є особливими атрибутами, наданими Laravel Mix, і вони застосовують особливу реактивну поведінку до видозміненого DOM.

3.3 Інтерфейс користувача та інструкція по використанню

Під час створення автоматизованої системи електронної реєстрації субсидій та пільгових надбавок. було розроблено веб-прототип на базі PHP framework Laravel. Так як, автоматизована система електронної реєстрації субсидій та пільгових надбавок орієнтована на на велику кількість користувачів, вона повинна мати зручний та зрозумілий інтерфейс по відношенню до користувачів програмних продуктів. Необхідно зазначити, що саме поняття зручний та зрозумілий інтерфейс - це вектор, який містить відповідно до міжнародної класифікації сім вимог до інтерфейсу користувача:

- відповідність завдань, що вирішуються користувачем;
- відповідність очікуванням користувача;
- легкість застосування;
- керованість;
- стійкість до помилок;
- адаптованість;
- легкість вивчення.

Графічний інтерфейс користувача програмою під framework Laravel, побудований, як було вже сказано, на основі використання Blade Master(лістинг 3.13), які формують шаблони. View що зображуються (рис. 3.10), складають головні компоненти сторінки що формується.

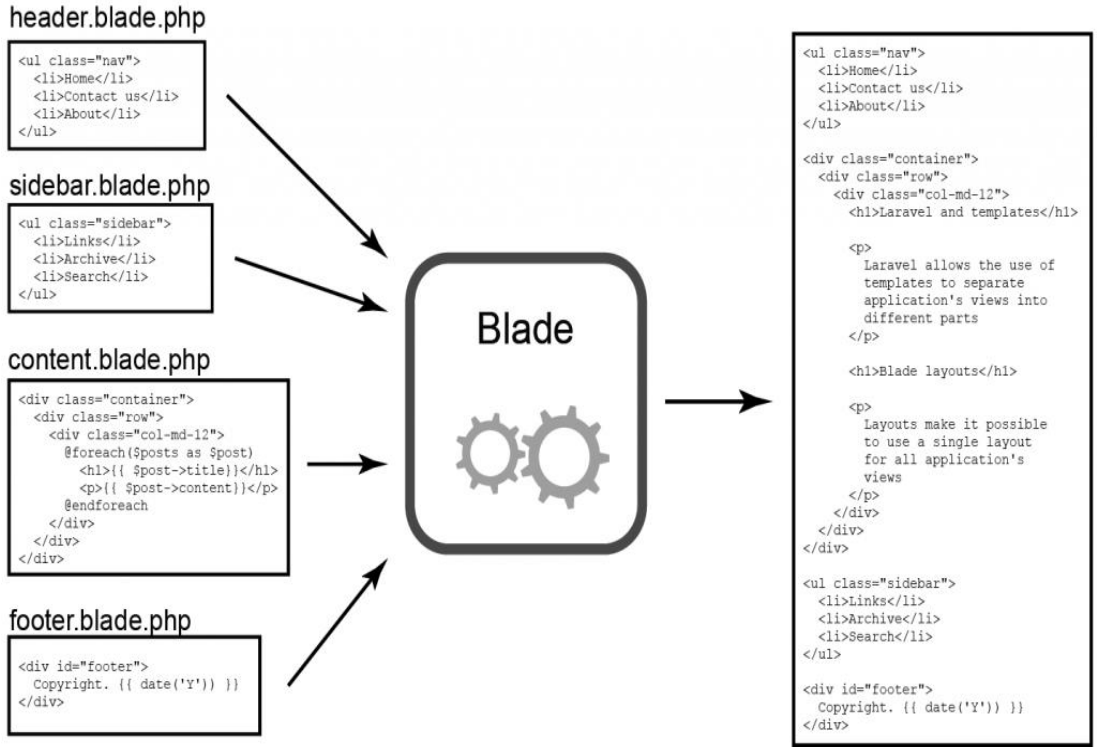


Рисунок 3.10 – Компоненти master.blade.php

Отже, головну сторінку веб-додатку представлено на рисунку 3.11:

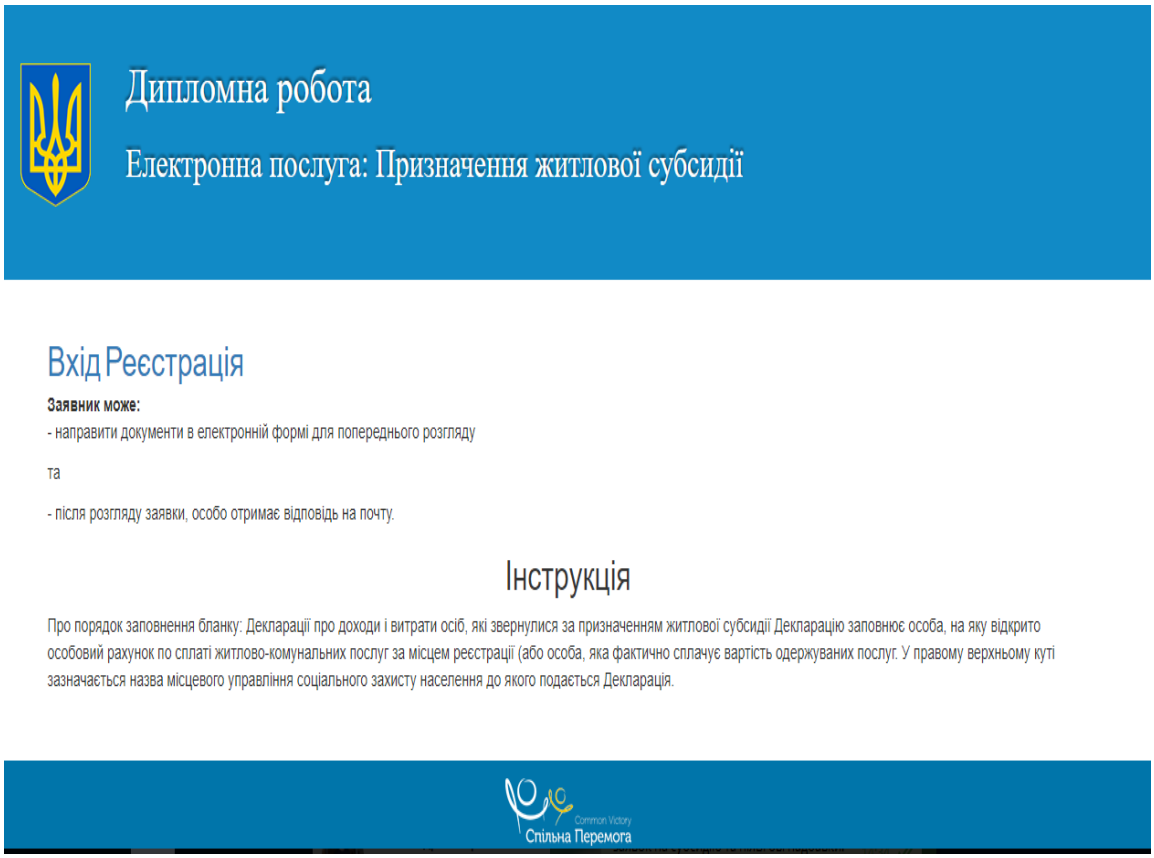


Рисунок 3.11 – Головна сторінка веб-додатку

На головній сторінці користувач має можливість входу в систему, реєстрації, та перегляду порядок заповнення дланку.

Отже перейшовши до сторінки входу прототипу користувач бачe наступне:

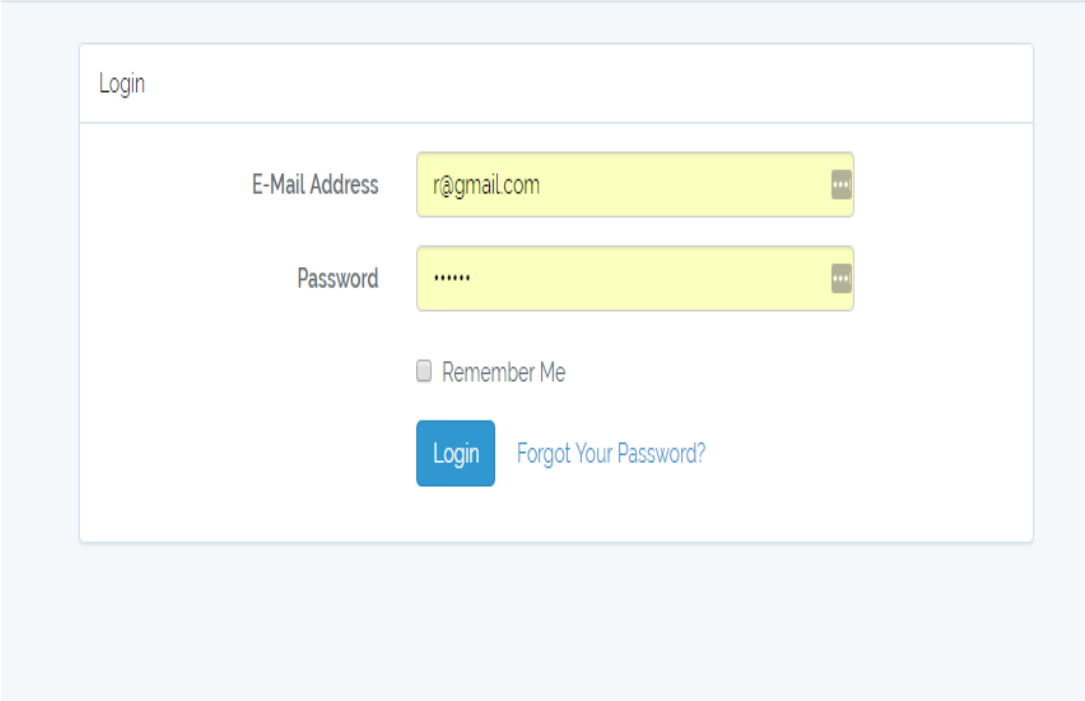


Рисунок 3.12 –Сторінка входу веб-додатку

На цій сторінці користувач ідентифікується в системі як зареєстрована ооба. Для цього йому потрібно вести свій логін (email), та пароль. Пароль має бути не менше шести символів.Також, якщо втрачено пароль його можно відновити перейшовши на сторінку Forgot Your Password:

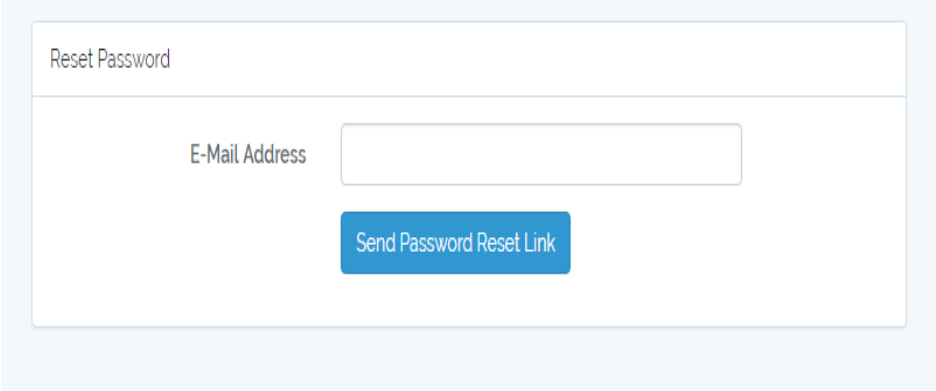


Рисунок 3.13 –Сторінка відновлення пароля

Потрібно ввести свій email та відправити його. Після чого система автоматично згенерує пароль та відправить користувачу на почту.

Відповідно, відзначивши на формі входу Remember Me система запам’ятає користувача.

Якщо повернутися на головну сторінку то користувач може зареєструватися в системі.

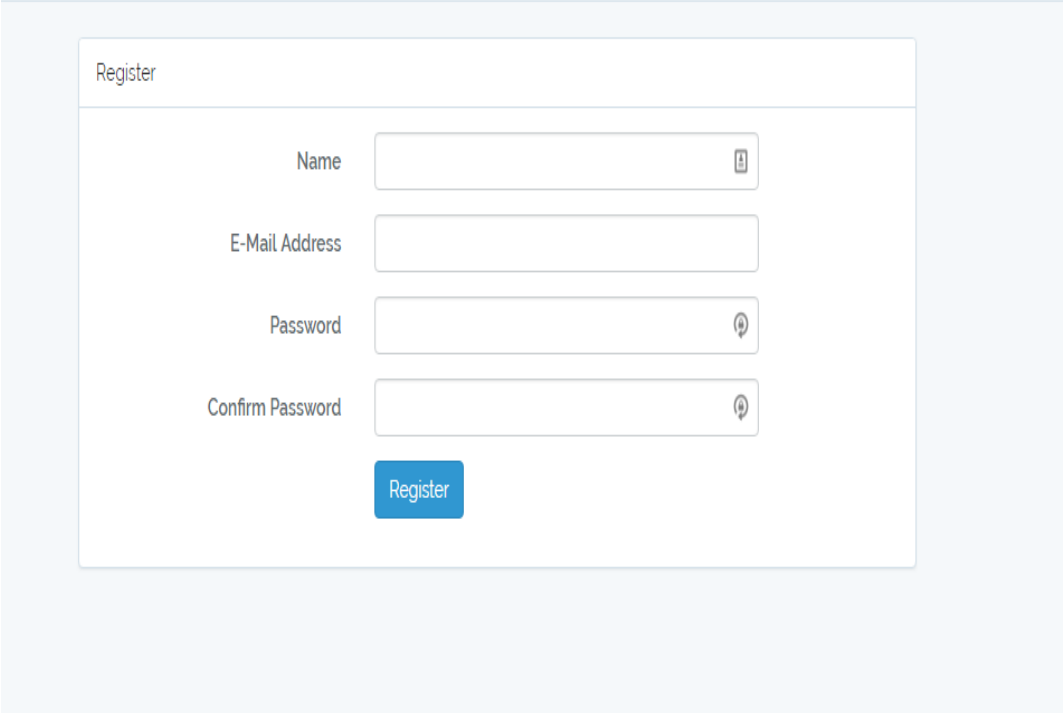


Рисунок 3.14 –Сторінка реєстрації користувача

Люди які завітали вперше на даний портал, потрібно буде зареєструватися, коректно вівши такі поля, як ім’я, email адресу, пароль та

пройти підтвердженн я пароля. В разі допушення помилки система розпізнає її та зробить підказку(рис. 3.15).

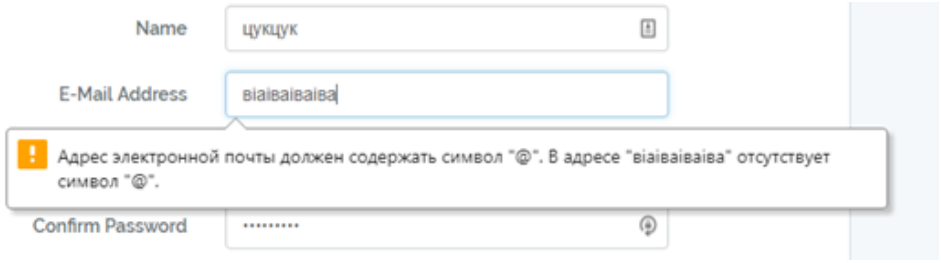


Рисунок 3.15 –Сторінка валідації форми

Після авторизації або реєстрації прототип системи відправе вас на сторінку профіля.

Рисунок 3.16 –Сторінка профіля

На даній сторінці присутне навігаційне вікно по профілю користувача, яке складається з пунктів для управління та електроної реєстрації заяви про надання субсидії та пільгових надбавок. Також знаходиться випадаючий список для виходу з системи (рис. 3.17).

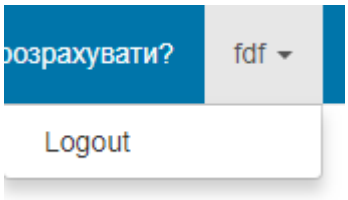


Рисунок 3.17 –Вихід з системи

Складання заяви на субсидію розмежовано на 3 розділи, які описані на головній сторінці профіля.

Першим кроком навігаційного вікна є складання заяви про призначення житлової супсидіїї:

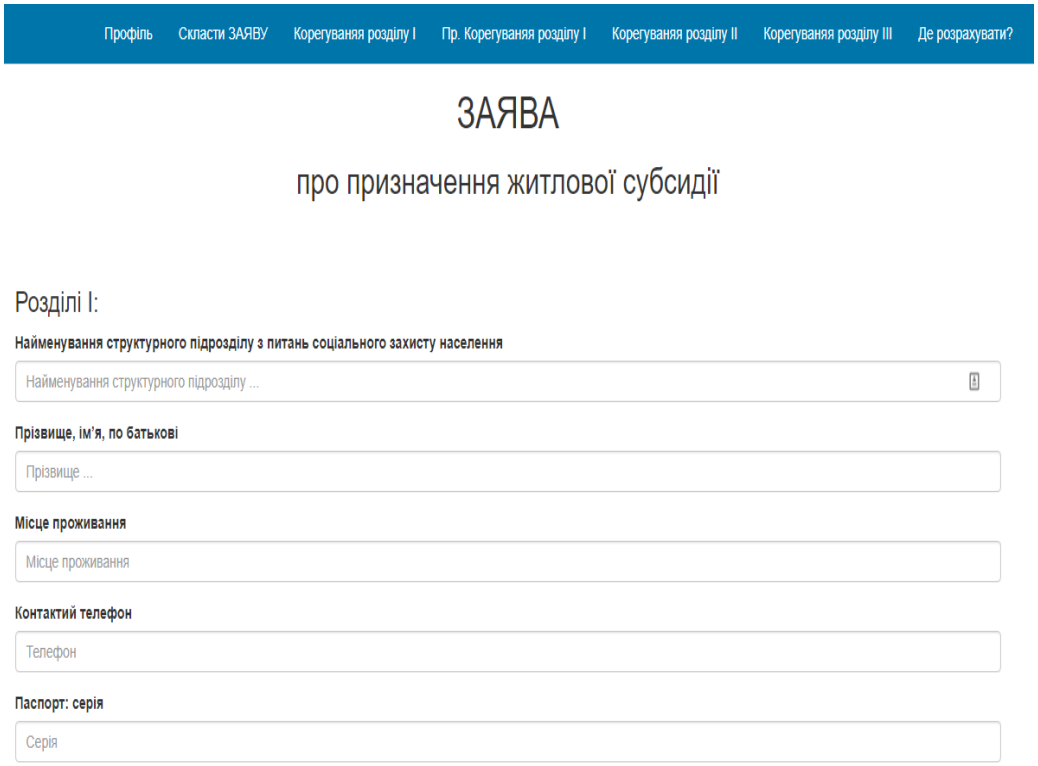


Рисунок 3.18 – Сторінка розділу I

Сторінка відображає розділ один в якому потрібно заповнити дані про користувача і складається з таких полів:

- прізвище, ім’я, по батькові;
- місце проживання;
- контактий телефон;
- паспорт: серія;
- ким і коли виданий;
- реєстраційний номер облікової картки платника податків.

Якщо поля заповнені коректно, після відправки даних на сервер ми переходимо до продовження розділу один і заповнюємо Відомості про житлово-комунальні послуги, якими користуються особи, які зареєстровані (фактично проживають) у житловому приміщенні/будинку.

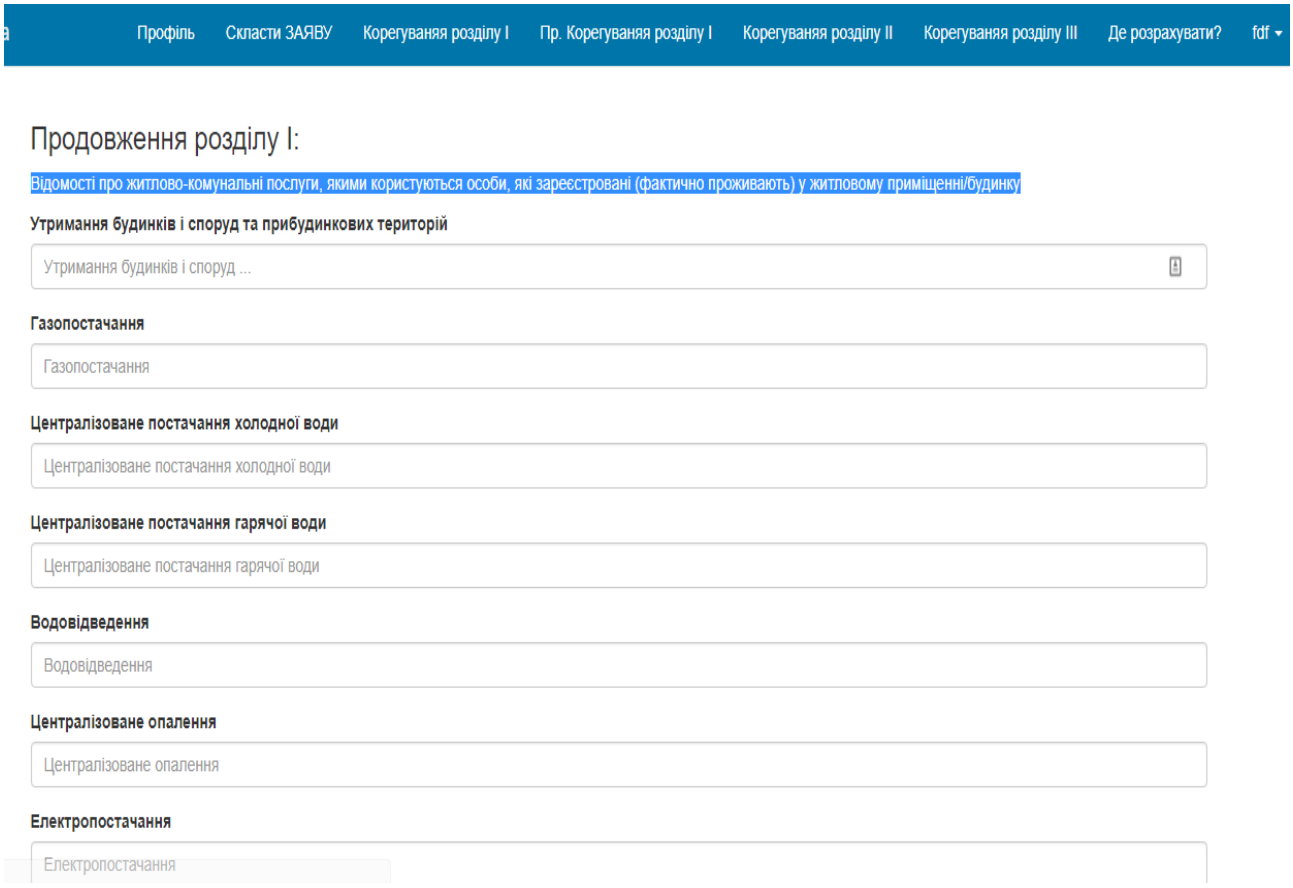


Рисунок 3.19 – Сторінка продовження розділу I

Опишемо поля якві потрібно коректно ввести:

- Утримання будинків і споруд та прибудинкових територій;
- Газопостачання
- Централізоване постачання холодної води
- Централізоване постачання гарячої води
- Водовідведення
- Централізоване опалення
- Електропостачання
- Вивезення побутових відходів

Після відправлення форми система автоматично переправе користувача на сторінку заповнення електронної субсидії розділу два (рис. 3.20).

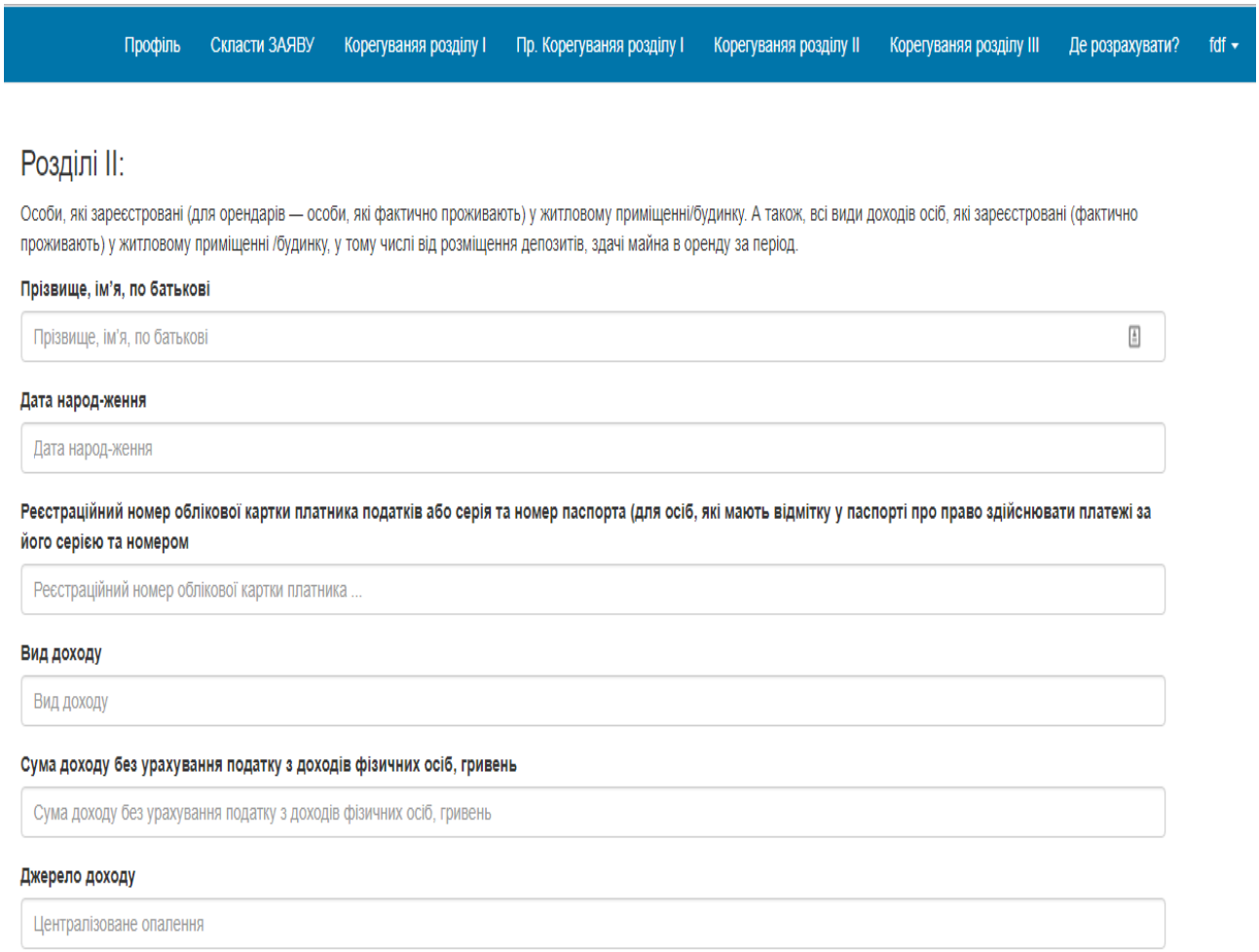


Рисунок 3.20 – Сторінка розділу II

Дана частина відповідає за особи, які зареєстровані (для орендарів — особи, які фактично проживають) у житловому приміщенні/будинку. А також, всі види доходів осіб, які зареєстровані (фактично проживають) у житловому приміщенні /будинку, у тому числі від розміщення депозитів, здачі майна в оренду за період.

Розглянемо поля які потрібно заповнити:

- прізвище, ім’я, по батькові;
- дата народ-ження;
- реєстраційний номер облікової картки платника податків або серія та номер паспорта (для осіб, які мають відмітку у паспорті про право здійснювати платежі за його серією та номером;
- вид доходу;
- Сума доходу без урахування податку з доходів фізичних осіб, гривень
- Джерело доходу

Перейдемо до третього розділу реєстрації субсидії

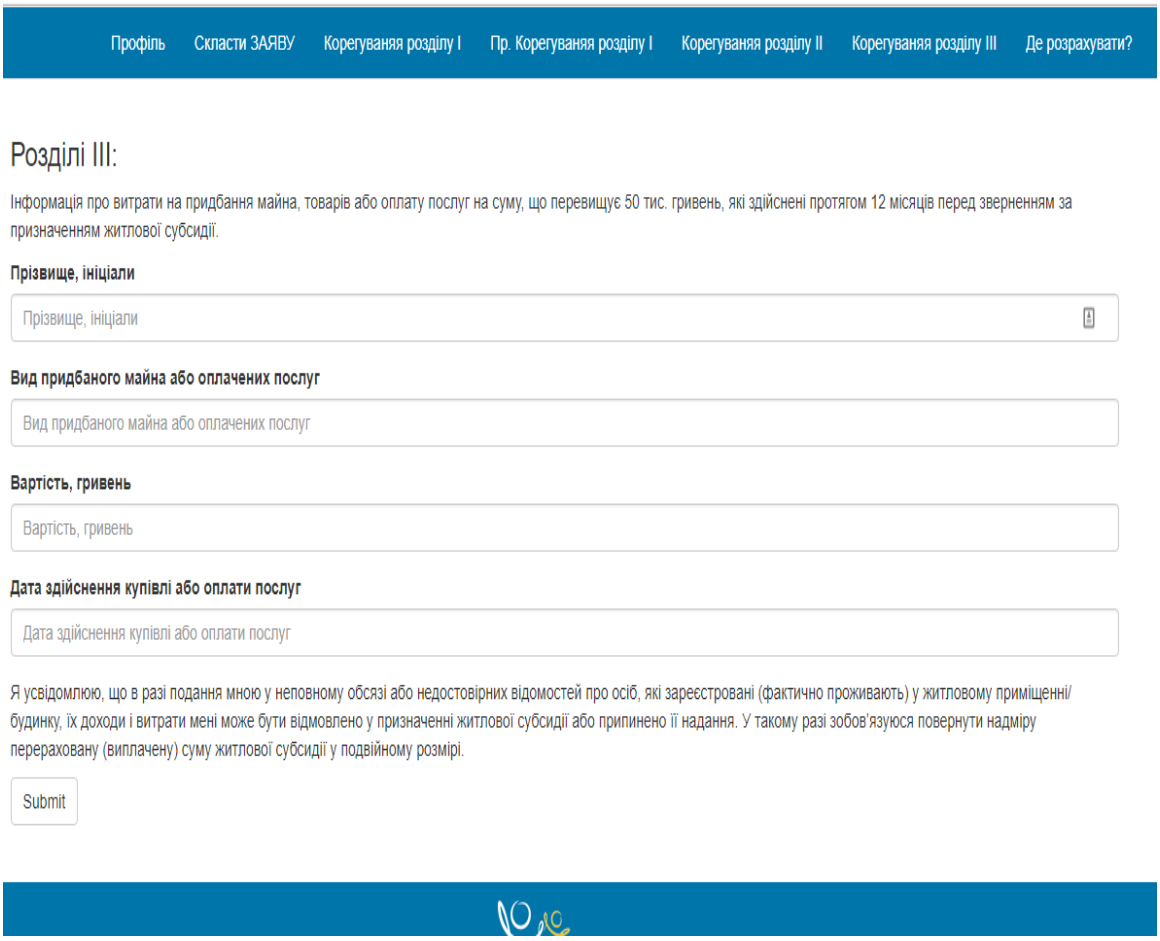


Рисунок 3.21 – Сторінка розділу III

Третій розділ інформую про витрати на придбання майна, товарів або оплату послуг на суму, що перевищує 50 тис. гривень, які здійснені протягом 12 місяців перед зверненням за призначенням житлової субсидії. Також даний розділ несе попередження, що в разі подання у неповному обсязі або недостовірних відомостей про осіб, які зареєстровані (фактично проживають) у житловому приміщенні/будинку, їх доходи і витрати може бути відмовлено у призначенні житлової субсидії або припинено її надання. У такому разі люди зобов'язані повернути надміру перераховану (виплачену) суму житлової субсидії у подвійному розмірі.

Опишимо поля третього розділу:

- прізвище, ініціали;
- вид придбаного майна або оплачених послуг;

- вартість, гривень;
- дата здійснення купівлі або оплати послуг.

Після того як ми завершили складати заяву система направить нас на сторінку привітання.

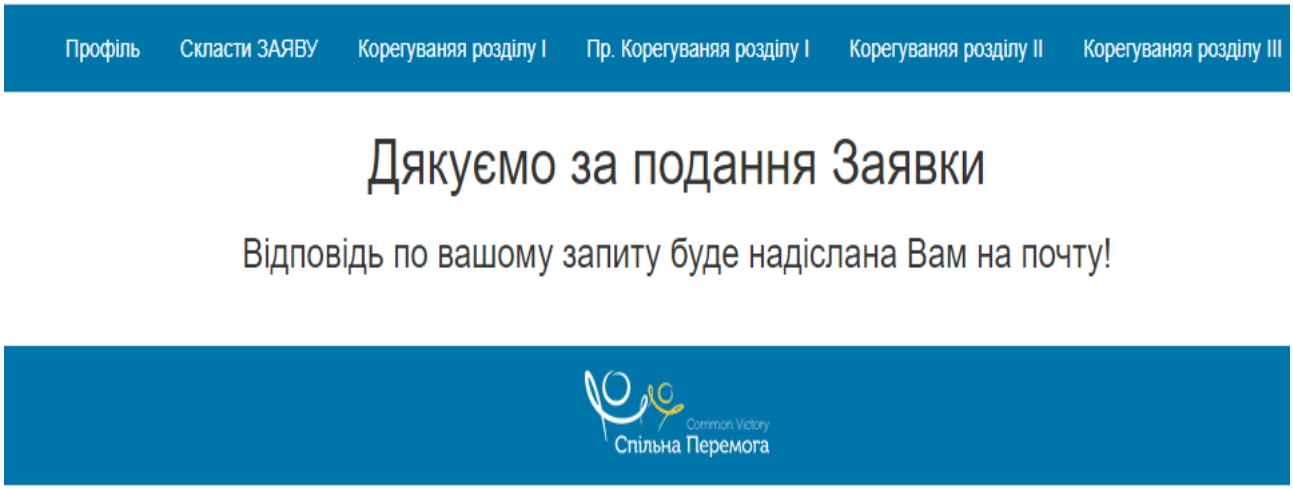


Рисунок 3.22 – Сторінка привітання

Для того щоб можно користувач міг маніпулювати своїми даними, було створено відображення дани заявок. Переглянемо інформацію про перший розділ подачі електронної субсидії.

Рисунок 3.23 – Сторінка запису що до I розділу

На рисунку 3.23 ми спостерігаємо кількість поданних документів по першому розділі.

Для того щоб відредагувати дані потрібно нажав кнопку Edit перейти до сторінки.

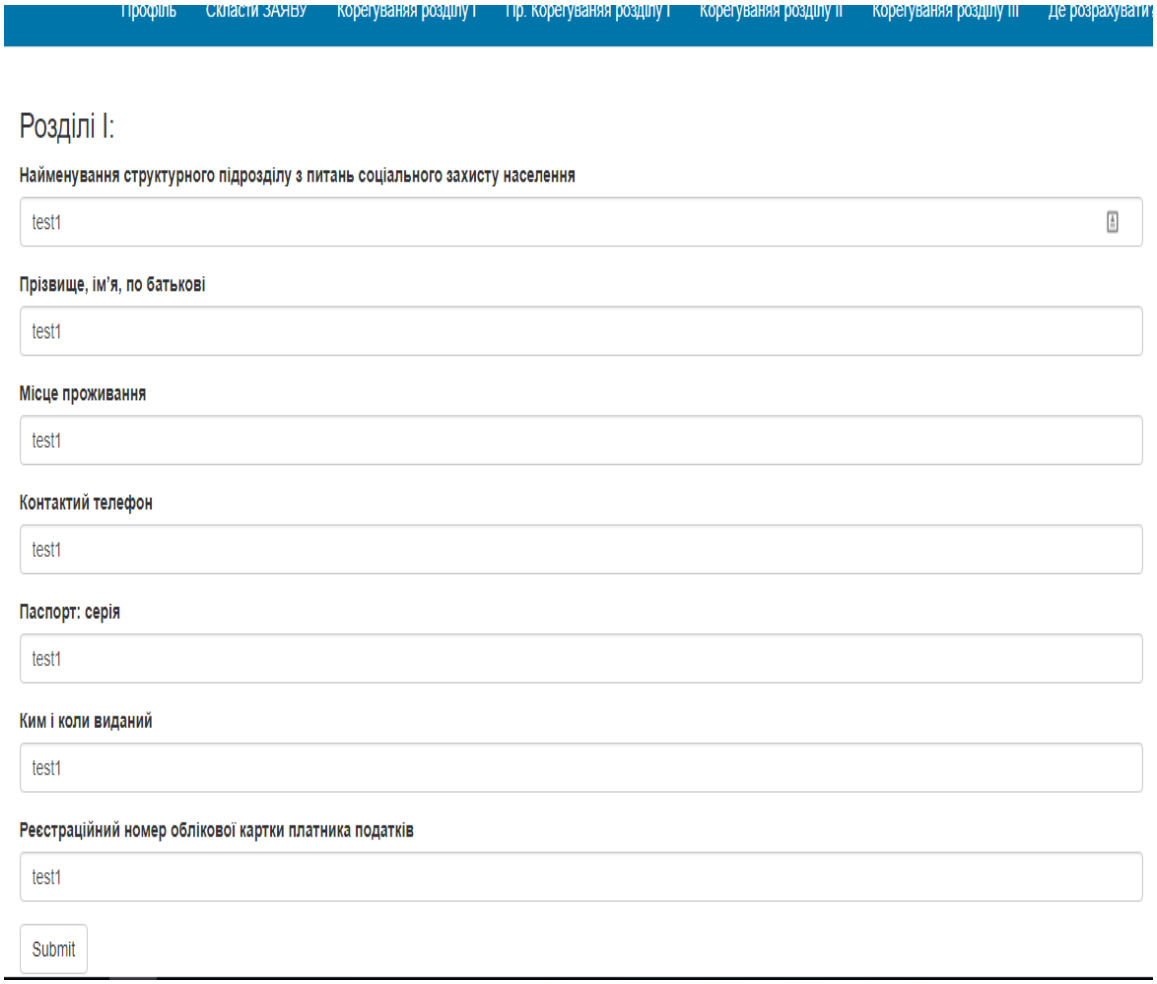


Рисунок 3.24 – Сторінка редагування що до I розділу

Ті поля що були описані для заповнення підтягнулися автоматично, потрібно лише виправити помилку та відправити дані. Але якщо з поля видалити значення система видасть помилку валідації.

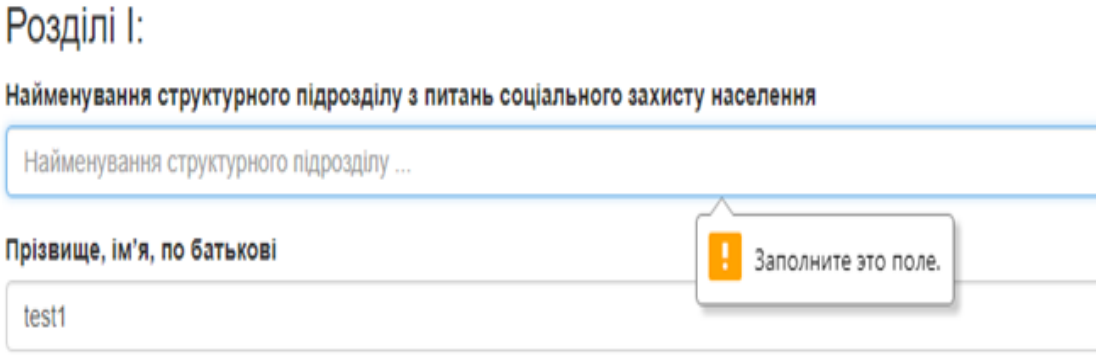


Рисунок 3.25 – Приклад валідації редагування що до I розділу

Такеж саме спостерігається і для інших розділів даного прототипу реєстрації субсидії та пільгових надбавок.

Переглянемо сторінки для продовження першого розділу .

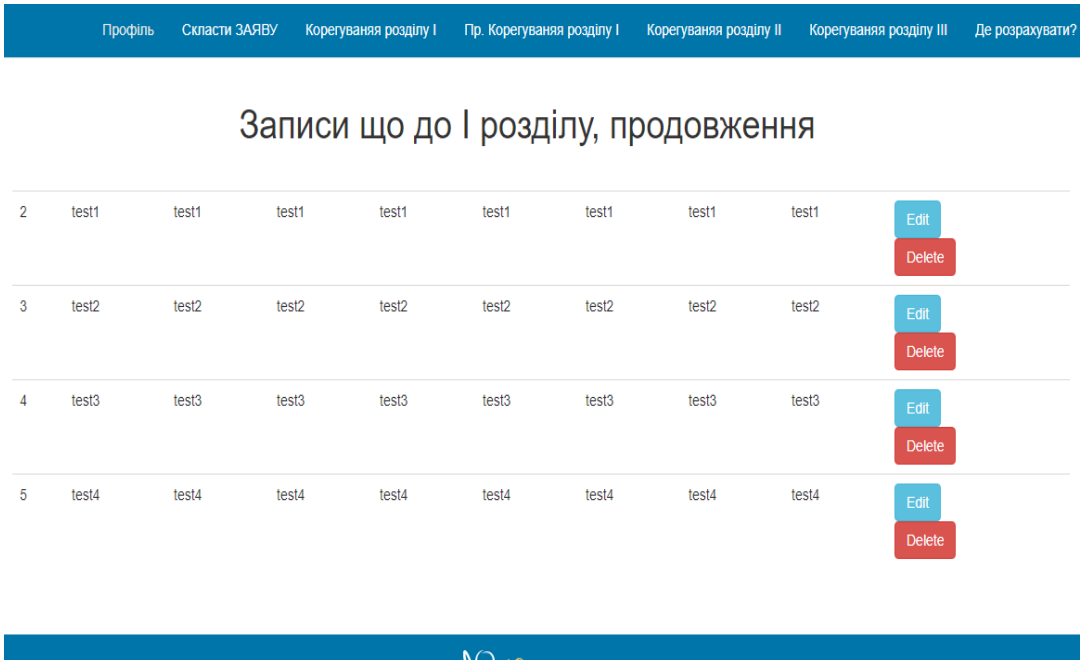


Рисунок 3.26 – Сторінка запису що до продовження I розділу

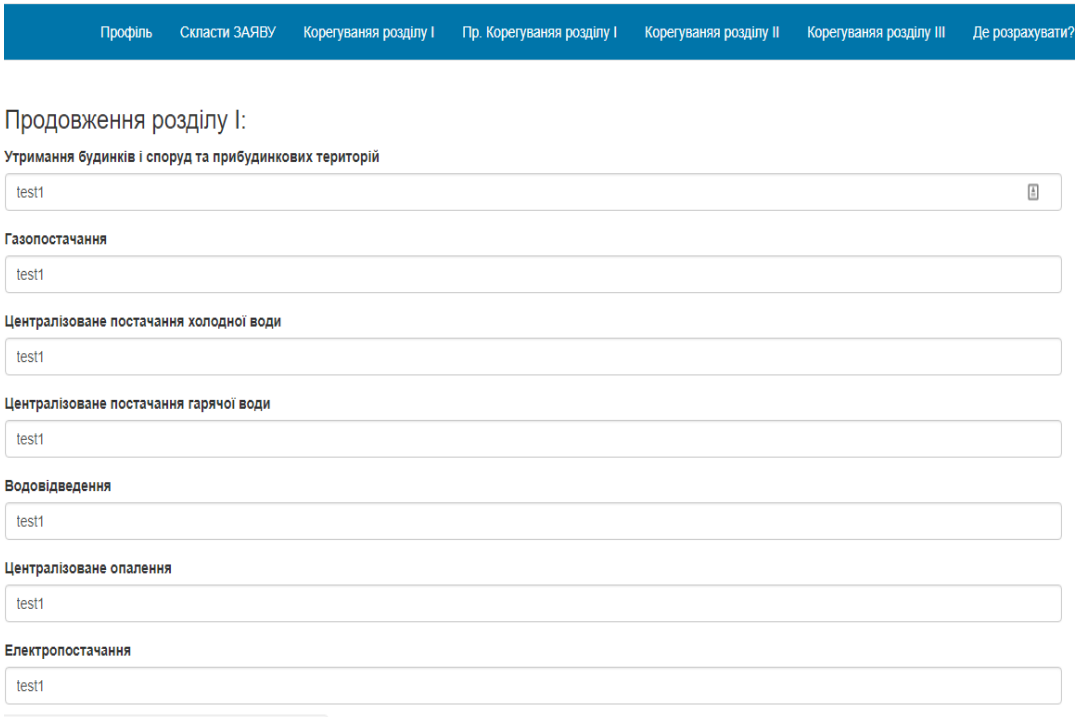


Рисунок 3.27 – Сторінка редагування що до продовження I розділу

Тепер переглянемо сторінки для другого розділу.

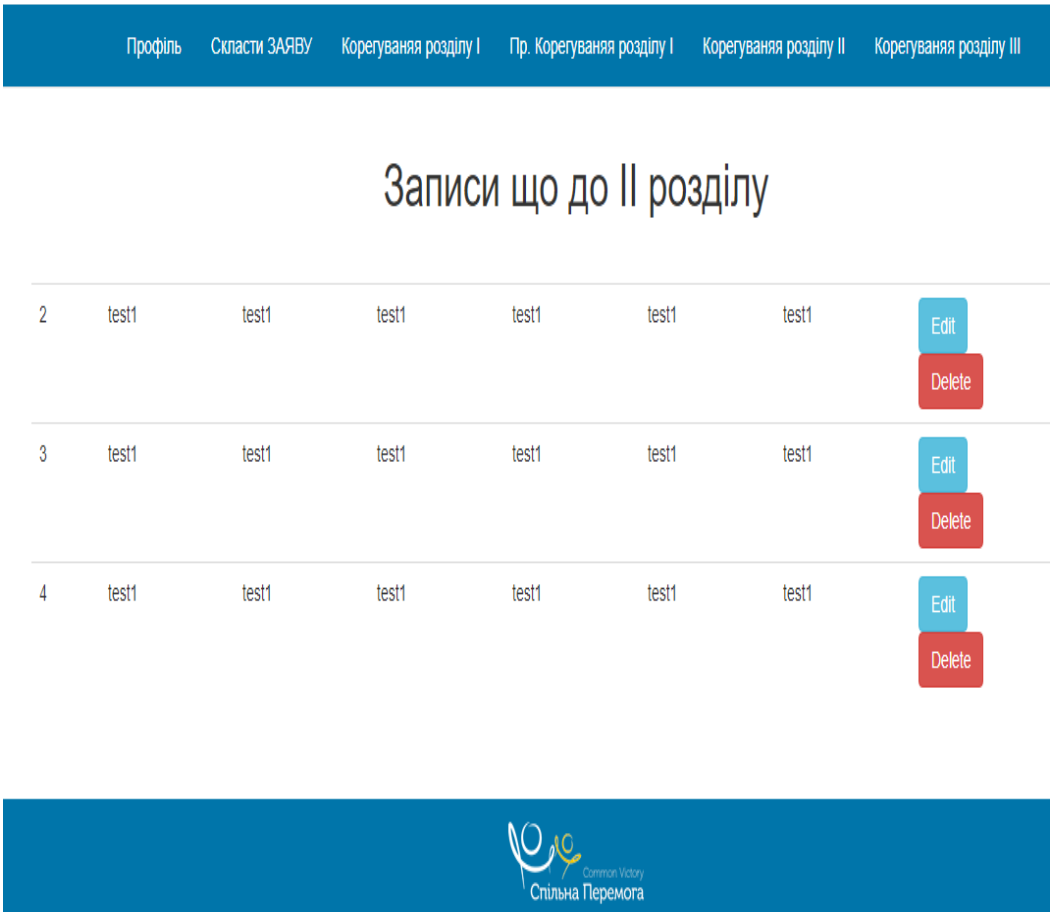


Рисунок 3.28 – Сторінка запису що до продовження II розділу

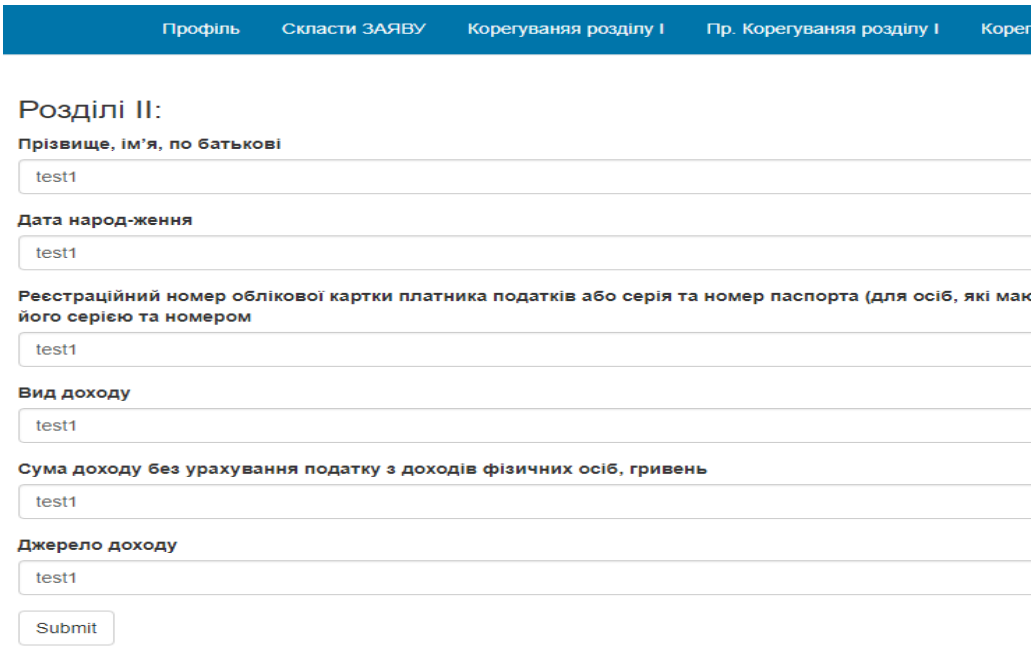


Рисунок 3.29 – Сторінка редагування що до II розділу

Також переглянемо третій розділ.

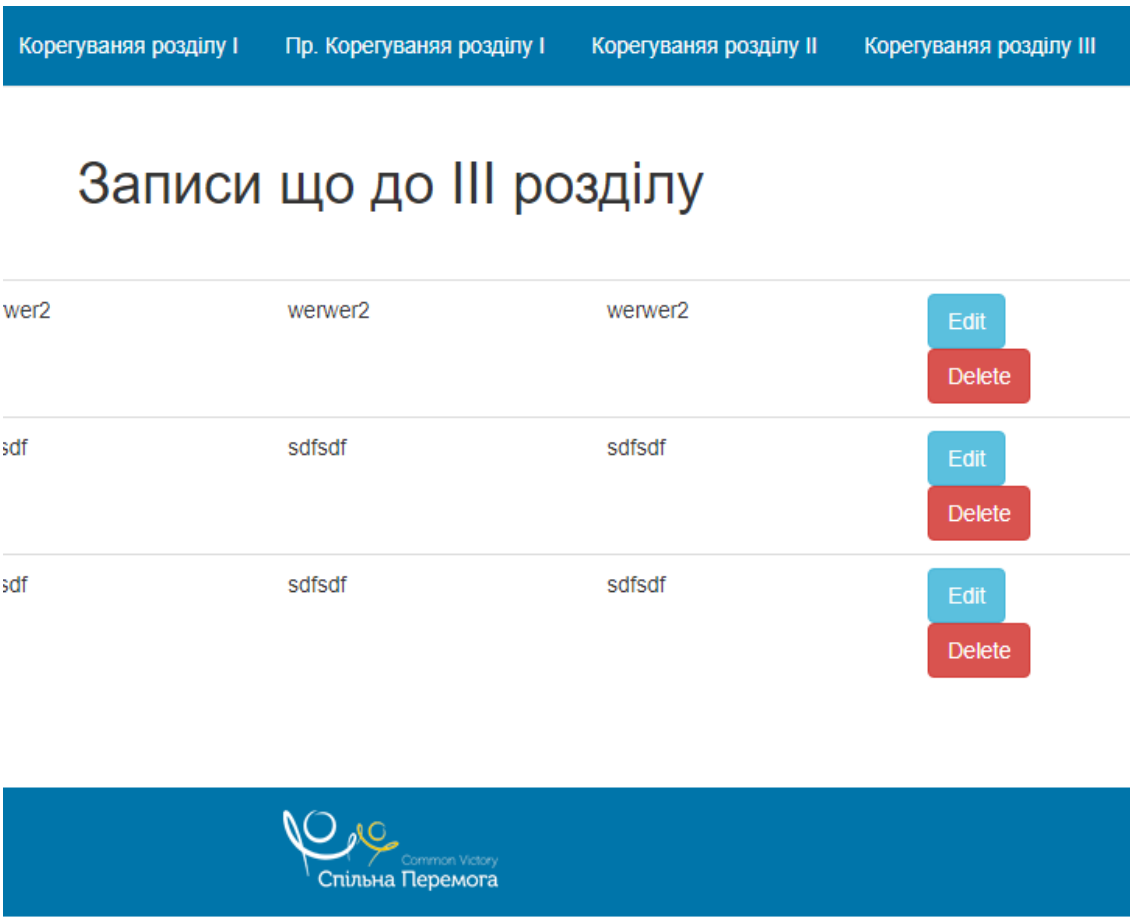


Рисунок 3.30 – Сторінка запису що до продовження III розділу

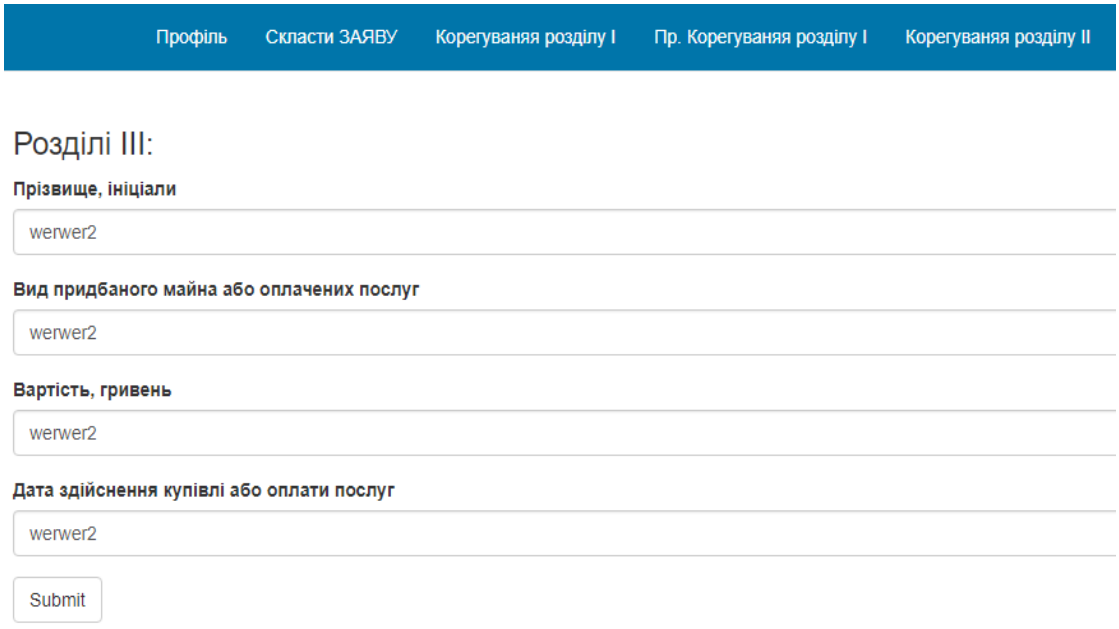


Рисунок 3.31 – Сторінка редагування що до III розділу

Також в даному прототипі можна видалити записи.

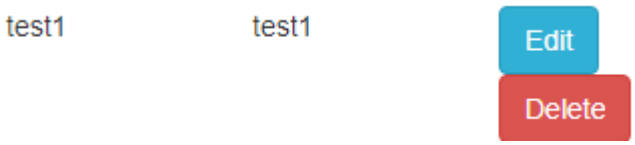


Рисунок 3.32 – Видалення даних

Також нажавши на панелі навігаці «Де розрахувати», перейдемо на портал розрахунку житлової субсидії.

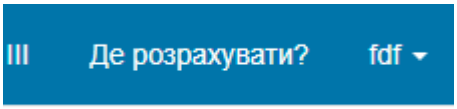


Рисунок 3.33 – Розрахунок субсидії.

3.4 Оцінка очікуваних ефектів від впровадження веб-орієнтованої інформаційної системи

Дана система автоматизації покликана вирішити основних 3 завдання:

– зменшення витрат часу на облік даних подання субсидій та пільгових надбавок;

– зменшення витрат часу на аналіз поточного стану подання субсидій та пільгових надбавок;

– покращення організованості й упорядкованості даних щодо подання субсидій та пільгових надбавок.

Застосування даної системи покращить діяльність самої реєстрації субсидій та пільгових надбавок:

– скорочення ручного обліку інформації;

– пришвидшення процесів пошуку інформації;

– зменшення ймовірності збоїв та помилок у роботі;

– зменшення часу на обробку та аналіз інформації про реєстрацію субсидій та пільгових надбавок.

Як відомо продуктивність та ефективність від провадження автоматизованої системи визначають порівнянням результатів її роботи і затрат всіх видів ресурсів, необхідних для її створення і розвитку.

Вкладення ресурсів у створення програмного продукту розраховуються за формулою:

$$K = K_1 + K_2 + K_3, \tag{3.1}$$

де K_1 – витрати на устаткування, грн.;

K_2 – витрати на ліцензійні програмні продукти, грн.;

K_3 – витрати на створення програмного продукту, грн.

Приймаємо К1 = 0, оскільки всі вищі учбові заклади використовують комп'ютери для своєї роботи, К2 = 0, оскільки не передбачено закупівлю додаткових програмних продуктів для впровадження системи.

Витрати на створення програмного продукту K_3 розраховуємо по формулі:

$$K_3 = З_1 + З_2 + З_3, \tag{3.2}$$

де $З_1$ – витрати праці програмістів-розробників, грн.;

$З_2$ – витрати комп'ютерного часу, грн.;

$З_3$ – непрямі (накладні) витрати, грн.

Витрати праці програмістів-розробників:

$$З_1 = \sum_{k=1}^{K} N_k \cdot r_k \cdot T_k \cdot K_{зар}, \quad (3.3)$$

де N_k – кількість розробників к-й професії, чол.;

r_k – годинна зарплати розробника к-й професії, грн.;

T_k – трудомісткість розробки для к-го розробника (кількість витраченого розробником часу), ч.;

Кзар = 1,3685– коефіцієнт відрахувань до фонду заробітної плати.

Візьмемо, що N_k = 1, оскільки в розробці програмного продукту приймала участь лише 1 людина.

Годинну заробітну плату працівника розрахуємо за формулою:

$$r_k = M_k / F_к^{мес}, \quad (3.4)$$

де M_k – місячна зарплата к-го розробника, грн.;

Fкмес – місячний фонд часу його роботи, година.

Приймаємо місячну заробітну плату програміста-розробника по Сумській області рівну 4000 грн., тоді маємо, що кількість календарних днів на 2017 рік = 365, кількість святкових днів = 10, кількість вихідних днів = 104, кількість днів, робота в які не проводиться = 114, кількість робочих днів = 251, кількість днів, що передують святковим, в які робочий день коротший на 1 годину в день = 4. Всього робочих годин за рік, при 40 годинному робочому тижні = 2004 години. Всього робочих годин за місяць = 2004/12=167 год.

Тоді, годинна зарплата розробника, враховуючи це, буде обчислена як:

$r_k = \frac{4000}{167} = 24{,}39$ грн./год.

Трудомісткість розробки $T_к$ включає час виконання робіт і в даному випадку є рівним 300 годин.

Тепер розрахуємо витрати праці програміста-розробника: $З_1$=1·24,39·300·1,3685 = 10013,32 грн.

Витрати комп'ютерного часу розрахуємо за формулою:

$$З_2 = C_k \cdot F_0, \tag{3.5}$$

де C_k – вартість комп'ютерної години, грн.;

F0 – витрати комп'ютерного часу на розробку програми, годин.

Вартість комп'ютерної години обчислюється по формулі:

$$C_k = C_A + C_Э + C_{ТО}, \tag{3.6}$$

де C_A – амортизаційні відрахування, грн.;

$C_ф$ – енерговитрати, грн.;

$C_{ТО}$ – витрати на техобслуговування, грн.

Амортизаційні відрахування знайдемо за формулою:

$$C_A = C_i \cdot N_A / F_{год}, \tag{3.7}$$

де C_i = 6000 грн. – балансова вартість і-го устаткування, яке використовувалося для створення , грн.;

N_A – річна норма амортизації і-го устаткування, долі;

$F_{год}$ – річний фонд часу роботи і-го устаткування, година

Відповідно до чинного законодавства квартальна норма амортизації основних фондів 4 групи, які були задіяні у розробці складає 15%, тоді річна норма амортизації буде дорівнювати $N_4 = 0,6$.

$F_{год} = 2004$ год.

$C_A = 6000 \cdot 0,6/2004 = 1,8$ грн.

Енерговитрати розрахуємо за формулою:

$$C_Э = P_Э \cdot C_{кВт}. \quad (3.8)$$

Комп'ютер сучасної моделі в середньому затрачає 800 Вт за годину, тоді $P_e = 0,08$ кВт/год.

Вартість 1 кВт/год. для споживачів другого класу (не промислові підприємства) становлять $C_{кВт} = 93,46$коп. за кВт/год.

$C_Э = 0,08 \cdot 0,9346 = 0,0747$ грн./год.

Витрати на техобслуговування розраховуємо за формулою:

$$C_{ТО} = r_{ТО} \cdot \lambda, \quad (3.9)$$

де $r_{ТО}$ – годинна зарплата працівника обслуговуючого устаткування, грн; приймаємо $r_{ТО} = 3000/167 = 17,96$ грн. /година;

λ – періодичність обслуговування:

$$\lambda = N_{ТО} / F_{тес}, \quad (3.10)$$

де $N_{ТО}$ – кількість обслуговувань устаткування в місяць, приймаємо $N_{ТО} = 1$

Місячний фонд часу роботи устаткування 167 годин.

$\lambda = 1/167 = 0,006$.

Витрати на техобслуговування складуть: $C_{ТО}$ =17,97·0,006 = 0,11 грн.

Звідси вартість комп'ютерної години: C_k = 1,8 + 0,0744 + 0,11 = 1,98 грн.

Отже витрати комп'ютерного часу складуть: $З_2$ = 1,98·300 = 594 грн.

Непрямі витрати приймемо: $З_3$ = 400 грн..

Звідси $К_З$ = 400 + 594 + 10013,32 = 11007,32 грн.

Витрати на створення системи складають: $К$ = 0 + 0 + 11007,32 = 11007,32 грн.

Тепер можна розрахувати річний ефект від впровадження автоматизованої системи та термін окупності системи.

Річна економія від зниження витрат на розробку проекту визначається за формулою:

$$Ер = Едв – Епв \text{ ,} \qquad (3.11)$$

де *Ер* – сума річної економії, грн.;

Едв – річні витрати на обробку інформації до впровадження системи, грн.;

Епв – річні витрати на обробку інформації після впровадження системи, грн.

Нехай середні витрати часу на ведення обліку та аналізу стану реєстрації субсидії та пільгових надбавок складають 30 хв.. З допомогою даної програми дані витрати можна скоротити до 15 хв. Тоді можна розрахувати економію часу, яка дорівнює 50%. За день здійснюється декілька подібних перевірок. Середнє вивільнення часу за один робочий день складатиме 30 хв., або 12,5 % робочого дня.

Користувач системи, нехай, в середньому отримує заробітну плату у розмірі 5000 грн. в місяць. Тоді фонд оплати праці одного працівника складає в середньому 60000.

Звідси річна економія складатиме, згідно до формули:

$E_р$ = (60000– 60000*0,125) = 7500 грн./рік.

Розрахуємо термін окупності капіталовкладень – період часу, протягом якого окупаються витрати на автоматизовану систему визначаються по формулі:

$$T_р = K / Ep. \quad (3.12)$$

$T_р$ = 11007,32 / 7500 = 1,47 років.

При ефективному використанні капіталовкладень розрахунковий термін окупності $T_р$ повинен бути менше нормативного $T_Н = 2,4$, у нашому випадку $T_р < T_н$. так як 0,85 < 2,4.

Отже, як бачимо розробка і використання програмного продукту є економічно доцільною, так як річна економія складе 7500 грн. за рік, а термін окупності капіталовкладень складає 1,47, тобто проект окупиться за півтора року.

Слід зауважити, що даний прототип може використовуватись на необмеженій кількості робочих станцій під керівництвом операційних систем Linux.

ВИСНОВКИ

Субсидія – це безготівкова допомога родині за адресом, яка забезпечує погашення витрат з оплати житлово-комунальних послуг.

Пільги – це переваги встановлені законодавством або іншими нормативними актами, що надається особі, або групі осіб порівняно з іншими громадянами.

Під час опалювального сезону виникає необхідність у занесенні великої кількості заявок на подання та отримання субсидій і пільгових надбавок персонального характеру. Особливістю даної галузі досліджень є те, що з необхідністю стикаються багато людей.

Аналіз стану автоматизаціъ подання заявок на отримання субсидій і пільгових надбавок показав, що зараз у спеціалізованих програмних продуктів вбудований доволі широкий функціонал, проте існують серйозні недоліки. Перш за все до недоліків відноситься те, що призначення субсидій в електронній формі діє тільки на певних областях. Дана тенденція у сфері ще раз підтверджує необхідність створення нової автоматизації подання заявок на отримання субсидій і пільгових надбавок.

В роботі визначено перелік вимог, яким повинна відповідати автоматизована система обліку персональних фінансів. Досліджено нормативно-правову базу України на предмет законодавчих актів, що стосуються отримання субсидій і пільгових надбавок.

Виходячи із сформованих вимог до автоматизованої системи обліку персональних фінансів, було розроблено проект та алгоритм вирішення задачі побудови системи за допомогою діаграми концептуальних класів IDEF0

Для розробки прототипу системи було обрано framework php Laravel. Кращим типом сховища даних стало MySql, через вимоги щодо

портативності, зручності розгортання та впровадження, та через високу кількість даних, які потрібно зберігати.

Для повноцінного функціонування автоматизованої системи на кінцевій робочій станції користувача необхідна лише встановити LAMP.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Гриценко К. Г. Програма виробничої практики [Текст] / Уклад.: канд. техн. наук, доцент, К.Г.Гриценко, канд. тех. наук, доцент, С.М. Новак. – Суми : УАБС, 2010.

2. Великий тлумачний словник сучасної української мови (з дод та допов.) / Уклад. і голов. ред. В. Т. Бусел. – К.: Ірпінь: ВТФ “Перун”, 2007. – 1736 с.

3. Про захист персональних даних [Електронний ресурс] / Режим доступу: http://zakon4.rada.gov.ua/laws/show/2297-17. – Заголовок з екрану.

4. LINQ to XML [Електронний ресурс] / Режим доступу: http://msdn.microsoft.com/en-us/library/bb387098.aspx. – Заголовок з екрану.

5. Нові правила отримання житлової субсидії [Електронний ресурс] / Режим доступу: https://glavcom.ua/publications/subsidiji-za-novimi-pravilami-yak-rozrahuvati-ta-otrimati-dopomogu-vid-derzhavi-351680.html. – Заголовок з екрану.

6. Електронна послуга: Призначення житлової субсидії [Електронний ресурс] / Режим доступу: https://subsidii.mlsp.gov.ua/. – Заголовок з екрану.

7. Все о субсидиях в Украине на 2017-2018 [Електронний ресурс] / Режим доступу: https://24tv.ua/ru/subsidija_2017_2018_ukraina_online_dokumenty_kalkuljator_kak_ofomit_subsidii_n882033/. – Заголовок з екрану.

8. Юредичний портал України [Електронний ресурс] / Режим доступу:http://www.lawportal.com.ua/zhitlova-subsidija-novi-pravila.html. – Заголовок з екрану.

9. Заповнення декларацію доходів [Електронний ресурс] / Режим доступу:https://subsidii.mlsp.gov.ua/help/dec/zapovnennya_deklaratsii_pro_dokh

odi_i_vitrati_osib__yaki_zvernulisya_za_priznachennyam_zhitlovoi_subsidii.htm. – Заголовок з екрану.

10. Законодавство України [Електронний ресурс] / Режим доступу: http://zakon3.rada.gov.ua/laws/show/848-95-п. – Заголовок з екрану.

11. Податкова соціальна пільга [Електронний ресурс] / Режим доступу: https://byhgalter.com/kazakova-podatkova-socialna-pilga-2018-shho-novogo/. – Заголовок з екрану.

12. Опис стандарту IDEF0 | EasyCode [Електронний ресурс] / Режим доступу: http://easy-code.com.ua/2011/03/opis-standartu-idef0. – Заголовок з екрану.

13. Нотація IDEF0 [Електронний ресурс] / Режим доступу: http://www.businessstudio.ru/wiki/docs/v4/doku.php/ru/csdesign/bpmodeling/idef0. – Заголовок з екрану.

14. Laravel Installation [Електронний ресурс] / Режим доступу: https://laravel.com/docs/5.5/installation#installation. – Заголовок з екрану.

15. Laravel [Електронний ресурс] / Режим доступу: https://laracasts.com/skills/laravel. – Заголовок з екрану.

16. How To Install MySQL on Ubuntu 16.04 [Електронний ресурс] / Режим доступу: https://www.digitalocean.com/community/tutorials/how-to-install-mysql-on-ubuntu-16-04. – Заголовок з екрану.

17. MySql [Електронний ресурс] / Режим доступу: http://znaimo.com.ua/MySQL. – Заголовок з екрану.

18. Bringing MySQL to the web [Електронний ресурс] / Режим доступу: https://www.phpmyadmin.net/. – Заголовок з екрану.

19. The Apache Software Foundation [Електронний ресурс] / Режим доступу: https://blogs.apache.org/foundation/entry/apache-is-open. – Заголовок з екрану.

20. How To Move an Apache Web Root to a New Location on Ubuntu 16.04 [Електронний ресурс] / Режим доступу:

https://www.digitalocean.com/community/tutorials/how-to-move-an-apache-web root-to-a-new-location-on-ubuntu-16-04. – Заголовок з екрану.

21. How To Install the Apache Web Server on Ubuntu 16.04 [Електронний ресурс] / Режим доступу: https://www.digitalocean.com/community/tutorials/how-to-install-the-apache-web-server-on-ubuntu-16-04. – Заголовок з екрану.

22. Mapping URLs to Filesystem Locations [Електронний ресурс] / Режим доступу: https://docs.phpmyadmin.net/en/latest/index.html. – Заголовок з екрану.

23. PHP - Taking the world by storm [Електронний ресурс] / Режим доступу: http://www.phpbbhq.com/developmentofphp.php. – Заголовок з екрану.

24. PhpStorm Early Access Program [Електронний ресурс] / Режим доступу: https://www.jetbrains.com/phpstorm/eap/. – Заголовок з екрану.

25. Docker Support in PhpStorm [Електронний ресурс] / Режим доступу: https://confluence.jetbrains.com/display/PhpStorm/Docker+Support+in+PhpStorm. – Заголовок з екрану.

26. Documentation PHP [Електронний ресурс] / Режим доступу: http://php.net/docs.php. – Заголовок з екрану.

27. How to Install and Configure PHP 7.0 [Електронний ресурс] / Режим доступу: https://www.vultr.com/docs/how-to-install-and-configure-php-70-or-php-71-on-ubuntu-16-04. – Заголовок з екрану.

28. JavaScript - Document Object Model or DOM [Електронний ресурс]/ Режим доступу:https://www.tutorialspoint.com/javascript/javascript_html_dom.htm. – Заголовок з екрану.

29. Documentation Jquery [Електронний ресурс] / Режим доступу: https://learn.jquery.com/. – Заголовок з екрану.

30. The Progressive JavaScript Framework [Електронний ресурс] / Режим доступу: https://vuejs.org/. – Заголовок з екрану.

31. Установить Linux, Apache, MySQL, PHP [Електронний ресурс] / Режим доступу: https://www.digitalocean.com/community/tutorials/linux-apache-mysql-php-lamp-ubuntu-16-04-ru– Заголовок з екрану.

32. LAMP [Електронний ресурс] / Режим доступу: https://uk.wikipedia.org/wiki/LAMP– Заголовок з екрану.

33. Official Ubuntu Documentation [Електронний ресурс] / Режим доступу: https://help.ubuntu.com/– Заголовок з екрану.

34. RussianDocumentation Ubuntu [Електронний ресурс] / Режим доступу: https://help.ubuntu.com/community/RussianDocumentation/– Заголовок з екрану.

35. Настройка веб сервера Apache через Htaccess [Електронний ресурс] / Режим доступу: http://htaccess.net.ru/– Заголовок з екрану.

36. Основы .htaccess на примерах [Електронний ресурс] / Режим доступу: https://habrahabr.ru/post/31054/– Заголовок з екрану.

37. Bootstrap [Електронний ресурс] / Режим доступу: https://getbootstrap.com/– Заголовок з екрану.

38. HTML [Електронний ресурс] / Режим доступу: http://devdocs.io/html/– Заголовок з екрану.

39. Learn HTML [Електронний ресурс] / Режим доступу: https://www.codecademy.com/learn/learn-html– Заголовок з екрану.

40. CSS Reference [Електронний ресурс] / Режим доступу: https://www.w3schools.com/cssref/default.asp– Заголовок з екрану.

Додаток А

SUMMARY

Marchenko R. V. Automation of submission of applications for subsidies and preferential allowances. – Masters-level Qualification Thesis. Sumy State University, Sumy, 2017.

The paper examines the essence of filing applications for subsidies and preferential allowances, models and forms in the context of national development priorities. The analysis of the main factors that influence the solution of this problem was carried out. The main objective of this study is to develop an automated system for submitting a subsidy application and preferential allowances.

Key words: allowances, subsidies, registration of social assistance, separate categories of the population, social protection of the population.

АНОТАЦІЯ

Марченко Р. В. Автоматизація подання заявок на отримання субсидій і пільгових надбавок. – Кваліфікаційна магістерська робота. Сумський державний університет, Суми, 2017 р.

У роботі досліджено сутність подання заявок на субсидію та пільгові надбавки, моделі та форми у контексті пріоритетів національного розвитку. Проведений аналіз основних факторів, які впливають на розв’язання даної задачі. Основною метою цього дослідження є розробка автоматизованої системи подання заявок на субсидію та пільгові надбавки.

Ключові слова: пільги, субсидія, оформлення соціальних допомог, окремим категоріям населення, соціальний захист населення .

Додаток Б

Файл Kernel.php

```php
<?php

namespace App\Console;

use Illuminate\Console\Scheduling\Schedule;
use Illuminate\Foundation\Console\Kernel as ConsoleKernel;

class Kernel extends ConsoleKernel
{
    /**
     * The Artisan commands provided by your application.
     *
     * @var array
     */
    protected $commands = [
        //
    ];

    /**
     * Define the application's command schedule.
     *
     * @param  \Illuminate\Console\Scheduling\Schedule  $schedule
     * @return void
     */
    protected function schedule(Schedule $schedule)
    {
        // $schedule->command('inspire')
        //          ->hourly();
    }

    /**
     * Register the commands for the application.
     *
     * @return void
     */
    protected function commands()
    {
        $this->load(__DIR__.'/Commands');

        require base_path('routes/console.php');
    }
}
```

Додаток В

Файл Handler.php

```php
<?php

namespace App\Exceptions;

use Exception;
use Illuminate\Foundation\Exceptions\Handler as ExceptionHandler;

class Handler extends ExceptionHandler
{
    /**
     * A list of the exception types that are not reported.
     *
     * @var array
     */
    protected $dontReport = [
        //
    ];

    /**
     * A list of the inputs that are never flashed for validation exceptions.
     *
     * @var array
     */
    protected $dontFlash = [
        'password',
        'password_confirmation',
    ];

    /**
     * Report or log an exception.
     *
     * This is a great spot to send exceptions to Sentry, Bugsnag, etc.
     *
     * @param  \Exception  $exception
     * @return void
     */
    public function report(Exception $exception)
    {
        parent::report($exception);
    }

    /**
     * Render an exception into an HTTP response.
     *
     * @param  \Illuminate\Http\Request  $request
     * @param  \Exception  $exception
     * @return \Illuminate\Http\Response
     */
    public function render($request, Exception $exception)
    {
        return parent::render($request, $exception);
    }
}
```

Додаток Г

Файл ForgotPasswordController.php

```php
<?php

namespace App\Http\Controllers\Auth;

use App\Http\Controllers\Controller;
use Illuminate\Foundation\Auth\SendsPasswordResetEmails;

class ForgotPasswordController extends Controller
{
    /*
    |--------------------------------------------------------------------------
    | Password Reset Controller
    |--------------------------------------------------------------------------
    |
    | This controller is responsible for handling password reset emails and
    | includes a trait which assists in sending these notifications from
    | your application to your users. Feel free to explore this trait.
    |
    */

    use SendsPasswordResetEmails;

    /**
     * Create a new controller instance.
     *
     * @return void
     */
    public function __construct()
    {
        $this->middleware('guest');
    }
}
```

Додаток Ґ

Файл LoginController.php

```php
<?php

namespace App\Http\Controllers\Auth;

use App\Http\Controllers\Controller;
use Illuminate\Foundation\Auth\AuthenticatesUsers;

class LoginController extends Controller
{
    /*
    |--------------------------------------------------------------------------
    | Login Controller
    |--------------------------------------------------------------------------
    |
    | This controller handles authenticating users for the application and
    | redirecting them to your home screen. The controller uses a trait
    | to conveniently provide its functionality to your applications.
    |
    */

    use AuthenticatesUsers;

    /**
     * Where to redirect users after login.
     *
     * @var string
     */
  //  protected $redirectTo = '/home';

    public function redirectTo(){
        return '/profile/'.auth()->id();
    }

    /**
     * Create a new controller instance.
     *
     * @return void
     */
    public function __construct()
    {
        $this->middleware('guest')->except('logout');
    }
}
```

Додаток Д

Файл RegisterController.php

```php
<?php

namespace App\Http\Controllers\Auth;

use App\User;
use App\Http\Controllers\Controller;
use Illuminate\Support\Facades\Validator;
use Illuminate\Foundation\Auth\RegistersUsers;

class RegisterController extends Controller
{
    /*
    |--------------------------------------------------------------------------
    | Register Controller
    |--------------------------------------------------------------------------
    |
    | This controller handles the registration of new users as well as their
    | validation and creation. By default this controller uses a trait to
    | provide this functionality without requiring any additional code.
    |
    */

    use RegistersUsers;

    /**
     * Where to redirect users after registration.
     *
     * @var string
     */
    //protected $redirectTo = '/home';
    public function redirectTo(){
        return '/profile/'.auth()->id();
    }
    /**
     * Create a new controller instance.
     *
     * @return void
     */
    public function __construct()
    {
        $this->middleware('guest');
    }

    /**
     * Get a validator for an incoming registration request.
     *
     * @param  array  $data
     * @return \Illuminate\Contracts\Validation\Validator
     */
    protected function validator(array $data)
    {
        return Validator::make($data, [
            'name' => 'required|string|max:255',
            'email' => 'required|string|email|max:255|unique:users',
            'password' => 'required|string|min:6|confirmed',
        ]);
    }
```

```
    /**
     * Create a new user instance after a valid registration.
     *
     * @param  array  $data
     * @return \App\User
     */
    protected function create(array $data)
    {
        $user =  User::create([
            'name' => $data['name'],
            'email' => $data['email'],
            'password' => bcrypt($data['password']),

        ]);

        $user->profile()->create();
        return $user;
    }

    protected function handleUserWasAuthenticated(Request $request,
$throttles)
    {
        if ($throttles) {
            $this->clearLoginAttempts($request);
        }

        if (method_exists($this, 'authenticated')) {
            return $this->authenticated($request, Auth::guard($this-
>getGuard())->user());
        }

        return redirect()->intended($this->redirectPath());
    }

}
```

Додаток Е

Файл ResetPasswordController.php

```php
<?php

namespace App\Http\Controllers\Auth;

use App\Http\Controllers\Controller;
use Illuminate\Foundation\Auth\ResetsPasswords;

class ResetPasswordController extends Controller
{
    /*
    |--------------------------------------------------------------------------
    | Password Reset Controller
    |--------------------------------------------------------------------------
    |
    | This controller is responsible for handling password reset requests
    | and uses a simple trait to include this behavior. You're free to
    | explore this trait and override any methods you wish to tweak.
    |
    */

    use ResetsPasswords;

    /**
     * Where to redirect users after resetting their password.
     *
     * @var string
     */
    protected $redirectTo = '/home';

    /**
     * Create a new controller instance.
     *
     * @return void
     */
    public function __construct()
    {
        $this->middleware('guest');
    }
}
```

Додаток Є

Файл CostController.php

```php
<?php

namespace App\Http\Controllers;

use App\Http\Requests\CostsRequest;
use Illuminate\Http\Request;
use App\Costs;
use Illuminate\Support\Facades\Auth;

class CostsController extends Controller
{
    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function index()
    {
        $costs = Costs::where('user_id','=', Auth::id())->get();
        return view('/interface.page.costs.index', compact('costs'));
    }

    /**
     * Show the form for creating a new resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function create()
    {
        return view('/interface.page.costs.create');
    }

    /**
     * Store a newly created resource in storage.
     *
     * @param  \Illuminate\Http\Request  $request
     * @return \Illuminate\Http\Response
     */
    public function store(CostsRequest $request)
    {
        $revenues = new Costs($request->all());
        $revenues->name = $request->name;
        $revenues->kind = $request->kind;
        $revenues->cost_property = $request->cost_property;
        $revenues->data_costs = $request->data_costs;
        $revenues->user_id = Auth::id();
        $revenues->save();
        return redirect('/sencs');
    }


    /**
     * Show the form for editing the specified resource.
     *
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function edit($id)
    {
```

```
        $costs = Costs::find($id);
        if(!$costs){
            abort(404);
        }
        return view('/interface.page.costs.edit', compact('costs'));
    }

    /**
     * Update the specified resource in storage.
     *
     * @param  \Illuminate\Http\Request  $request
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function update(CostsRequest $request, $id)
    {
        Costs::find($id)->update($request->all());
        return redirect('/costs');
    }

    /**
     * Remove the specified resource from storage.
     *
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function destroy($id)
    {
        $costs = Costs::find($id);
        if(!$costs){
            abort(404);
        }
        $costs->delete();
        return redirect('/costs');
    }
}
```

Додаток Ж

Файл DataUserController.php

```php
<?php

namespace App\Http\Controllers;

use App\Http\Requests\DataUsersRequest;
use App\DataUser;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;

class DataUsersController extends Controller
{
    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function index()
    {
        $datas = DataUser::where('user_id','=', Auth::id())->get();
        return view('/interface.page.data_user.index', compact('datas'));
    }

    /**
     * Show the form for creating a new resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function create()
    {
        return view('/interface.page.data_user.create');
    }

    /**
     * Store a newly created resource in storage.
     *
     * @param  \Illuminate\Http\Request  $request
     * @return \Illuminate\Http\Response
     */
    public function store(DataUsersRequest $request)
    {
        $data = new DataUser($request->all());
        $data->structure_unit = $request->structure_unit;
        $data->name = $request->name;
        $data->place_of_residence = $request->place_of_residence;
        $data->phone = $request->phone;
        $data->pasport_seria = $request->pasport_seria;
        $data->issuance_pasport = $request->issuance_pasport;
        $data->card_taxes = $request->card_taxes;
        $data->user_id = Auth::id();
        $data->save();
        return redirect('/housing_communals/create');
    }


    /**
     * Show the form for editing the specified resource.
     *
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
```

```
    public function edit($id)
    {
        $data = DataUser::find($id);
        if(!$data){
            abort(404);
        }
        return view('/interface.page.data_user.edit', compact('data'));
    }

    /**
     * Update the specified resource in storage.
     *
     * @param  \Illuminate\Http\Request  $request
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function update(DataUsersRequest $request, $id)
    {
        DataUser::find($id)->update($request->all());
        return redirect('/data_users');
    }

    /**
     * Remove the specified resource from storage.
     *
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function destroy($id)
    {
        $data = DataUser::find($id);
        if(!$data){
            abort(404);
        }
        $data->delete();
        return redirect('/data_users');
    }
}
```

Додаток З

Файл HomeController.php

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class HomeController extends Controller
{
    /**
     * Create a new controller instance.
     *
     * @return void
     */
    public function __construct()
    {
        $this->middleware('auth');
    }

    /**
     * Show the application dashboard.
     *
     * @return \Illuminate\Http\Response
     */
    public function index()
    {
        return view('home');
    }
}
```

Додаток И

Файл HousingCommunalController.php

```php
<?php

namespace App\Http\Controllers;

use App\HousingCommunal;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
use App\Http\Requests\HousingCommunalRequest;

class HousingCommunalsController extends Controller
{
    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function index()
    {
        $comunals = HousingCommunal::where('user_id','=', Auth::id())->get();
        return view('/interface.page.housing_communal.index',
compact('comunals'));
    }

    /**
     * Show the form for creating a new resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function create()
    {
        return view('/interface.page.housing_communal.create');
    }

    /**
     * Store a newly created resource in storage.
     *
     * @param  \Illuminate\Http\Request  $request
     * @return \Illuminate\Http\Response
     */
    public function store(HousingCommunalRequest $request)
    {
        $comunal = new HousingCommunal($request->all());
        $comunal->house = $request->house;
        $comunal->gas = $request->gas;
        $comunal->cold_water = $request->cold_water;
        $comunal->hot_water = $request->hot_water;
        $comunal->drainage = $request->drainage;
        $comunal->centralized_heating = $request->centralized_heating;
        $comunal->electricity_supply = $request->electricity_supply;
        $comunal->household_waste_removal = $request-
>household_waste_removal;
        $comunal->user_id = Auth::id();
        $comunal->save();
        return redirect('/revenues/create');
    }
```

```
    /**
     * Show the form for editing the specified resource.
     *
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function edit($id)
    {
        $comunal = HousingCommunal::find($id);
        if(!$comunal){
            abort(404);
        }
        return view('/interface.page.housing_communal.edit',
compact('comunal'));
    }

    /**
     * Update the specified resource in storage.
     *
     * @param  \Illuminate\Http\Request  $request
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function update(HousingCommunalRequest $request, $id)
    {
        HousingCommunal::find($id)->update($request->all());
        return redirect('/housing_communals');
    }

    /**
     * Remove the specified resource from storage.
     *
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function destroy($id)
    {
        $comunal = HousingCommunal::find($id);
        if(!$comunal){
            abort(404);
        }
        $comunal->delete();
        return redirect('/housing_communals');
    }
}
```

Додаток І

Файл ProfileController.php

```php
<?php

namespace App\Http\Controllers;

use App\Profile;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;

class ProfilesController extends Controller
{
    public  function show($id){
       $profile = Profile::with('user')->where('user_id', $id)->get();
       if(!empty($profile)){
           return view('/interface.page.profile', compact('profile'));
       }
       else{
           abort(404);
       }

    }

    public function sencs(){
        return view('/interface.page.secncs');
    }
}
```

Додаток Ї

Файл RevenuesController.php

```php
<?php

namespace App\Http\Controllers;

use App\Http\Requests\RevenuesRequvest;
use Illuminate\Http\Request;
use App\Revenues;
use Illuminate\Support\Facades\Auth;

class RevenuesController extends Controller
{
    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function index()
    {
        $revenues = Revenues::where('user_id','=', Auth::id())->get();
        return view('/interface.page.revenues.index', compact('revenues'));
    }

    /**
     * Show the form for creating a new resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function create()
    {
        return view('/interface.page.revenues.create');
    }

    /**
     * Store a newly created resource in storage.
     *
     * @param  \Illuminate\Http\Request  $request
     * @return \Illuminate\Http\Response
     */
    public function store(RevenuesRequvest $request)
    {
        $revenues = new Revenues($request->all());
        $revenues->full_name = $request->full_name;
        $revenues->birth = $request->birth;
        $revenues->card = $request->card;
        $revenues->type_of_income = $request->type_of_income;
        $revenues->amount_income = $request->amount_income;
        $revenues->source_income = $request->source_income;
        $revenues->user_id = Auth::id();
        $revenues->save();
        return redirect('/costs/create');
    }


    /**
     * Show the form for editing the specified resource.
     *
     * @param  int  $id
```

```
     * @return \Illuminate\Http\Response
     */
    public function edit($id)
    {
        $revenues = Revenues::find($id);
        if(!$revenues){
            abort(404);
        }
        return view('/interface.page.revenues.edit', compact('revenues'));
    }

    /**
     * Update the specified resource in storage.
     *
     * @param  \Illuminate\Http\Request  $request
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function update(Request $request, $id)
    {
        Revenues::find($id)->update($request->all());
        return redirect('/revenues');
    }

    /**
     * Remove the specified resource from storage.
     *
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function destroy($id)
    {
        $revenues = Revenues::find($id);
        if(!$revenues){
            abort(404);
        }
        $revenues->delete();
        return redirect('/revenues');
    }
}
```

Додаток Й

Файл Profile.php

```php
<?php

namespace App\Http\Middleware;

use Closure;


class Profile
{
    /**
     * Handle an incoming request.
     *
     * @param  \Illuminate\Http\Request  $request
     * @param  \Closure  $next
     * @return mixed
     */
    public function handle($request, Closure $next)
    {
        $article = \App\Profile::find($request->parameter('id'));

        if ($article->user_id != Auth::user()->id)
        {
            abort(404);
        }
        return $next($request);
    }
}
```

Додаток К

Файл CostsRequest.php

```php
<?php

namespace App\Http\Requests;

use Illuminate\Foundation\Http\FormRequest;

class CostsRequest extends FormRequest
{
    /**
     * Determine if the user is authorized to make this request.
     *
     * @return bool
     */
    public function authorize()
    {
        return true;
    }

    /**
     * Get the validation rules that apply to the request.
     *
     * @return array
     */
    public function rules()
    {
        return [
            'name' => 'required',
            'kind' => 'required',
            'cost_property' => 'required',
            'data_costs' => 'required',
        ];
    }
}
```

Додаток Л

Файл DataUsersRequest.php

```php
<?php

namespace App\Http\Requests;

use Illuminate\Foundation\Http\FormRequest;

class DataUsersRequest extends FormRequest
{
    /**
     * Determine if the user is authorized to make this request.
     *
     * @return bool
     */
    public function authorize()
    {
        return true;
    }

    /**
     * Get the validation rules that apply to the request.
     *
     * @return array
     */
    public function rules()
    {
        return [
            'structure_unit' => 'required',
            'name' => 'required',
            'place_of_residence' => 'required',
            'phone' => 'required',
            'pasport_seria' => 'required',
            'issuance_pasport' => 'required',
            'card_taxes' => 'required',
        ];
    }
}
```

Додаток М

Файл HousingCommunalRequest.php

```php
<?php

namespace App\Http\Requests;

use Illuminate\Foundation\Http\FormRequest;

class HousingCommunalRequest extends FormRequest
{
    /**
     * Determine if the user is authorized to make this request.
     *
     * @return bool
     */
    public function authorize()
    {
        return true;
    }

    /**
     * Get the validation rules that apply to the request.
     *
     * @return array
     */
    public function rules()
    {
        return [
            'house' => 'required',
            'gas' => 'required',
            'cold_water' => 'required',
            'hot_water' => 'required',
            'drainage' => 'required',
            'centralized_heating' => 'required',
            'electricity_supply' => 'required',
            'household_waste_removal' => 'required',
        ];
    }
}
```

Додаток Н

Файл RevenuesRequvest.php

```php
<?php

namespace App\Http\Requests;

use Illuminate\Foundation\Http\FormRequest;

class RevenuesRequvest extends FormRequest
{
    /**
     * Determine if the user is authorized to make this request.
     *
     * @return bool
     */
    public function authorize()
    {
        return true;
    }

    /**
     * Get the validation rules that apply to the request.
     *
     * @return array
     */
    public function rules()
    {
        return [
            'full_name' => 'required',
            'birth' => 'required',
            'card' => 'required',
            'type_of_income' => 'required',
            'amount_income' => 'required',
            'source_income' => 'required',
        ];
    }
}
```

Додаток О

Файл AuthServiceProvider.php

```php
<?php

namespace App\Providers;

use Illuminate\Support\Facades\Gate;
use Illuminate\Foundation\Support\Providers\AuthServiceProvider as
ServiceProvider;

class AuthServiceProvider extends ServiceProvider
{
    /**
     * The policy mappings for the application.
     *
     * @var array
     */
    protected $policies = [
        'App\Model' => 'App\Policies\ModelPolicy',
    ];

    /**
     * Register any authentication / authorization services.
     *
     * @return void
     */
    public function boot()
    {
        $this->registerPolicies();

        //
    }
}
```

Додаток П

Файл RouteServiceProvider.php

```php
<?php

namespace App\Providers;

use Illuminate\Support\Facades\Route;
use Illuminate\Foundation\Support\Providers\RouteServiceProvider as
ServiceProvider;

class RouteServiceProvider extends ServiceProvider
{
    /**
     * This namespace is applied to your controller routes.
     *
     * In addition, it is set as the URL generator's root namespace.
     *
     * @var string
     */
    protected $namespace = 'App\Http\Controllers';

    /**
     * Define your route model bindings, pattern filters, etc.
     *
     * @return void
     */
    public function boot()
    {
        //

        parent::boot();
    }

    /**
     * Define the routes for the application.
     *
     * @return void
     */
    public function map()
    {
        $this->mapApiRoutes();

        $this->mapWebRoutes();

        //
    }

    /**
     * Define the "web" routes for the application.
     *
     * These routes all receive session state, CSRF protection, etc.
     *
     * @return void
     */
    protected function mapWebRoutes()
    {
        Route::middleware('web')
             ->namespace($this->namespace)
             ->group(base_path('routes/web.php'));
```

```
    }

    /**
     * Define the "api" routes for the application.
     *
     * These routes are typically stateless.
     *
     * @return void
     */
    protected function mapApiRoutes()
    {
        Route::prefix('api')
             ->middleware('api')
             ->namespace($this->namespace)
             ->group(base_path('routes/api.php'));
    }
}
```

Додаток Р

Файл Costs.php

```php
<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Costs extends Model
{
    protected $fillable = [
        'name',
        'kind',
        'cost_property',
        'data_costs',
        'user_id',
    ];
}
```

Додаток С

Файл DataUser.php

```php
<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class DataUser extends Model
{
    protected $fillable = [
        'structure_unit',
        'name',
        'place_of_residence',
        'phone',
        'pasport_seria',
        'issuance_pasport',
        'card_taxes',
        'user_id',
    ];
}
```

Додаток Т

Файл HousingCommunal.php

```php
<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class HousingCommunal extends Model
{
    protected $fillable = [
        'house',
        'gas',
        'cold_water',
        'hot_water',
        'drainage',
        'centralized_heating',
        'electricity_supply',
        'household_waste_removal',
        'user_id',
    ];
}
```

Додаток У

Файл Profile.php

```php
<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Profile extends Model
{
    protected $fillable = [
        'user_id',
    ];

    public function user()
    {
        return $this->belongsTo(User::class, 'user_id');
    }
}
```

Додаток Ф

Файл Revenues.php

```php
<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Revenues extends Model
{
    protected $table = "revenues";
    protected $fillable = [
        'full_name',
        'birth',
        'card',
        'type_of_income',
        'amount_income',
        'source_income',
        'user_id',
    ];
}
```

Додаток Х

Файл User.php

```php
<?php

namespace App;

use Illuminate\Notifications\Notifiable;
use Illuminate\Foundation\Auth\User as Authenticatable;

class User extends Authenticatable
{
    use Notifiable;

    /**
     * The attributes that are mass assignable.
     *
     * @var array
     */
    protected $fillable = [
        'name', 'email', 'password',
    ];

    /**
     * The attributes that should be hidden for arrays.
     *
     * @var array
     */
    protected $hidden = [
        'password', 'remember_token',
    ];


    public function profile()
    {
        return $this->hasOne(Profile::class, 'user_id', 'id');
    }
}
```

Додаток Ц

Файл CreateUsersTable.php

```php
<?php

use Illuminate\Support\Facades\Schema;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreateUsersTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('users', function (Blueprint $table) {
            $table->increments('id');
            $table->string('name');
            $table->string('email')->unique();
            $table->string('password');
            $table->rememberToken();
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('users');
    }
}
```

Додаток Ч

Файл CreatePasswordResetsTable.php

```php
<?php

use Illuminate\Support\Facades\Schema;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreatePasswordResetsTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('password_resets', function (Blueprint $table) {
            $table->string('email')->index();
            $table->string('token');
            $table->timestamp('created_at')->nullable();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('password_resets');
    }
}
```

Додаток Ш

Файл CreateProfilesTable.php

```php
<?php

use Illuminate\Support\Facades\Schema;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreateProfilesTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('profiles', function (Blueprint $table) {
            $table->increments('id');
            $table->integer('user_id');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('profiles');
    }
}
```

Додаток Щ

Файл CreateDataUsersTable.php

```php
<?php

use Illuminate\Support\Facades\Schema;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreateDataUsersTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('data_users', function (Blueprint $table) {
            $table->increments('id');
            $table->string('structure_unit');
            $table->string('name');
            $table->string('place_of_residence');
            $table->string('phone');
            $table->string('pasport_seria');
            $table->string('issuance_pasport');
            $table->string('card_taxes');
            $table->integer('user_id');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('data_users');
    }
}
```

Додаток Ю

Файл CreateHousingCommunalsTable.php

```php
<?php

use Illuminate\Support\Facades\Schema;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreateHousingCommunalsTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('housing_communals', function (Blueprint $table) {
            $table->increments('id');
            $table->string('house');
            $table->string('gas');
            $table->string('cold_water');
            $table->string('hot_water');
            $table->string('drainage');
            $table->string('centralized_heating');
            $table->string('electricity_supply');
            $table->string('household_waste_removal');
            $table->integer('user_id');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('housing_communals');
    }
}
```

Додаток Я

Файл CreateRevenuesTable.php

```php
<?php

use Illuminate\Support\Facades\Schema;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreateRevenuesTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('revenues', function (Blueprint $table) {
            $table->increments('id');
            $table->string('full_name');
            $table->string('birth');
            $table->string('card');
            $table->string('type_of_income');
            $table->string('amount_income');
            $table->string('source_income');
            $table->integer('user_id');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('revenues');
    }
}
```

Додаток АБ

Файл CreateCostsTable.php

```php
<?php

use Illuminate\Support\Facades\Schema;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreateCostsTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('costs', function (Blueprint $table) {
            $table->increments('id');
            $table->string('name');
            $table->string('kind');
            $table->string('cost_property');
            $table->string('data_costs');
            $table->integer('user_id');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('costs');
    }
}
```

Додаток АВ

Файл style.css

```
header.main-header .header-container {
    margin: 0 auto;
    padding: 60px 0px;
    background-color: rgba(8, 134, 196, 0.96);
    font-family: "PTSansCaptionBold";
    background-image: url(../image/crest.png);
    background-repeat: no-repeat;
    background-position: 75px 82px;
}

header.main-header .text {
    padding-left: 190px;
    text-shadow: 0 -2px 3px rgba(0, 3, 1, 0.64);
    color: #fff;
}

.show-title{
    text-align: center;
}

.name-profile{
    text-transform: uppercase;
}

.title-end{
    text-align: center;
}

.mini-tite{
    text-align: center;
}

.title-index{
    text-align: center;
    margin-bottom: 40px;
}

.link-style a{
    font-size: 30px;
}

.link-style{
    margin-top: 40px;
}

.new_footer {
    margin-top: 40px;
    width: 100%;
    height: 80px;
    background: #0075aa;
}

.footer_logo {
    text-align: center;
    padding-top: 1%;
}
```

```
.footer_logo_img {
    margin: auto;
}

.navbar-default {
    background-color: #0075aa;
    border-color: #e7e7e7;
}

.navbar-default .navbar-brand{
    color: #ffffff;
}

.navbar-default .navbar-brand:hover {
    color: #e7e7e7;
    background-color: transparent;
}

.navbar-default .navbar-nav>li>a {
    color: #ffffff;
}

.navbar-default .navbar-nav>li>a:hover {
    color: #e7e7e7;
}
```

Додаток АГ

Файл index.php

```php
<?php

/**
 *
 * @package  Laravel
 * @author   Taylor Otwell <taylor@laravel.com>
 */

define('LARAVEL_START', microtime(true));

/*
|--------------------------------------------------------------------------
| Register The Auto Loader
|--------------------------------------------------------------------------
|
| Composer provides a convenient, automatically generated class loader for
| our application. We just need to utilize it! We'll simply require it
| into the script here so that we don't have to worry about manual
| loading any of our classes later on. It feels great to relax.
|
*/

require __DIR__.'/../vendor/autoload.php';

/*
|--------------------------------------------------------------------------
| Turn On The Lights
|--------------------------------------------------------------------------
|
| We need to illuminate PHP development, so let us turn on the lights.
| This bootstraps the framework and gets it ready for use, then it
| will load up this application so that we can run it and send
| the responses back to the browser and delight our users.
|
*/

$app = require_once __DIR__.'/../bootstrap/app.php';

/*
|--------------------------------------------------------------------------
| Run The Application
|--------------------------------------------------------------------------
|
| Once we have the application, we can handle the incoming request
| through the kernel, and send the associated response back to
| the client's browser allowing them to enjoy the creative
| and wonderful application we have prepared for them.
|
*/

$kernel = $app->make(Illuminate\Contracts\Http\Kernel::class);

$response = $kernel->handle(
    $request = Illuminate\Http\Request::capture()
);

$response->send();
```

```
$kernel->terminate($request, $response);
```

Додаток АҐ

Файл bootstrap.js

```
window._ = require('lodash');

/**
 * We'll load jQuery and the Bootstrap jQuery plugin which provides support
 * for JavaScript based Bootstrap features such as modals and tabs. This
 * code may be modified to fit the specific needs of your application.
 */

try {
    window.$ = window.jQuery = require('jquery');

    require('bootstrap-sass');
} catch (e) {}

/**
 * We'll load the axios HTTP library which allows us to easily issue requests
 * to our Laravel back-end. This library automatically handles sending the
 * CSRF token as a header based on the value of the "XSRF" token cookie.
 */

window.axios = require('axios');

window.axios.defaults.headers.common['X-Requested-With'] = 'XMLHttpRequest';

/**
 * Next we will register the CSRF Token as a common header with Axios so that
 * all outgoing HTTP requests automatically have it attached. This is just
 * a simple convenience so we don't have to attach every token manually.
 */

let token = document.head.querySelector('meta[name="csrf-token"]');

if (token) {
    window.axios.defaults.headers.common['X-CSRF-TOKEN'] = token.content;
} else {
    console.error('CSRF token not found: https://laravel.com/docs/csrf#csrf-
x-csrf-token');
}

/**
 * Echo exposes an expressive API for subscribing to channels and listening
 * for events that are broadcast by Laravel. Echo and event broadcasting
 * allows your team to easily build robust real-time web applications.
 */

// import Echo from 'laravel-echo'

// window.Pusher = require('pusher-js');

// window.Echo = new Echo({
//     broadcaster: 'pusher',
//     key: 'your-pusher-key',
//     cluster: 'mt1',
```

```
//     encrypted: true
// });
```

Додаток АД

Файл master.blade.php

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <meta name="description" content="">
    <meta name="author" content="">
    {{--<title>@yield('title')</title>--}}
    <title>DIPLOM</title>
    @include('interface.css')
    @yield('css')
</head>
<body>
@include('interface.nav')

@yield('content')

@include('interface.footer')
@include('interface.scripts')
@yield('scripts')
</body>
</html>
```

Додаток АЕ

Файл master.index.blade.php

```
@extends('interface.master_profile')
@section('content')
    <div class="container">
    <h1 class="title-index"> Записи що до І розділу, продовження </h1>
    <div class="container">
        <table class="table">
            <tbody class="index-body">
            @foreach($comunals as $comunal)
                <tr>
                    <td>{{$comunal->id}}</td>
                    <td>{{$comunal->house}}</td>
                    <td>{{$comunal->gas}}</td>
                    <td>{{$comunal->cold_water}}</td>
                    <td>{{$comunal->hot_water}}</td>
                    <td>{{$comunal->drainage}}</td>
                    <td>{{$comunal->centralized_heating}}</td>
                    <td>{{$comunal->electricity_supply}}</td>
                    <td>{{$comunal->household_waste_removal}}</td>
                    <td class="display-margin">
                        <a class="btn btn-small btn-info" href="{{
URL::to('housing_communals/' . $comunal->id . '/edit') }}">Edit</a>
                        {{ Form::open(array('url' => 'housing_communals/' .
$comunal->id)) }}
                        {{ Form::hidden('_method', 'DELETE') }}
                        {{ Form::submit('Delete', array('class' => 'btn btn-
small del btn-danger')) }}
                        {{ Form::close() }}
                    </td>
                </tr>
            @endforeach

            </tbody>
        </table>
    </div>
    </div>
@endsection
```

Додаток АЄ

Файл master.edit.blade.php

```
@extends('interface.master_profile')
@include('interface.errors')
@section('content')
    <form method="post" action="{{
action('HousingCommunalsController@update', $comunal->id) }}" method="POST"
enctype="multipart/form-data">
        {!! method_field('patch') !!}
        {{csrf_field()}}
        <div class="container">
            <h3>Продовження розділу I:</h3>
            <div>
                <div class="form-group">
                    <label>Утримання будинків і споруд та прибудинкових
територій</label>
                    <input type="text" value="{{$comunal->house}}"
class="form-control" name="house"
                           placeholder="Утримання будинків і споруд ..."
required>
                </div>
                <div class="form-group">
                    <label>Газопостачання</label>
                    <input type="text" value="{{$comunal->gas}}" class="form-
control" name="gas" placeholder="Газопостачання" required>
                </div>
                <div class="form-group">
                    <label>Централізоване постачання холодної води</label>
                    <input class="form-control" value="{{$comunal-
>cold_water}}" name="cold_water" placeholder="Централізоване постачання
холодної води" required>
                </div>
                <div class="form-group">
                    <label>Централізоване постачання гарячої води</label>
                    <input class="form-control" value="{{$comunal-
>hot_water}}" name="hot_water" placeholder="Централізоване постачання гарячої
води" required>
                </div>
                <div class="form-group">
                    <label>Водовідведення</label>
                    <input class="form-control" value="{{$comunal-
>drainage}}" name="drainage" placeholder="Водовідведення" required>
                </div>
                <div class="form-group">
                    <label>Централізоване опалення</label>
                    <input class="form-control" value="{{$comunal-
>centralized_heating}}" name="centralized_heating"
placeholder="Централізоване опалення" required>
                </div>
                <div class="form-group">
                    <label>Електропостачання</label>
                    <input class="form-control" value="{{$comunal-
>electricity_supply}}" name="electricity_supply"
placeholder="Електропостачання" required>
                </div>
                <div class="form-group">
                    <label>Вивезення побутових відходів</label>
                    <input class="form-control" value="{{$comunal-
>household_waste_removal}}" name="household_waste_removal"
```

```
placeholder="Вивезення побутових відходів" required>
                </div>
                <button type="submit" style="float: left;" class="btn btn-
default right">Submit</button>
            </div>
```

Додаток АЖ

Файл create.blade.php

```
@extends('interface.master_profile')
@include('interface.errors')
@section('content')
    <form method="post" action="{{ action('HousingCommunalsController@store')
}}" enctype="multipart/form-data">
        {{ csrf field() }}
        <div class="container">
            <h3>Продовження розділу I:</h3>
            <div>
                <p>Відомості про житлово-комунальні послуги, якими
користуються особи, які
                    зареєстровані (фактично проживають) у житловому
приміщенні/будинку
                </p>
            </div>
            <div>
                <div class="form-group">
                    <label>Утримання будинків і споруд та прибудинкових
територій</label>
                    <input type="text" class="form-control" name="house"
                           placeholder="Утримання будинків і споруд ..."
required>
                </div>
                <div class="form-group">
                    <label>Газопостачання</label>
                    <input type="text" class="form-control" name="gas"
placeholder="Газопостачання" required>
                </div>
                <div class="form-group">
                    <label>Централізоване постачання холодної води</label>
                    <input class="form-control" name="cold_water"
placeholder="Централізоване постачання холодної води" required>
                </div>
                <div class="form-group">
                    <label>Централізоване постачання гарячої води</label>
                    <input class="form-control" name="hot_water"
placeholder="Централізоване постачання гарячої води" required>
                </div>
                <div class="form-group">
                    <label>Водовідведення</label>
                    <input class="form-control" name="drainage"
placeholder="Водовідведення" required>
                </div>
                <div class="form-group">
                    <label>Централізоване опалення</label>
                    <input class="form-control" name="centralized_heating"
placeholder="Централізоване опалення" required>
                </div>
                <div class="form-group">
                    <label>Електропостачання</label>
                    <input class="form-control" name="electricity_supply"
placeholder="Електропостачання" required>
                </div>
                <div class="form-group">
                    <label>Вивезення побутових відходів</label>
```

```
                    <input class="form-control"
name="household_waste_removal" placeholder="Вивезення побутових відходів"
required>
                </div>
                <div>
                    <p>
                        У разі коли прийняття рішення щодо моєї заяви
потребує окремого рішення
                        місцевих органів виконавчої влади/місцевого
самоврядування або утвореної ними
                        комісії, прошу розглянути/не розглядати мою заяву
відповідними органами або
                        утвореною ними комісією (необхідне підкреслити).

                    </p>
                </div>
                <button type="submit" style="float: left;" class="btn btn-
default right">Відправити</button>
            </div>
        </div>
    </form>
@endsection
```

Додаток А3

Файл web.php

```php
<?php

/*
|--------------------------------------------------------------------------
| Web Routes
|--------------------------------------------------------------------------
|
| Here is where you can register web routes for your application. These
| routes are loaded by the RouteServiceProvider within a group which
| contains the "web" middleware group. Now create something great!
|
*/

Route::get('/', 'AppController@home');

Auth::routes();

Route::get('/home', 'HomeController@index')->name('home');
Route::get('/sencs', 'ProfilesController@sencs')->name('sencs');
Route::get('/profile/{id}', 'ProfilesController@show')->middleware('auth');
//Route::resource('data users', 'DataUsersController')->middleware('auth');
Route::resource('data_users', 'DataUsersController', ['except' => [
    'show'
]])->middleware('auth');

Route::resource('housing_communals', 'HousingCommunalsController', ['except'
=> [
    'show'
]])->middleware('auth');

Route::resource('revenues', 'RevenuesController', ['except' => [
    'show'
]])->middleware('auth');

Route::resource('costs', 'CostsController', ['except' => [
    'show'
]])->middleware('auth');
```

Додаток АИ

Файл .env

```
APP_NAME=Laravel
APP_ENV=local
APP_KEY=base64:Zk4Fc1CzIu1XPiFAoSQ7g4FZyU+xrrsNK1VrsNYFk4c=
APP_DEBUG=true
APP_LOG_LEVEL=debug
APP_URL=http://localhost

DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=diplom
DB_USERNAME=root
DB_PASSWORD=

BROADCAST_DRIVER=log
CACHE_DRIVER=file
SESSION_DRIVER=file
SESSION_LIFETIME=120
QUEUE_DRIVER=sync

REDIS_HOST=127.0.0.1
REDIS_PASSWORD=null
REDIS_PORT=6379

MAIL_DRIVER=smtp
MAIL_HOST=smtp.mailtrap.io
MAIL_PORT=2525
MAIL_USERNAME=null
MAIL_PASSWORD=null
MAIL_ENCRYPTION=null

PUSHER_APP_ID=
PUSHER_APP_KEY=
PUSHER_APP_SECRET=
PUSHER_APP_CLUSTER=mt1
```