МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ

Факультет комп'ютерних наук,
управління та адміністрування
Кафедра інформаційних технологій

Бакалаврська кваліфікаційна робота

на тему: Розробка програмного додатку для планування дитячого раціону

Виконав студент 4 курсу групи К-45
Напрям 122 комп'ютерні науки,
Тарчинська Юлія Вячеславівна
_____

Керівник асистент
Шуптар-Пориваєва Наталія Йосипівна

Консультант д.х.н., професор
Кругляк Юрій Олексійович

Рецензент к.геогр.н., доцент
Бургаз Олексій Анатолійович

Одеса 2019

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

БД – база даних

ОС – операційна система

ПЗ – програмне забезпечення

СКБД – Система керування базами даних

СНД – Співдружність Незалежних Держав

API – Application Programming Interface, Прикладний програмний інтерфейс

CSS – Cascading Style Sheets, Каскадні таблиці стилів

DFD – Data Flow Diagram, опис елементарних процесів, потоків даних, сховищ даних і зовнішніх сутностей

DOM – Document Object Model, Об'єктна модель документа

HTML – Hypertext Markup Language, Мова розмітки гіпертекстових документів

HTTP – HyperText Transfer Protocol, протокол передачі гіпертексту

IDEF0 – Integration Definition for Function Modeling, методологія функціонального моделювання

ISAM – Indexed Sequential Access Method – індексно-послідовний метод доступу

ISO – International Organization for Standardization, Міжнародна організація зі стандартизації

MVC – Model-view-controller, Модель – вигляд – контролер

NTFS – New Technology File System, файлова система нової технології

PHP – Hypertext Preprocessor, гіпертекстовий препроцесор

SADT – Structured Analysis and Design Teqnique, метод структурного аналізу й техніка проектування моделі системи за допомогою функціональних діаграм

SQL – Structured Query Language, мова структурованих запитів

SVG – Scalable Vector Graphics, масштабована векторна графіка

UML – Unified Modeling Language, уніфікована мова моделювання

XHTML – Extensible Hypertext Markup Language, Розширювана мова розмітки гіпертексту

XML – Extensible Markup Language, Розширювана мова розмітки

XUL – User Interface Language – мова розмітки для створення графічних інтерфейсів кристувача

ВСТУП

У наш час батьки вже не уявляють своє життя без дитячих садків. Адже куди ж діти цей маленький скарб, щоб можно було спокійно піти на роботу і не перейматися, що з дитиною щось трапиться. Ніхто навіть і не думає залишати їх самих вдома.

Дитсадок – сучасний помічник працюючим батькам. Тут і нагодують, і навчать новому, і спати вкладуть, і погарти з різними іграшками дадуть, ще й перші знайомі та друзі з'являться. Для цього ж звісно є спеціально навчені вихователі та персонал, котрий піклується про чистоту, їжу та безпеку маленьких бешкетників.

Чи міркували ви про те, скільки всього потрібно для створення звичайного дитсадку? Взяти хоча б питання харчування. Чи здогадувалися ви скільки часу та сил витрачають кухарі, щоб скласти меню хоча б на тиждень? Скільки ще витрачається часу кожного дня, аби порахувати скільки дітей сьогодні будуть їсти, чи вистачить продуктів на складі та яку саме страву сьогодні приготувати?

І все це робиться на аркушах паперу: усі рецепти записуються на них, список, привезених сьогодні, продуктів, списки наявних продуктів та розклад харчування на тиждень. Але ж папір так легко загубити, розлити якусь рідину на нього, випадково викинути у смітник. Уявіть як було би набагато легше, якщо усе це було би у електронному вигляді. Коли усі рецепти вже є, потрібно тільки натиснути на потрібний і відкриється одразу список потрібних продуктів, ще й кількість наявних покаже.

Тому метою дипломної роботи «Розробка програмного додатку для планування дитячого раціону» є:

– створення додатку для вибору раціону;
– заповнення вибору страв;
– реалізація можливості розрахунку кількості порцій;

– реалізація складу для збереження продуктів;
– реалізація можливості порівняння потрібних продуктів та наявних.

Розробка даної системи буде включати в себе кілька областей, такі як:

– аналіз області;
– розробка бази даних;
– написання коду програми;
– реалізація інтерфейсу.

Результатом роботи буде готовий додаток, яким можна користуватися у всіх дитсадках.

1 АНАЛІТИЧНА ЧАСТИНА

В даному розділі проводиться перший етап розробки програмного додатку «Дитячий раціон», а саме розбір предметної області, виявлення актуальності та постановки концептуальних питань щодо структури системи та їх реалізації.

1.1 Аналіз предметної області

Сучасний темп життя не дозволяє багатьом мамам і татам залишатися вдома та виховувати своє чадо самостійно. Тому дитячі садки стали невід'ємною частиною нашого світу.

Часом стає цікаво, хто ж спочатку придумав такий важливий проект для дітей?

Перша колективна навчальна робота з маленькими дітьми виникла в Європі. Спроба створити установу для роботи з дітьми була зроблена Р.Оуеном в 1802 році в Шотландії. Але саме визначення «дитячий сад» введено в обіг Фрідріхом Фребелем, відомим німецьким педагогом, який втілив свою блискучу ідею, перебуваючи на 60-му році життя.

Знайомство Фребеля з директором освітньої школи А.Г.Грунерит, послідовником Йогана Пестолацціі – засновника системи розвивального навчання дитини, зародило у Фрідріха надзвичайну любов до педагогіки.

Все життя Фребель присвятив вивченню цієї цікавої науки і вдосконалення своїх знань в ній. Ідеалістичні ідеї педагога, що сформувалися під впливом німецької філософії, припускали концепцію саморозвитку людини. Згідно з принципами Фребеля, дитина від природи наділена багатьма талантами, які при вмілому і грамотному підході можна і потрібно розвивати. Так, малюк в обов'язковому порядку повинен отримати дошкільну підготовку, щоб виявити свої виключні індивідуальні природні дані і почати їх вдосконалювати.

Діти, на думку Фребеля, є Божими рослинами, квітами і основне завдання вчителя, як працівника саду, вирощувати їх з особливою любов'ю. Звідси назва такої установи для освітньої роботи з дітьми – «Kindergarten» – перекладається як «дитячий сад».

Саме Фрідріхом Фребелем створені основні принципи функціонування дитячого садка і вироблена методика підготовки дітей, заснована на теорії гри. Творча інтерпретація ідеї Йогана Пестолацці успішно реалізувалася на практиці першим засновником дитячого садка.

Відвідувати такий дитячий сад могли навіть діти бідних сімей. Дітей годували 3 рази в день і займали різними розвиваючими заняттями. Але плати, встановленої за перебування в саду, виявилося недостатньо для утримання подібного закладу, Фрідріх платив всім вихователям зі своєї кишені. Через два роки дитячий сад закрився. Але чутки про подібний досвід поширилися по світу, і в різних частинах стали з'являтися нові установи, що спеціалізуються на дошкільну освіту.

Перші дитячі садки відкрилися в Росії через 30 років першого досвіду Фрідріха Фребеля. З'явилися вони в містах Семиградську і Петербурзі. На той момент такі установи були платними і досить дорогими, їх могли дозволити собі тільки люди, що володіють значними фінансовими можливостями. У дитячі установи приймалися діти від 3 до 8 років, з ними займалися досвідчені педагоги, які шляхом ігрових методик розвивали здібності дітей і готували до вступу в школу.

За часів бурхливого розвитку дитячих садків, вийшов журнал «Дитячий сад», видавцем якого стала А.С.Симанович – засновниця четвертого дитячого садка в Петербурзі. У журналі докладно викладалися нові принципи роботи з дітьми перед школою.

Перший абсолютно безкоштовний дитячий сад відкрився в 1866 році при благодійної організації, де дітей навчали малюванню, макраме, популярним технікам ручної роботи, молитвам і багатьом іншим напрямам. Тут же було організовано швейне виробництво дитячого одягу, кухня і школа. Поді-

бний заклад проіснував недовго в зв'язку з недостатнім фінансуванням і незабаром зачинилося, як це сталося і з дитячим садом Фребеля.

Але прорив в галузі дошкільної освіти почав давати свої плоди: через тридцять років в Росії стало з'являтися безліч дитячих садків для різних верств населення. Кожен батько міг отримати місце для своєї дитини відповідно до своїх фінансових можливостей. З тих пір велика увага стала приділятися освіті педагогів, які займаються розвитком малюків. Вихователі відвідували безліч курсів на дошкільну тематику і вдосконалювали свої навички роботи з дітьми.

Пізніше на основі досвіду створення дитячих садків стали з'являтися ясла, де містилися діти, починаючи з 2-х місяців.

На початку 60-х років XX століття була сформована єдина система роботи дитячих садків, закріплена нормативними актами, яка діє по сьогоднішній день, зазнаючи деякі трансформації через зміну різних зовнішніх факторів.

В наші дні дитсадками користуються майже всі батьки і так як вони люблять своїх дітей, кожного з них турбує безпека та ситість їх квіточок[1][1)].

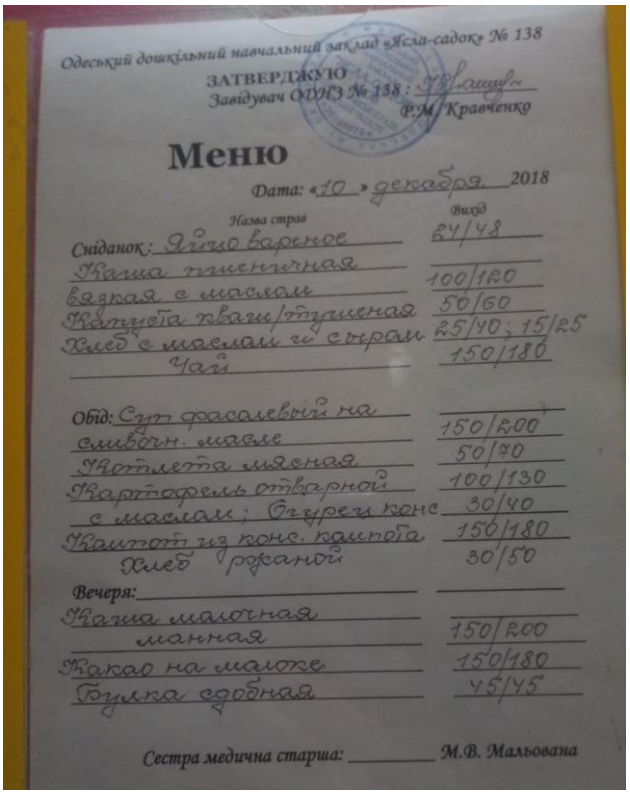
1.2 Огляд аналогів

Дивовижно, але програмних аналогів даному проекту ще досі немає. Тому, це стає дуже великою перевагою для створення цього продукту.

Все, що є зараз, так це звичайні шматки аркушів, на яких кожен день складають меню.

Приклади такого меню зображені на рис. 1-2:

[1)] [1] История появления детских садов. URL: http://lubopitnie.ru/istoriya–poyavleniya–detskih–sadov/. (дата звернення 25.03.2019)



Одеський дошкільний навчальний заклад «Ясла-садок» № 138

ЗАТВЕРДЖУЮ
Завідувач ОДНЗ № 138: _____
Р.М. Кравченко

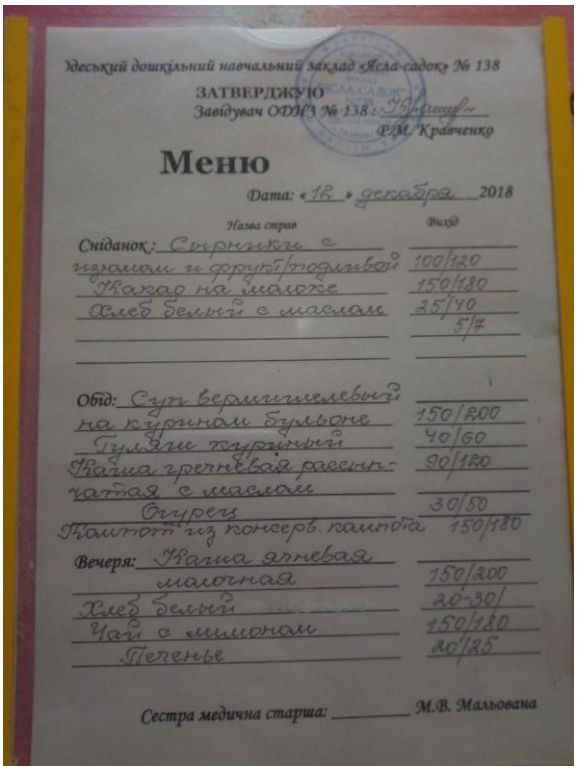
Меню

Дата: «10» декабря 2018

Назва страв	Вихід
Сніданок: Яйцо вареное	24/48
Каша пшеничная вязкая с маслом	100/120
Капуста кваш/тушеная	50/60
Хлеб с маслом и сыром	25/40; 15/25
Чай	150/180
Обід: Суп фасолевый на смывочн. масле	150/200
Котлета мясная	50/70
Картофель отварной с маслом; Огурец конс	100/130
Компот из конс. плодов	30/40
Хлеб ржаной	150/180
Вечеря:	30/50
Каша молочная манная	150/200
Какао на молоке	150/180
Булка сдобная	45/45

Сестра медична старша: _____ М.В. Мальована

Рисунок 1 – Написане вручну меню на день в дитячому садку



Одеський дошкільний навчальний заклад «Ясла-садок» № 138

ЗАТВЕРДЖУЮ
Завідувач ОДНЗ № 138: _____
Р.М. Кравченко

Меню

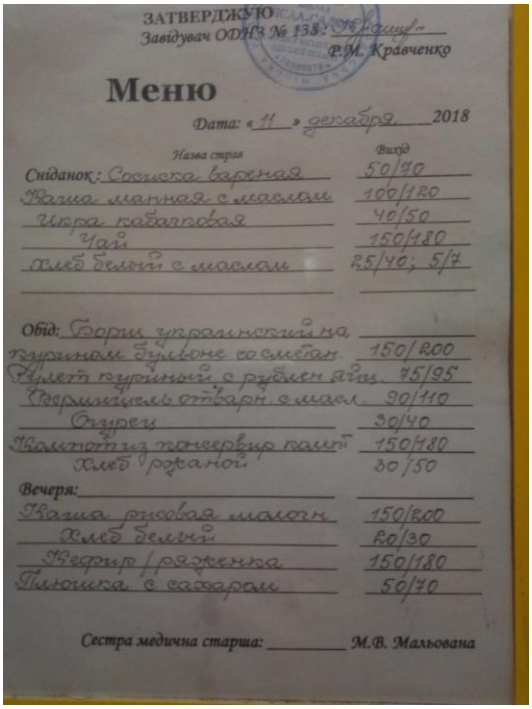
Дата: «18» декабря 2018

Назва страв	Вихід
Сніданок: Сырники с изюмом и фрукт/подливой	100/120
Какао на молоке	150/180
Хлеб белый с маслом	25/40
	5/7
Обід: Суп вермишелевый на курином бульоне	150/200
Тушеи куриный	40/60
Каша гречневая рассыпчатая с маслом	90/120
Огурец	30/50
Компот из консерв. плодов	150/180
Вечеря: Каша ячневая молочная	150/200
Хлеб белый	20-30/
Чай с лимоном	150/180
Печенье	20/25

Сестра медична старша: _____ М.В. Мальована

Рисунок 2 – Написане вручну меню на день в дитячому садку

Як ми бачимо, кожного дня пишеться нове меню (рис. 3-4):



ЗАТВЕРДЖУЮ
Завідувач ОДНЗ № 138: _____
Р.М. Кравченко

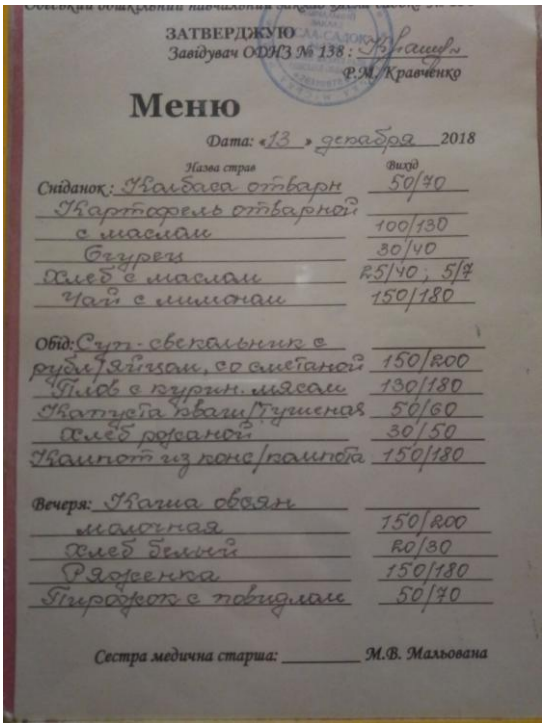
Меню

Дата: «11» декабря 2018

Назва страв	Вихід
Сніданок: Сосиска варёная	50/90
Каша манная с маслом	100/180
Икра кабачковая	40/50
Чай	150/180
Хлеб белый с маслом	25/40; 5/7
Обід: Борщ украинский на курином бульоне со сметан.	150/200
Рулет куриный с рубленым яйц.	75/95
Вермишель отварн. с масл.	90/110
Огурец	30/40
Компот из консерв. плодов	150/180
Хлеб ржаной	30/50
Вечеря:	
Каша рисовая молочн.	150/200
Хлеб белый	20/30
Кефир / ряженка	150/180
Пирожок с сахаром	50/70

Сестра медична старша: _____ М.В. Мальована

Рисунок 3 – Написане вручну меню на день в дитячому садку



ЗАТВЕРДЖУЮ
Завідувач ОДНЗ № 138: _____
Р.М. Кравченко

Меню

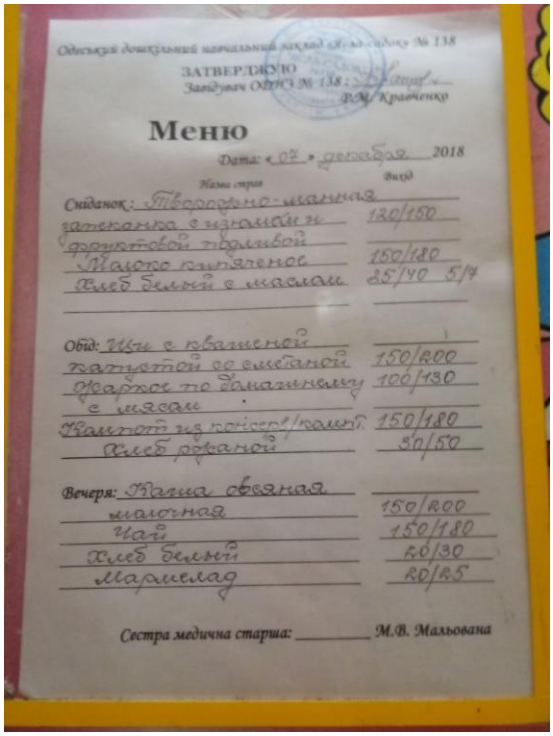
Дата: «13» декабря 2018

Назва страв	Вихід
Сніданок: Колбаса отварн	50/70
Картофель отварной с маслом	100/130
Огурец	30/40
Хлеб с маслом	25/40; 5/7
Чай с лимоном	150/180
Обід: Суп-свекольник с рубленым яйцом, со сметаной	150/200
Плов с курин. мясом	130/180
Капуста кваш/тушеная	50/60
Хлеб ржаной	30/50
Компот из конс/плодов	150/180
Вечеря: Каша овсян молочная	150/200
Хлеб белый	20/30
Ряженка	150/180
Пирожок с повидлом	50/70

Сестра медична старша: _____ М.В. Мальована

Рисунок 4 – Написане вручну меню на день в дитячому садку

Нижче зображені ще приклади (рис.5-6):



Одеський дошкільний навчальний заклад «Ясла-садок» № 138

ЗАТВЕРДЖУЮ
Завідувач ОДНЗ № 138: _____
Р.М. Кравченко

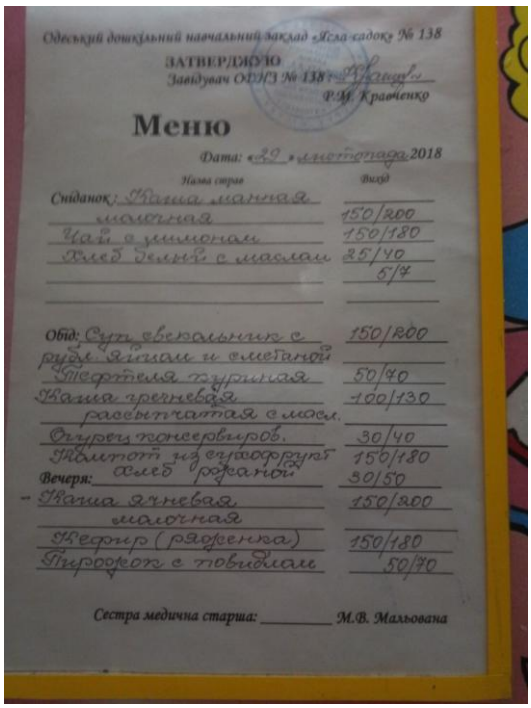
Меню

Дата: «07» декабря 2018

Назва страв	Вихід
Сніданок: Творожно-манная запеканка с изюмом и фруктовой подливой	120/150
Какао натуральное	150/180
Хлеб белый с маслом	25/40 5/7
Обід: Щи с квашеной капустой со сметаной	150/200
Жаркое по домашнему с мясом	100/130
Компот из кураги/изюма	150/180
Хлеб ржаной	30/50
Вечеря: Каша овсяная молочная	150/200
Чай	150/180
Хлеб белый	20/30
Мармелад	20/25

Сестра медична старша: _____ М.В. Мальована

Рисунок 5 – Написане вручну меню на день в дитячому садку



Одеський дошкільний навчальний заклад «Ясла-садок» № 138

ЗАТВЕРДЖУЮ
Завідувач ОДНЗ № 138: _____
Р.М. Кравченко

Меню

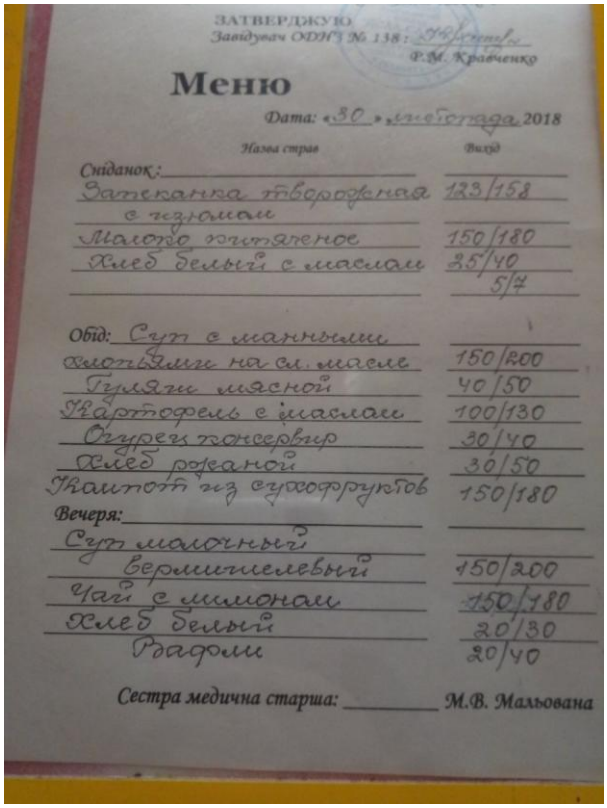
Дата: «29» листопада 2018

Назва страв	Вихід
Сніданок: Каша манная молочная	150/200
Чай с лимоном	150/180
Хлеб белый с маслом	25/40 5/7
Обід: Суп овощной с рубленым яйцом и сметаной	150/200
Тефтеля рубленая	50/40
Каша гречневая рассыпчатая с масл.	100/130
Огурец консервиров.	30/40
Компот из сухофруктов	150/180
Вечеря: хлеб ржаной	30/50
Каша ячневая молочная	150/200
Кефир (ряженка)	150/180
Пирожок с повидлом	50/70

Сестра медична старша: _____ М.В. Мальована

Рисунок 6 – Написане вручну меню на день в дитячому садку

Кожне меню пишеться від руки, що завдає незручності (рис.7):


ЗАТВЕРДЖУЮ
Завідувач ОДНЗ № 138: _____
Р.М. Кравченко

Меню

Дата: «30» листопада 2018

Назва страв	Вихід
Сніданок:	
Запеканка творожная с изюмом	123/158
Молоко кипяченое	150/180
Хлеб белый с маслом	25/40
	5/7
Обід: Суп с манными клецками на сл. масле	150/200
Тушеная мясной	40/50
Картофель с маслом	100/130
Огурец консервир	30/40
Хлеб ржаной	30/50
Компот из сухофруктов	150/180
Вечеря:	
Суп молочный вермишелевый	150/200
Чай с лимоном	150/180
Хлеб белый	20/30
Рафаэли	20/40

Сестра медична старша: _____ М.В. Мальована

Рисунок 7 – Написане вручну меню на день в дитячому садку

1.3 Вибір програмних засобів

1.3.1 Поняття програмного додатку

Програмний додаток (або програмний застосунок) – користувацька комп'ютерна програма, що дає змогу вирішувати конкретні прикладні задачі користувача. Поняття введено, щоб підкреслити відмінність від операційної системи, драйверів, бібліотек, системних утиліт тощо (які забезпечують функціювання власне комп'ютерної системи та підтримують її працездатність) та засобів і середовищ розробки.

Застосунок у своїй роботі спирається на системне програмне забезпечення і використовує (застосовує) концепції, функціональність і можливості,

закладені у середовище, де він виконується: операційної системи, мов програмування, бібліотек, архітектурних шаблонів тощо.

Застосунок потребує безпосередньої взаємодії та забезпечує користувачеві розв'язання певної задачі. Цим самим поняття прикладного ПЗ є протилежним до системного та іншого допоміжного ПЗ (наприклад, операційна система), котрі «лише» забезпечують можливість виконання роботи, але не приносять безпосередньої користі користувачеві.

Проте таке чітке розмежування не завжди можливе, особливо у вбудованих системах (наприклад, мобільний телефон або кавоварка) де програмне та апаратне забезпечення є єдиним цілим.

Окрім того, виробники системного ПЗ нерідко вбудовують також і прикладні програми, відомим прикладом є ОС Windows, де стандартно вбудовано велику кількість застосунків. Ця практика не завжди відповідає вимогам користувачів певної системи.

Є чимало стандартизованого прикладного ПЗ, котре спрямовано на задоволення потреб якомога ширшого кола. Але значна частина прикладного ПЗ розробляється індивідуально «з нуля» або ж на основі стандартних програм для рішення вузьких завдань, наприклад: в межах однієї компанії чи галузі.

Програмний застосунок працює ізольовано від інших додатків і вимагає наявності оператора (людини, яка працює з програмою). Людина взаємодіє із десктопною програмою за допомогою стандартного інтерфейсу.

Вони не вимагають для своєї роботи підключення до інтернету, мають більш високу швидкодію, їх інсталювання може залежати від використовуваної операційної системи, бо програмний додаток вимагає встановлення на кожний комп’ютер користувача, який бажає працювати з даним додатком.

Microsoft Word, Excel, медіа-програвачі, однокористувацькі ігри – в загальному вигляді всі програми, які встановлюються (інсталюються від анг. install) у нас на комп'ютерах, є настільними додатками.

Для роботи такого додатку необхідні достатні технічні ресурси комп'ютера, сам додаток і набір бібліотек (файлів) функцій для роботи з додатком. І оскільки користувач має доступ до системних файлів десктопної програми, даний тип програмного забезпечення в цьому плані вважається більш уразливим, так як залежить від необачних дій користувача[2][1)].

1.3.2 Система керування базами даних MySQL

MySQL – це система управління базами даних. База даних являє собою структуровану сукупність даних. Ці дані можуть бути будь-якими – від простого списку майбутніх покупок до переліку експонатів картинної галереї або величезної кількості інформації в корпоративній мережі. Для запису, вибірки і обробки даних, що зберігаються в комп'ютерній базі даних, необхідна система управління базою даних, якою і є MySQL. Оскільки комп'ютери справляються з обробкою великих обсягів даних, управління базами даних відіграє центральну роль в обчисленнях. Реалізовано таке управління може бути по-різному – як у вигляді окремих утиліт, так і у вигляді коду, що входить до складу інших додатків. MySQL – це система керування базами даних. У реляційній базі даних дані зберігаються не всі скопом, а в окремих таблицях, завдяки чому досягається виграш в швидкості і гнучкості. Таблиці зв'язуються між собою за допомогою відносин, завдяки чому забезпечується можливість об'єднувати при виконанні запиту дані з декількох таблиць. Програмне забезпечення MySQL – це ПЗ з відкритим кодом. ПЗ з відкритим кодом означає, що застосовувати і модифікувати його може будь-хто. Таке ПЗ можна одержувати по Internet і використовувати без-

[1)] [2] Лабораторія СЕТ. Термін «application» українською. URL: http://www.setlab.net/?view=application–term. (дата звернення 25.03.19)

коштовно. При цьому кожен користувач може вивчити вихідний код і змінити його відповідно до своїх потреб. MySQL має API та конектори для мов Delphi, C, C ++, Ейфель, Java, Лисп, Perl, PHP, Python, Ruby, Smalltalk, Компонентний Паскаль Tcl, бібліотеки для мов платформи .NET, а також забезпечує підтримку для ODBC за допомогою ODBC-драйвера MyODBC . Розмір таблиці обмежений її типом. У загальному випадку тип MyISAM обмежений граничним розміром файлу в файлової системі операціонной системи. Наприклад, в NTFS цей розмір теоретично може бути до 32 ексабайт. У разі InnoDB одна таблиця може зберігатися в декількох файлах, які мають єдиний табличний простір. Розмір останнього може досягати 64 терабайт. MySQL є дуже швидким, надійним і легким у використанні. MySQL володіє також рядом зручних можливостей, розроблених в тісному контакті з користувачами. ПЗ MySQL є системою клієнт-сервер, що забезпечує підтримку різних обчислювальних машин баз даних, а також кілька різних клієнтських програм і бібліотек, засоби адміністрування і широкий спектр програмних інтерфейсів (API)[3][1)].

1.3.3 Мова створення гіпертекстових документів HTML

HTML (HyperText Makeup Language – мова створення гіпертекстових документів) – стандарний сервіс мережі Інтернет, прийнятий WWW– консорціумом (W3C). HTML – це звичайний текст (інформація) плюс керуючі елементи – теги (зручне представлення цієї самої інформації). Теги – (від англ. Tag) це засіб, дозволяючий браузеру надавати наявну у вас інформацію у вигляді, зручному для сприйняття, пошука, читання. Це в своєму роді мова програмування, що визначає, як буде виглядати на екрані наступний рядок, що потрібно виділити, а що – навпаки – настільки очевидно, що немає необхідності це підкреслювати. Тегом вважається певний вираз, поміщене в

[1)] [3] MySQL. Руководство администратора MySQL. Administrator's Guide. M.: Издательский дом «Вильямс», 2005. 624 с.

дужки виду <>. Умовно теги можна розділити на граничні і унарні. Унарні теги являють собою очевидні операції, які веліли браузеру вчинити певну дію по осягненню даного тега при обрамленні html – документа. Приклад – найбільш часто вживається тег
 – перехід на інший рядок, весь наступний за ним гіпертекст браузер буде представляти, починаючи з нового рядка. Унарні теги не належать до певних ділянок тексту, а, швидше за все, до всього документа. З граничними тегами складніше, але треба враховувати, що їх набагато більше, ніж унарних. Граничні теги визначають, як буде виглядати на екрані користувача частина тексту, визначених цими елементами, в кінцевому тегу ці параметри присутні не повинні (браузером вони ігноруються). Деякі параметри визначені за замовчуванням браузером, деякі – користувачем браузера, але є і такі, визначати які необхідно при створенні сторінки (приклад – той самий параметр href тега <a>).

З усього вищесказаного випливає, що, знаючи найуживаніші теги і їх параметри, будь-який користувач легко може редагувати і створювати гіпертекст. При цьому необхідно враховувати, що теги можуть включати в себе інші теги. Структура гіпертекстових файлів надзвичайно проста. HTML-документ повинен починатися тегом <html> і закінчуватися </html> . Інформація поза цими тегами ігнорується, або видається в безсторонньому вигляді. Крім цього все, огороджене тегами <html> і </ html> ділиться на дві частини.

Заголовок (менша частина). Обмежується тегами <head> і </ head> . Містить, як правило мета-інформацію, тобто додаткові дані про сторінку, і заголовок сторінки, що виноситься як правило в заголовок вікна браузера (у віконній графічній системі).

Заголовок визначається тегами <title> і </ title>, його присутність необов'язкова, але бажана.

Основна частина документа (тіло). Обмежується тегами <body> і </body>. Визначається те, що виводиться в головному вікні браузера. Тут

зосереджується вся інформація сторінки. Також може бути відсутнім при використанні фреймів[4][1)].

1.3.4 Каскадна таблиця стилів CSS

CSS (англ. Cascading Style Sheets – каскадні таблиці стилів) – технологія опису зовнішнього вигляду документа, оформленого мовою розмітки.

Переважно використовується як засіб оформлення веб-сторінок в форматі HTML і XHTML, але може застосовуватися з будь-якими видами документів в форматі XML, включаючи SVG і XUL.

Каскадні таблиці стилів використовуються творцями веб-сторінок для завдання кольорів, шрифтів, розташування і інших аспектів представлення веб–документа. Основною метою розробки CSS було розділення вмісту (написаного на HTML або іншій мові розмітки) і оформлення документа (написаного на CSS). Цей поділ може збільшити доступність документа, надати велику гнучкість і можливість управління його поданням, а також зменшити складність і повторюваність в структурному вмісті. Крім того, CSS дозволяє представляти один і той же документ в різних стилях або методах виведення, таких як екранне уявлення, друк, читання голосом (спеціальним голосовим браузером або програмою читання з екрану), або при виведенні пристроями, що використовують шрифт Брайля. Каскадні таблиці стилів – це стандарт, який визначає уявлення даних в браузері. Якщо HTML надає інформацію про структуру документа, то таблиці стилів повідомляють як він повинен виглядати.

Стиль – це сукупність правил, що застосовуються до елементу гіпертексту і визначають спосіб його відображення. Стиль включає всі типи

[1)] [4] HTML – язык разметки гипертекстов. URL: http://html.find–info.ru/html/articles/article_8.htm. (дата звернення 26.03.19)

елементів дизайну: шрифт, фон, текст, кольори посилань, поля і розташування об'єктів на сторінці.

Таблиця стилів – це сукупність стилів, які можна застосувати до гіпертекстових документів.

Каскадування – це порядок застосування різних стилів до вебсторінки. Браузер, що підтримує таблиці стилів, буде послідовно застосовувати їх відповідно до пріоритету: спочатку пов'язані, потім впроваджені і, нарешті, вбудовані стилі. Інший аспект – успадкування (inheritance), – означає, що якщо не вказано інше, то конкретний стиль буде застосований до всіх дочірніх елементів гіпертекстового документа. Наприклад, якщо ви застосуєте певний колір тексту в тезі , то усі теги всередині цього блоку будуть відображатися цим же кольором.

Використання каскадних таблиць дає можливість розділити вміст і його уявлення і гнучко управляти відображенням гіпертекстових документів шляхом зміни стилів. Таблиці стилів будуються відповідно до визначеного порядку (синтаксисом), в іншому випадку вони не можуть нормально працювати. Таблиці стилів складаються з певних частин.

Селектор (Selector). Селектор – це елемент, до якого будуть застосовуватися призначаються стилі. Це може бути тег, клас або ідентифікатор об'єкта гіпертекстового документа.

Властивість (Property). Властивість визначає одну або декілька характеристик селектора. Властивості задають формат відображення селектора: відступи, шрифти, вирівнювання, розміри і т.д.

Значення (Value). Значення – це фактичні числові або строкові константи, що визначають властивість селектора.

Опис (Declaration). Сукупність властивостей і їх значень.

Правило (Rule). повний опис стилю (селектор + опис).

Таким чином, таблиця стилів – це набір правил, які задають значення властивостей селекторів, перерахованих в цій таблиці[5][1)].

1.3.5 Бібліотека jQuery

Дана бібліотека дозволяє змінювати вміст HTML-документів шляхом маніпулювання об'єктами моделі, створюваної браузерами в процесі обробки HTML-коду (так звані DOM-маніпуляції).

Засоби jQuery надзвичайно виразні. Ця бібліотека дозволяє домогтися набагато більшого при набагато меншому обсязі коду, ніж в разі використання програмних DOM-інтерфейсів браузерів.

Методи jQuery застосовні до цілих груп елементів. Пропонований в DOM-моделі стандартний підхід, заснований на шаблонному ланцюжку дій "вибрати–повторити–змінити", більше не потрібно.

Наслідком цього є зменшення кількості циклів for в коді, а значить, і зниження ймовірності появи в ньому помилок.

Бібліотека jQuery справляється з відмінностями в реалізації DOM в різних браузерах (проблеми крос-браузерності).

Бібліотека jQuery має відкритий вихідний код.

Якщо принципи роботи будь-якого засобу не зовсім зрозумілі або отриманний результат не збігається з очікуваним, можливо звернутися безпосередньо до коду бібліотеки на JavaScript і, якщо це необхідно, внести відповідні зміни.

Кросплатформені можливості. Вона автоматично виправляє помилки і працює таким же чином в найбільш часто використовуваних браузерах, таких як Chrome, Firefox, Safari, MS Edge, IE, Android і iOS.

JQuery також робить Ajax набагато простіше. Ajax працює асинхронно з іншою частиною коду.

[1)] [5] IIT Lectures. Каскадные таблицы стилей. URL: https://iit–web–lectures.readthedocs.io/ru/latest/www/css.html. (дата звернення 27.03.19)

Це означає, що код, написаний на Ajax, може взаємодіяти з сервером і оновлювати його вміст без необхідності перезавантаження сторінки.

Різні браузери виконують Ajax API по-різному. Таким чином, код повинен відповідати всім браузерам.

Вручну, це важка і трудомістка робота. На щастя, jQuery виконує всю важку роботу і адаптує код для всіх веб-браузерів.

Потім є маніпулювання DOM, в якому є кілька методів, як це зробити. Простіше кажучи, він дозволяє вставляти і/або видаляти елементи DOM на HTML-сторінці, а також спрощує перенесення рядків.

Обхід документів HTML, а також виконання ефектів і обробка подій також поліпшені за допомогою jQuery[6][1)].

1.3.6 Бібліотека Electron

Electron – бібліотека з відкритим вихідним кодом, розроблена GitHub, дозволяє розробляти нативні графічні додатки для десктопних операційних систем за допомогою веб-технологій: HTML, CSS, і JavaScript. Electron досягає цього шляхом об'єднання Chromium і Node.js в єдине середовище виконання, а додатки можуть бути зібрані для виконання під Mac, Windows і Linux.

Варто відзначити перевагу, яку нам дає розробка десктопних додатків за допомогою Electron – легке створення призначеного для користувача інтерфейсу за допомогою веб-технологій і багатий API, завдяки якому можна створювати і налаштовувати вікно програми під будь-які потреби.

Те, з чого складається Electron, зображено на рис. 8.

[1)] [6] Professor Web. Библиотека jQuery. URL: https://professorweb.ru/my/javascript/jquery/level1/jquery_index.php (дата звернення 26.03.19)

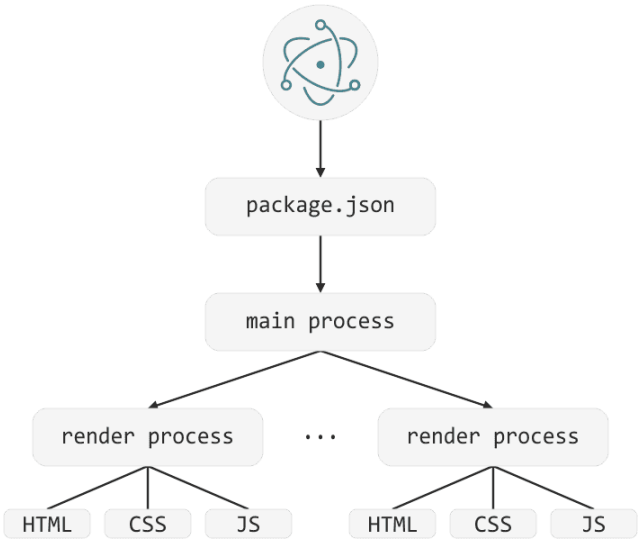


Рисунок 8 – Складові частини Electron

У Electron-додатку існує два типи процесів: основний процес і процес візуалізації.

Основний процес (main process) – це процес, який запускається з package.json. Скрипт, запущений в основному процесі, може відображати GUI, використовуючи веб-сторінки. У Electron-додатку завжди є один головний main process. Основний процес управляє процесами візуалізації.

Процес візуалізації (renderer process) відповідає за запуск призначеного для користувача інтерфейсу вашої програми, або, іншими словами, веб-сторінки, що є екземпляром webContents (відповідає за рендеринг і управління веб-сторінкою).

Electron використовує Node.js і бібліотеку рендерінгу з Chromium. По суті створюється звичайний web-додаток і загортається в виконуваний файл. При чому кожен додаток поставляється з власною версією Node.js і Chrome.

Але є недолік – це велике (в порівнянні з іншими технологіями) споживання пам'яті: порожній проект займає в пам'яті 100–200 мегабайт. Для деяких це причина відмовитися від використання таких додатків.

На даний момент існує багато популярних додатків написані на Electron (Slack, Skype, Discord, VSCode, Atom, Postman, Insomnia і т.д.). Багато з них є стандартами у своїй сфері або стрімко завойовують серця користувачів

(як, наприклад, ті ж VSCode і Insomnia). Люди просто використовують інструменти, які добре вирішують їх повсякденні завдання, незважаючи на деякі побічні ефекти. З іншого боку, комп'ютери стають дедалі потужнішими (як мінімум, зростання RAM не припинився). Резюмуючи, що підвищене споживання оперативної пам'яті не буде грати великої ролі в разі, якщо продукт буде дійсно гарний у своїй сфері.

Плюси даної технології:

– можливість використання напрацювань з web;

– простіше знайти фахівців в даній сфері;

– відпрацьований воркфлоу «дизайнер» – «верстальник» – «програміст», буде зручними інструментами на кожному етапі. [7][1)].

1.3.7 Середа розробки NodeJs

Node.js – це середовище виконання JavaScript, побудована на JavaScript-движку V8 з Chrome.

Node.js надає можливість писати продуктивний серверний код з використанням JavaScript. Node.js – це середовище виконання, що використовує той же JavaScript–движок V8, який ви можете знайти в браузері Google Chrome. Але цього недостатньо для успіху Node.js. У Node.js використовується libuv – крос-платформна бібліотека підтримки з акцентом на асинхронний ввід–висновок.

З точки зору розробника, Node.js однопоточний, але під капотом libuv використовує треди, події файлової системи, реалізує цикл подій, включає в себе тред-пулінг і так далі[8][2)].

[1)] [7] Habr. Почему мы выбрали Electron. URL: https://habr.com/ru/post/439946/. (дата звернення 27.03.19)

[2)] [8] A Medium Corporation. Начало работы с Node.js. URL: https://medium.com/devschacht/node–hero–chapter–1–239f7afeb1d1. (дата звернення 28.03.2019)

Node.js використовує модульну систему. Тобто вся вбудована функціональність розбита на окремі пакети або модулі. Модуль представляє блок коду, який може використовуватися повторно в інших модулях.

При необхідності ми можемо підключати потрібні нам модулі. Які вбудовані модулі є в node.js і які функції вони надають, можна дізнатися з документації.

Для завантаження модулів застосовується функція require (), в яку передається назва модуля.

Після отримання модуля можливо використовувати весь визначений у ній функціонал, який знову ж таки можна подивитися в документації.

Подібним чином можливо завантажувати і використовувати інші вбудовані модулі. При необхідності можливо створити свої[9][1)].

1.4 Постановка технічного завдання

Даний розділ представляє собою технічні вимоги до програмного забезпечення. В ньому будуть описані вимоги до програми, використовувані підходи до неї.

Програмний додаток «Дитячий раціон» повинен виконувати такі функції:

- заповнення вибору страв;
- при виборі страви повинен видаватися її рецепт;
- розраховувати кількість порцій на потрібну кількість дітей;
- наявність складу для збереження продуктів;
- можливість порівняння потрібних продуктів та наявних;
- можливість додавати або віднімати продукти зі складу.

При розробці технічних вимог було:

[1)] [9] Metanit.com. Введение в Nodejs. URL: https://metanit.com/web/nodejs/1.1.php (дата звернення 27.03.2019)

– проаналізовано різні підходи до функціональності програмного забезпечення;

– визначено основні технічні вимоги до програмного забезпечення, необхідний склад технічних, функціональних та апаратних модулів;

– критерії для функціонування виробу.

Повне найменування системи дипломного проекту: «Розробка програмного додатку для планування дитячого раціону».

2 ПРОЕКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

Даний розділ присвячено побудові бізнес-моделей сервісу, логічній та фізичній моделі. побудову бази даних та реалізацію за допомогою середи розробки.

2.1 Побудова функціональної моделі сервісу

Основу багатьох сучасних методологій моделювання бізнес-процесів склали методологія SADT (Structured Analysis and Design Technique – метод структурного аналізу й проектування), сімейство стандартів IDEF (Icam DEFinition, де Icam – це Integrated Computer-Aided Manufacturing) і алгоритмічні мови.

Основні типи методологій моделювання й аналізу бізнес–процесів:

− моделювання бізнес–процесів (Business Process Modeling).

Найбільше широко використовувана методологія опису бізнес–процесів – стандарт IDEF0. Моделі в нотації IDEF0 призначені для високорівневого опису бізнесу компанії у функціональному аспекті;

− опис потоків робіт (Work Flow Modeling) стандарт IDEF3.

Стандарт IDEF3 призначений для опису робочих процесів і близький до алгоритмічних методів побудови блок–схем;

− опис потоків даних (Data Flow Modeling).

Нотація DFD (Data Flow Diagramming), дозволяє відобразити послідовність робіт, виконаних в ході процесу, і потоки інформації, що циркулюють між цими роботами;

− інші методології. [10][1)]

[1)] [10] Ильин В.В. Моделирование бизнес–процессов. Практический опыт разработчика. Вильямс. 2006. 365 с.

Важливим елементом моделі бізнес-процесів є бізнес–правила або правила предметної області. Типовими бізнес-правилами є корпоративна політика й державні закони. Бізнес-правила звичайно формулюються в спеціальному документі й можуть відображатися в моделях.

Декомпозиція в загальному значенні – це метод, що дозволяє замінити рішення одної великої задачі рішенням серії менших задач, розщеплення об'єкта на складові частини за встановленим критерієм. Практично декомпозиція застосовується для деталізації бізнес–моделей.

Етапи опису бізнес–процесів по методологія SADT:

− визначення цілей опису;

− опис оточення, визначення входів і виходів бізнес-процесу, побудова IDEF0-діаграм;

− опис функціональної структури (дії процесу), побудова IDEF3-діаграм;

− опис потоків (матеріальних, інформаційних, фінансових) процесу, побудова DFD-діаграм;

− побудова організаційної структури процесу (відділи, учасники, відповідальні).

Методологія SADT (Structured Analysis and Design Technique – технологія структурного аналізу й моделювання) розроблена Дугласом Т. Россом в 1969–1973р. SADT – одна з найвідоміших і широко використовуваних методик моделювання. Нова назва методики, прийнята як стандарт – IDEF0 (Icam DEFinition) – частина програми ICAM (Integrated Computer– Aided Manufacturing – інтеграція комп'ютерних і промислових технологій).

Методологія SADT являє собою сукупність методів, правил і процедур, призначених для побудови функціональної моделі об'єкта якої–небудь предметної області[11][1)].

[1)] [11] Катренко А.В. Управління ІТ–проектами. Львів: «Новий світ 2000», 2013. 550с.

З погляду SADT модель зосереджена на функціях системи. Функціональна модель із необхідним ступенем деталізації являє собою систему функцій, які відображає свої взаємини через об'єкти системи.

Процес моделювання починається з аналізу предметної області й включає:

- збір інформації про досліджувану область;
- документування отриманої інформації;
- подання її у вигляді моделі.

SADT-модель дає повний і точний опис системи, що має конкретне призначення. Це призначення системи називається метою моделювання. Фактично мета визначає відповідні області в досліджуваній системі, на яких необхідно сфокусуватися в першу чергу.

Метою моделювання є одержання відповідей на деяку сукупність питань. Ці питання завжди присутні в процесі аналізу системи й керують створенням моделі. Якщо модель відповідає не на всі питання або її відповіді недостатньо точні, то побудована модель не досягла своєї мети. Визначаючи модель таким чином, методологія SADT закладає основи моделювання.

Точка зору визначає основний напрямок розвитку моделі й рівень необхідної деталізації. Чітке фіксування точки зору дозволяє спростити модель, відмовившись від деталізації й дослідження окремих компонентів, що не є для даної моделі важливими. Правильний вибір точки зору скорочує тимчасові витрати на побудову кінцевої моделі.

Деталізація процесів дозволяє виявити недоліки навіть там, де функціональність на перший погляд здається очевидною.

Кінцевим результатом цього процесу є набір взаємозалежних описів, починаючи з опису самого верхнього рівня всієї системи й кінчаючи докладним описом деталей або операцій. Кожне з таких описів називається діаграмою. SADT – модель поєднує діаграми в ієрархічну деревоподібну структуру, у якій верхня діаграма є найбільш загальною, а самі нижні найбільш деталізовані.

Модель SADT відображає функціональну структуру об'єкта, тобто вироблені ними дії й зв'язки між цими діями. Основні елементи цієї методології ґрунтуються на графічному поданні блокового моделювання. Виконання правил SADT вимагає достатньої строгості й точності, не накладаючи надмірних обмежень на дії аналітика.

Основу методології SADT становить графічна мова опису процесів. Модель в SADT являє собою сукупність ієрархічно впорядкованих і взаємозалежних діаграм. Кожна діаграма є одиницею опису системи й розташовується на окремому аркуші[12][1)].

Після опису системи в цілому проводиться розбивка її на великі фрагменти. Цей процес називається функціональною декомпозицією, а діаграми, які описують кожен фрагмент і взаємодію фрагментів, називаються діаграмами декомпозиції.

Після декомпозиції контекстної діаграми проводиться декомпозиція кожного великого фрагмента системи на більше дрібні й так далі, до досягнення потрібного рівня подробиці опису.

Елементи контекстної діаграми:

- блок (Робота);

Роботи означають пойменовані процеси, функції або задачі, які відбуваються протягом певного часу й мають розпізнавані результати. Роботи зображуються у вигляді прямокутників. Всі роботи повинні бути названі й визначені. Ім'я роботи повинне бути виражене дієсловом наказового способу або віддієслівним іменником, що позначає дію. Номер блоку розміщається в правому нижньому куті. Номера блоків використаються для ідентифікації на діаграмі й у відповідному тексті. Кожна робота може бути декомпозована.

Контекстна діаграма містить тільки одну саму головну роботу, виконувану системою.

- граничні стрілки;

[1)] [12] Управление проектом. Основы проектного управления: учебник под ред. проф. М. Л.Разу. М. : КНОРУС, 2006. 450 с.

Стрілки – це об'єкти реального світу або яка–небудь інформація, яка необхідні для виконання роботи або є результатом виконання роботи, представленої на діаграмі функціональним блоком.

Стрілки на контекстній діаграмі служать для опису взаємодії системи з навколишнім світом. Такі стрілки називаються граничними. Вони можуть починатися у границі діаграми й закінчуватися біля роботи, або навпаки, починатися у роботи й закінчуватися біля границі діаграми[13][1)].

Представимо графічне зображення функціонального блоку із граничними стрілками, які можуть бути пов'язані з цим блоком, на рис. 9.

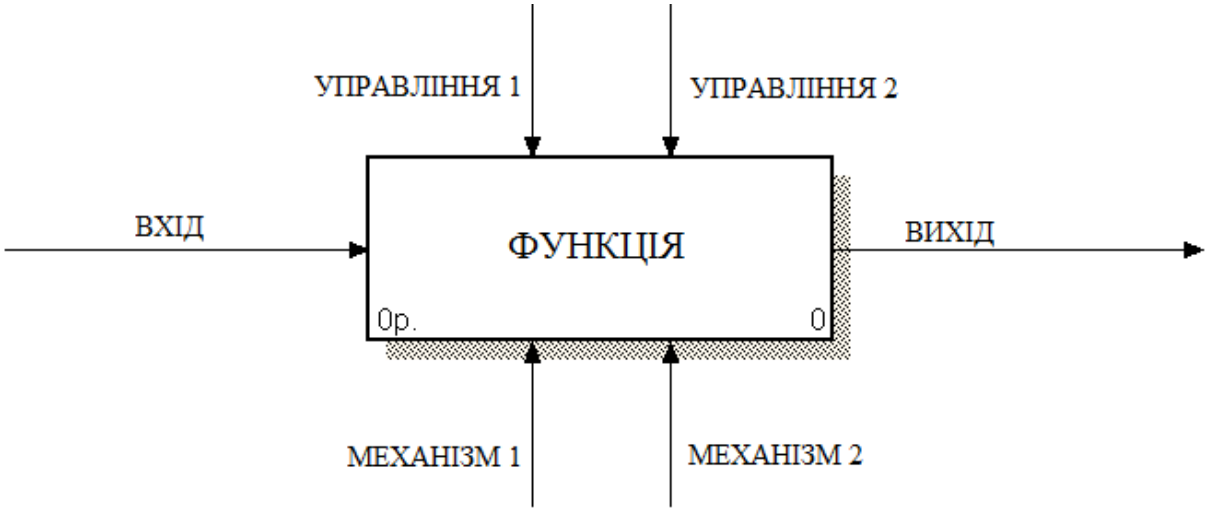


Рисунок 9 – Функціональний блок із граничними стрілками

Побудова SADT–моделі починається з подання всієї системи у вигляді одного блоку й граничних стрілок, що зображують зв'язки з функціями поза системою.

Діаграма, що складається з одного блоку й стрілок, визначає границі системи й називається контекстною діаграмою моделі. При цьому блок зображує границі системи: все що знаходиться всередині нього, є частиною си-

[1)] [13] Черемных С.В., Семенов И.О., Ручкин В.С. Моделирование и анализ систем. IDEF-технологии. Учебник-практикум: М.: Финансы и статистика, 2006. 188 с.

стеми, а все що знаходиться поза ним, утворить зовнішнє середовище системи.

Робота – це блок, що представляє систему у вигляді одного функціонального модуля, деталізується на діаграмі декомпозиції за допомогою декількох блоків (звичайно від 3 до 6), з'єднаних внутрішніми стрілками. Ці блоки являють собою основні підфункції вихідної функції. Така декомпозиція виявляє повний набір робіт, кожна з яких представляється як блок, границі якого визначаються відповідними стрілками. Кожна із цих робіт може бути декомпозована подібним чином для більше детального подання.

Діаграма декомпозиції призначена для деталізації роботи. Роботи нижнього рівня – це теж саме, що робота верхнього рівня, але в більш детальному викладі. Як наслідок цього границі роботи верхнього рівня – це ті ж самі, що границі діаграми декомпозиції.

При декомпозиції якої–небудь роботи вхідні в неї й вихідні з неї стрілки повинні відобразитися на діаграмі декомпозиції, тобто повинна відбутися міграція стрілок. Одним зі способів контролю міграції стрілок служать ICOM–коди, призначені для ідентифікації граничних стрілок. ICOM – абревіатура від Input, Control, Output й Mechanism. Кожна гранична стрілка діаграми декомпозиції повинна бути позначена за допомогою коду ICOM, що збігає з позначенням відповідної стрілки на батьківській діаграмі. Код ICOM містить префікс, що відповідає типу стрілки I, C, O або M, і порядковий номер[14][1)].

Для зв'язку робіт між собою в діаграмах декомпозиції використовуються внутрішні стрілки, які починаються з однієї роботи й закінчуються в іншій. Розрізняють п'ять типів внутрішніх стрілок для зв'язків робіт.

[1)] [14] A Guide to the Project Management Body of Knowledge. Project Management Body Of Knowledge. Project Management Institute, Four Campus Boulevard, Newtown Square, PA 19073–3299 USA. 2004. 792 p.

Зв'язок по входу – зв'язок, при якому вихід вищестоящої роботи направляється на вхід нижчестоящої роботи. Зображений на рис. 10.

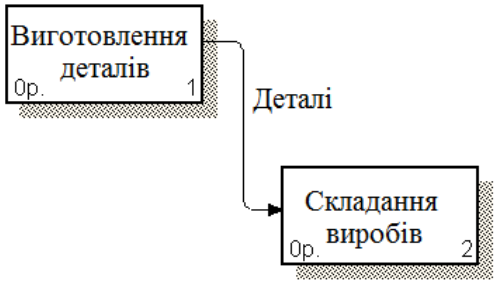


Рисунок 10 – Зв'язок по входу

Явні стрілки мають джерелом одну-єдину роботу й призначенням теж одну–єдину роботу.

Однак ті самі дані або об'єкти, породжені однією роботою, можуть використатися відразу в декількох інших роботах. Тоді ми говоримо про стрілку, що розгалужується. Зображена на рис. 11.

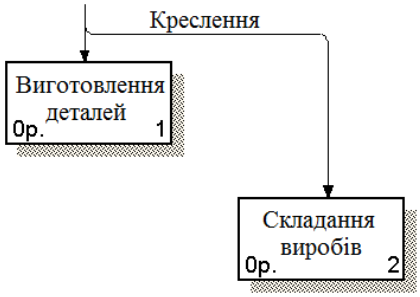


Рисунок 11 – Стрілка, що розгалужується

З іншого боку, стрілки, породжені в різних роботах, можуть являти собою однакові або однорідні дані або об'єкти, які надалі використовуються або переробляються в одному місці. Для моделювання такої ситуації використаються стрілки, що зливаються.

Сенс стрілок, що розгалужуються й зливаються передається іменуванням кожної гілки стрілок. Існують певні правила іменування таких стрілок.

Якщо стрілка іменована до розгалуження, а після розгалуження жодна з гілок не іменована, то мається на увазі, що кожна гілка моделює ті ж дані або об'єкти, що й гілка до розгалуження.

Правила іменування стрілок, що зливаються, повністю аналогічні. Неприпустимою буде вважатися ситуація, при якій стрілка після злиття не іменована, а до злиття не іменована яка-небудь із її гілок.

Модель SADT являє собою серію діаграм, що розбиває складний об'єкт на складові частини, які представляються у вигляді блоків. Деталі кожного з основних блоків показані у вигляді блоків на інших діаграмах. Кожна детальна діаграма є декомпозицією блоку з більш загальної діаграми. На кожному кроці декомпозиції більш загальна діаграма називається батьківською для більше детальної діаграми.

Стрілки, що входять у блок і виходять із нього на діаграмі верхнього рівня, є точно тими ж, що й стрілки, що входять у діаграму нижнього рівня й виходять з неї, тому що блок і діаграма представляють ту саму частину системи[15][1)].

Ці питання задають у зазначеному порядку для того, щоб спочатку вирішити дрібні, а потім перейти до більш глобального. Всі ці кроки вживають для вироблення думки про обґрунтованість і правильність діаграми.

З методології SADT тільки модель, що пройшла перевірку на коректність синтаксису, зв'язків між складовими її діаграмами й адекватність опису об'єкта моделювання може бути допущена до використання в подальшій роботі.

Побудуємо такі діаграми для нашого проекту.

[1)] [15] Грекул В.И. и др. Проектирование информационных систем. Курс лекций. Интернет-Университет Информационных Технологий: М.:2005. 624 с.

Для додатку Дитяче харчування функціональна діаграма має наступний вигляд (рис. 12):

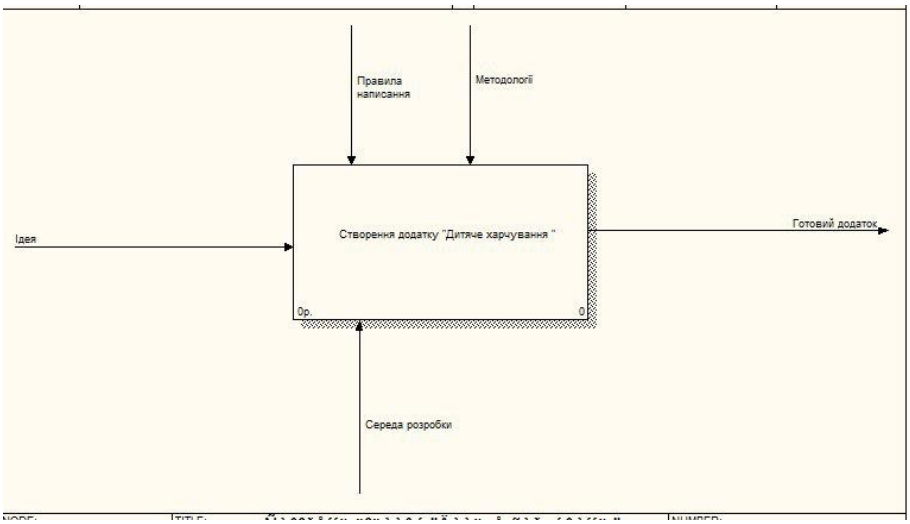


Рисунок 12 – Контекстна діаграма «Дитяче харчування»

З малюнку видно, що вхідними даними є ідея. На виході отримуємо готовий додаток. Головними чинниками, які впливають на розробку, є правила написання та методології. Основним механізмом є середа розробки. Наступним кроком буде декомпозиція основного процесу на декілька блоків (рис. 13):

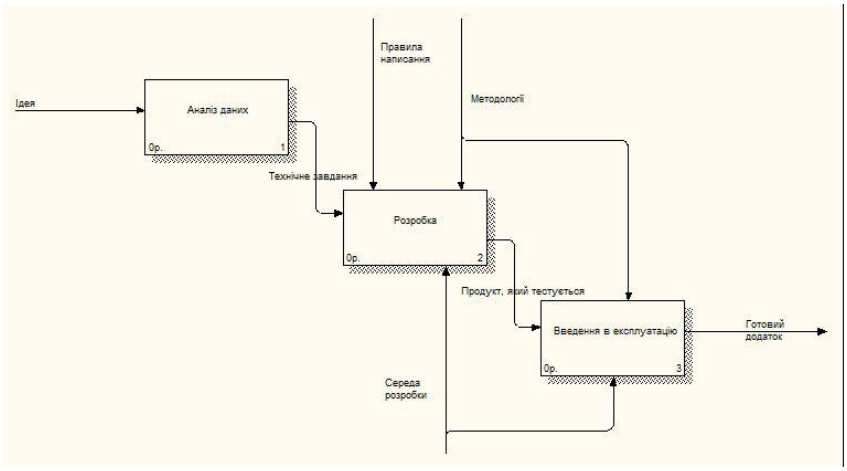


Рисунок 13 – Перша декомпозиція основного процесу

Кожна декомпозиція дає змогу більш детально роздивитися бізнес–процеси та зрозуміти як відбувається кожен етап розробки.

Наступним кроком є декомпозиція блоку «аналіз даних» (рис. 14):

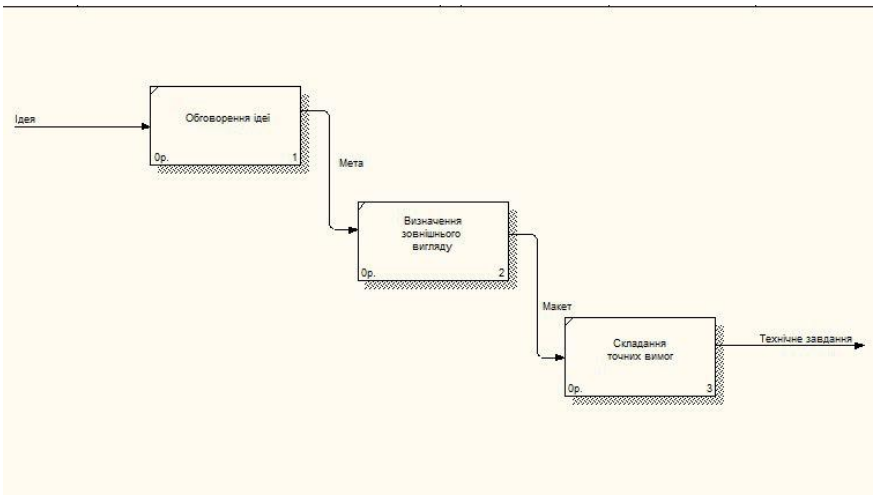


Рисунок 14 – Друга декомпозиція процесу «Аналіз даних»

Надалі необхідно повторити теж саме з блоками «Розробка» і «Введення в експлуатацію». Декомпозиція для блоку «Розробка» зображена на рис. 15:

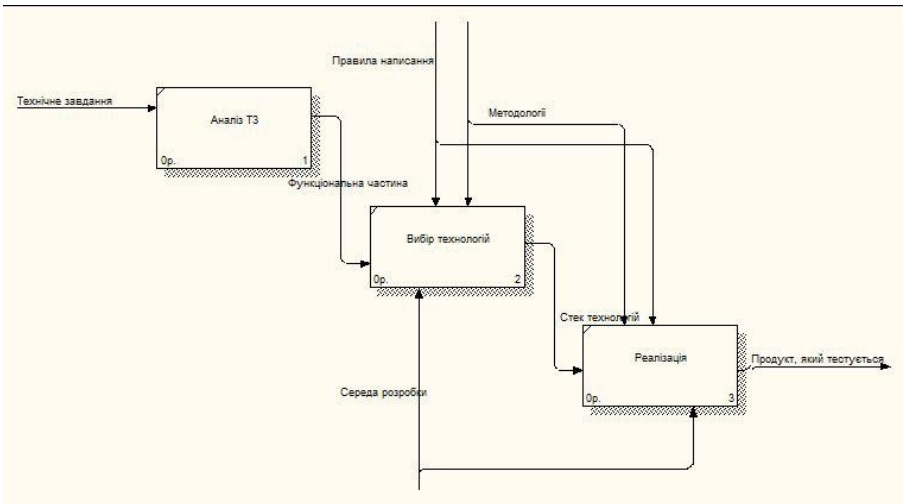


Рисунок 15 – Третя декомпозиція процесу «Розробка»

Завдяки розбиттю блоків видно вплив та взаємодію інформації під час роботи над сервісом.

Останнім етапом є декомпозиція блоку «Введення в експлуатацію» (рис. 16):

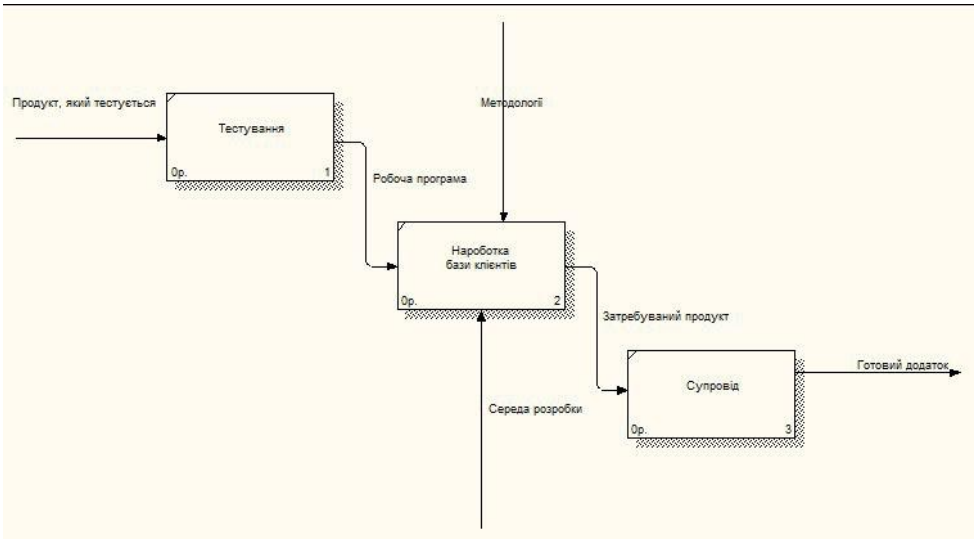


Рисунок 16 – Четверта декомпозиція процесу «Введення в експлуатацію»

В подальшому можна робити декомпозицію на декілька рівнів більше, але для побудови та реалізації програмного додатку «Дитяче харчування» трьох рівнів достатньо, щоб розуміти функціональну структуру[16][1)].

2.2 Побудова діаграми класів для додатку «Дитяче харчування»

Діаграма класів (англ. class diagram) – статичне представлення структури моделі. Відображає статичні (декларативні) елементи, такі як: класи, типи даних, їх зміст та відношення. Діаграма класів, також, може містити позначення для пакетів та може містити позначення для вкладених

[1)] [16]Мазур, И. И. Управление проектами : учеб. пособие / И. И. Мазур, В. Д. Шапиро, Н. Г. Ольдерогге; под общ. ред. И. И. Мазура.:М.: Омега-Л, 2005. 412 с.

пакетів. Також, діаграма класів може містити позначення деяких елементів поведінки, однак їх динаміка розкривається в інших типах діаграм[17][1) .

Діаграма класів служить для представлення статичної структури моделі системи в термінології класів об'єктно–орієнтованого програмування. На цій діаграмі показують класи, інтерфейси, об'єкти й кооперації, а також їхні відносини.

Для додатку «Дитяче харчування» діаграма класів, буде мати наступний вигляд (рис.17) :

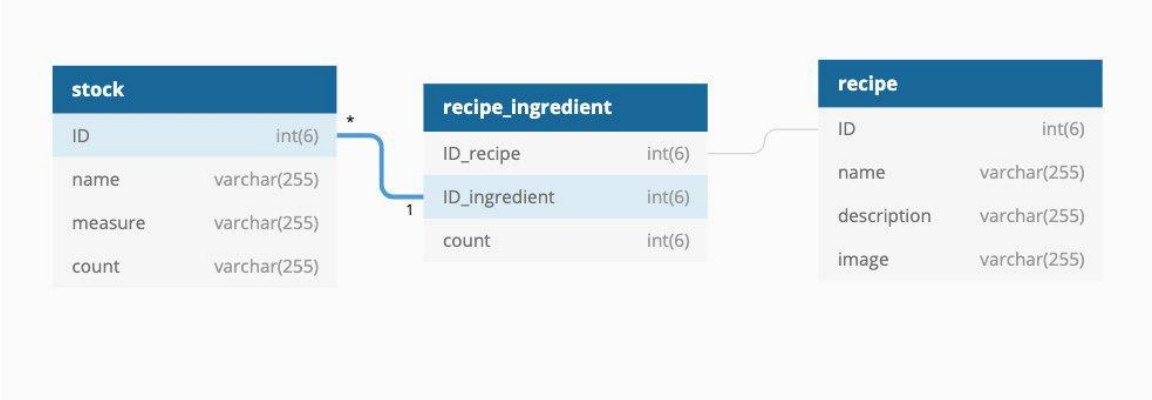


Рисунок 17 – Діаграма класів для контролерів додатку «Дитяче харчування»

[1)] [17] James Rumbaugh, Ivar Jacobson, Grady Booch. The unified modeling language reference manual (англ.). Addison Wesley Longman Inc. ISBN 0–201–30998–X. 1999. 732 p.

3 РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

Даний розділ присвячено реалізації проекту . Буде розглянуто програмний код побудований на основі результатів логічного проектування та отриманні діаграм классів.

3.1 Реалізація логіки додатку «Дитяче харчування»

Додаток побудований на методології MVC, яка використовує контролери. Контролер забезпечує «зв'язок» між користувачем і системою. Контролює і направляє дані від користувача до системи і навпаки. Використовує модель і уявлення для реалізації необхідного дії.

Контролери:

– IngredientController – контролер відповідає за обробку списку інгредієнтів;

– RecipeController – контролер відповідає за сторінку з рецептами;

– StockController – контролер відповідає за управління складом.

Моделі:

– Recipe – модель, зв’язана з базою даних рецептів;

– RecipeIngredient – модель, зв’язана з базою даних інгредієнтів;

– Stock – модель, зв’язана з базою даних складу.

Контролер сторінки інгредієнтів:

```
class IngredientController{
```

Метод показу списку інгредієнтів:

```
    public function actionView(){

        $ingredientList = array();
        $ingredientList = Stock::getIngredientList();
            include_once(ROOT .
'/views/ingredient/view.php');
        }
```

Метод додавання нового інгредієнта. За допомогою POST ми отримали назву та міру виміру.

```
            public function actionAdd() {

                if (isset($_POST['submit'])) {
                    $options['name'] = $_POST['name'];
                    $options['measure'] = $_POST['measure'];
                    $id = Stock::createIngredient($options);
                    header("Location: /ingredients");
                }
               require_once(ROOT .
'/views/ingredient/create.php');
            }
```

Метод видалення інгредієнтів:

```
public function actionDelete($id) {
             Stock::deleteIngredient($id);
             header("Location: /ingredients");
                 }
```

Контролер сторінки рецептів:

```
class RecipeController{
```

Метод отримання списку страв:

```
public function actionIndex(){
$recipeList = array();
$recipeList = Recipe::getRecipeList();
require_once(ROOT . '/views/recipe/index.php');
return true;
    }
```

Другий рядок визначає масив, в який на третьому рядку приходять дані з бази про рецепти, і четвертий рядок викликає відображення і передає в нього ось той масив.

Метод показу рецепту страви:

```
        public function actionView($id) {
```

```
            $recipeIngredient = RecipeIngre-
dient::getListIngredientByRecipe($id);
            $recipe = Recipe::getRecipeById($id);
            require_once(ROOT . '/views/recipe/recipe.php');
        }
```

Метод перенаправлення з головної сторінки на сторінку страви:

```
    public function actionRedirect() {
        header('Location: /recipes');
    }
```

Метод, за допомогою якого можна додавати нові інгредієнти:

```
    public function actionAdd() {
        $allIngredients = Stock::getIngredientList();
        require_once(ROOT . '/views/recipe/add.php');
    }
```

Метод, за допомогою якого можна видаляти страви:

```
    public function actionDelete($id) {
        Recipe::deleteRecipeById($id);
        header('Location: /recipes');
    }
```

Контролер складу:

```
class StockController{
```

Метод перегляду сторінки складу:

```
public function actionView(){
        $ingredientList = array();
        $ingredientList = Stock::getStockIngredientList();

        include_once(ROOT . '/views/stock/view.php');
        return $ingredientList;
          }
```

Метод додавання інгредієнту на склад:

```
        public function actionAdd(){
         if (isset($_POST['submit'])) {

                    $id = $_POST['id'];
               $count = $_POST['count'];

          $id = Stock::addIngredientToStock($id, $count);

                    header("Location: /stock/add");
               }
               else {
                    $ingredientList = array();
$ingredientList = Stock::getNonStockIngredientList();
               }
               include_once(ROOT . '/views/stock/add.php');
          }
```

Метод оновлення таблиці інгредієнтів на складі:

```
          public function actionUpdate($id) {

                if (isset($_POST['submit'])) {
                    $count = $_POST['count'];
               Stock::addIngredientToStock($id, $count);
                    header("Location: /stocks");
               }
               else {
                    $ingredient = array();
                    $ingredient = Stock::getIngredient($id);
          include_once(ROOT . '/views/stock/update.php');
               }
          }
```

Метод видалення інгредієнта зі складу:

```
          public function actionDelete($id) {
               $ingredientList = array();
$ingredientList = Stock::addIngredientToStock($id, NULL);
               header("Location: /stocks");
          }
```

Модель Recipe:

```
class Recipe {
```

Метод отримання списку страв з бази даних:

```
        public static function getRecipeList() {
            $db = Db::getConnetion();

            $recipeList = array();
            $result = $db -> query('SELECT * FROM recipe');
            $i = 1;

            while($row = $result->fetch()) {
                $recipeList[$i]['id'] = $row['ID'];
                $recipeList[$i]['name'] = $row['name'];
                $recipeList[$i]['description'] =
$row['description'];
                $recipeList[$i]['image'] = $row['image'];
                $i++;
            }
            return $recipeList;
        }
```

Метод отримання потрібного рецепта з бази даних:

```
        public static function getRecipeById($id) {
            $db = Db::getConnetion();
    $result = $db -> prepare('SELECT * FROM recipe where id =
:id');
            $result->bindParam(':id', $id, PDO::PARAM_INT);
            $result->execute();

            return $result->fetch();
        }
```

Другий рядок – це отримання підключення до бази даних. 3–4 рядки – формування самого запиту до бази. На п’ятій відбувається виконання, а на шостій повернення результату цього запиту.

Метод видалення рецепту з бази даних:

```
        public static function deleteRecipeById($id) {
            $db = Db::getConnetion();
            $recipeList = array();
            $result = $db -> prepare('DELETE from recipe where
id = :id');
            $result->bindParam(':id', $id, PDO::PARAM_INT);
            $result->execute();

            $result = $db -> prepare('DELETE from reci-
pe_ingredient where ID_recipe = :id');
            $result->bindParam(':id', $id, PDO::PARAM_INT);
            $result->execute();
```

```
            return TRUE;
```

Модель інгредієнтів:

```
class RecipeIngredient {
```

Метод отримання з бази даних необхідних інгредієнтів для рецепту страви:

```
    public static function getListIngredientByRe-
cipe($recipe_id){

                $db = Db::getConnetion();
                $ingredientList = array();
                $result = $db->prepare('select reci-
pe_ingredient.count as count, stock.ID as id, stock.name as
name, stock.measure as measure, stock.count as stock_count from
recipe_ingredient join stock on stock.ID = reci-
pe_ingredient.ID_ingredient where ID_recipe = :recipe_id');

                $result->bindParam(':recipe_id', $recipe_id,
PDO::PARAM_INT);
                $result->execute();
                $i = 0;
                while($row = $result->fetch()){
                    $ingredientList[$i]['id'] = $row['id'];
                    $ingredientList[$i]['name'] = $row['name'];
                    $ingredientList[$i]['count'] = $row['count'];
                    $ingredientList[$i]['measure'] =
$row['measure'];
                    $ingredientList[$i]['stock_count'] =
$row['stock_count'];
                        $i++;
                        }
                    return $ingredientList;
                    }
```

Модель складу:

```
class Stock {
```

Метод отримання списку інгредієнтів з бази даних:

```
    public static function getIngredientList() {
            $db = Db::getConnetion();
            $productList = array();
            $result = $db -> query('SELECT * FROM stock');
            $i = 0;
            while($row = $result -> fetch()){
```

```
                $productList[$i]['id'] = $row['ID'];
                $productList[$i]['name'] = $row['name'];
                $productList[$i]['measure'] =
$row['measure'];
                $productList[$i]['count'] = $row['count'];
                $i++;
            }
            return $productList;
        }
```

Метод отримання одного інгредієнта з бази даних:

```
        public static function getIngredient($id) {
            $db = Db::getConnetion();
            $productList = array();
            $sql = 'SELECT * FROM stock WHERE id = :id';
            $result = $db->prepare($sql);
            $result->bindParam(':id', $id, PDO::PARAM_INT);
            $result->setFetchMode(PDO::FETCH_ASSOC);
            $result->execute();

            return $result->fetch();
        }
```

Метод отримання інгредієнтів, які є на складі:

```
        public static function getStockIngredientList() {
            $db = Db::getConnetion();
            $productList = array();
            $result = $db -> query('SELECT * FROM stock where
stock.count is not NULL');
            $i = 0;
            while($row = $result -> fetch()){
                $productList[$i]['id'] = $row['ID'];
                $productList[$i]['name'] = $row['name'];
                $productList[$i]['measure'] =
$row['measure'];
                $productList[$i]['count'] = $row['count'];
                $i++;
            }
            return $productList;
        }
```

Метод отримання продуктів, яких немає на складі:

```
        public static function getNonStockIngredientList() {
            $db = Db::getConnetion();
            $productList = array();
```

```
                $result = $db -> query('SELECT * FROM stock where
stock.count is NULL or stock.count = 0');

                $i = 0;
                while($row = $result -> fetch()){
                     $productList[$i]['id'] = $row['ID'];
                     $productList[$i]['name'] = $row['name'];
                     $productList[$i]['measure'] =
$row['measure'];
                     $productList[$i]['count'] = $row['count'];
                     $i++;
                }
                return $productList;
           }
```

Метод створення інгредієнта:

```
      public static function createIngredient($options) {
                $db = Db::getConnetion();
                $sql = 'INSERT INTO stock '
                          . '(name, measure)'
                          . 'VALUES '
                          . '(:name, :measure)';
                $result = $db->prepare($sql);
                $result->bindParam(':name', $options['name'],
PDO::PARAM_STR);
                $result->bindParam(':measure', $op-
tions['measure'], PDO::PARAM_STR);

                if ($result->execute()) {
                     return $db->lastInsertId();
                }
                return 0;
           }
```

Метод додавання інгредієнту до складу:

```
           public static function addIngredientToStock($id,
$count) {
                $db = Db::getConnetion();
           $sql = 'UPDATE stock SET stock.count = :count_ WHERE
id = :id';
                $result = $db->prepare($sql);
                $result->bindParam(':count_', $count,
PDO::PARAM_INT);
                $result->bindParam(':id', $id, PDO::PARAM_INT);
                $result->execute();
                return TRUE;
           }
```

Метод видалення інгредієнту:

```
        public static function deleteIngredient($id) {

        $db = Db::getConnetion();
                $sql = 'DELETE from stock where id = :id';
                $result = $db->prepare($sql);
                $result->bindParam(':id', $id, PDO::PARAM_STR);
                $result->execute();
            $sql = 'DELETE from recipe_ingredient where
ID_ingredient = :id';
                $result = $db->prepare($sql);
                $result->bindParam(':id', $id, PDO::PARAM_STR);
                $result->execute();

                return TRUE;
            }
```

3.2 Огляд робочого додатку

Основними функціями даного додатку є:

– заповнення вибору страв;

– при виборі страви повинен видаватися її рецепт;

– розраховувати кількість порцій на потрібну кількість дітей;

– наявність складу для збереження продуктів;

– можливість порівняння потрібних продуктів та наявних;

– можливість додавати або віднімати продукти зі складу.

Створений додаток будується на аналізі існуючих письмових аналогів, що дає змогу зрозуміти, що необхідно та можна зробити краще.

Першим, що бачить користувач, буде головна сторінка, головною метою якої є показати основні страви та надати необхідну інформацію про їх складову (рис. 18). Тут користувач може обрати необхідну страву, щоб побачити потрібні інгредієнти та їх кількість, щоб надалі приготувати завтрак, обід або вечерю.

На головній сторінці присутня галерея з фото страв та короткими підписами під ними і кнопка додавання нової страви. Також зображено вкладки складу та інгредієнтів.

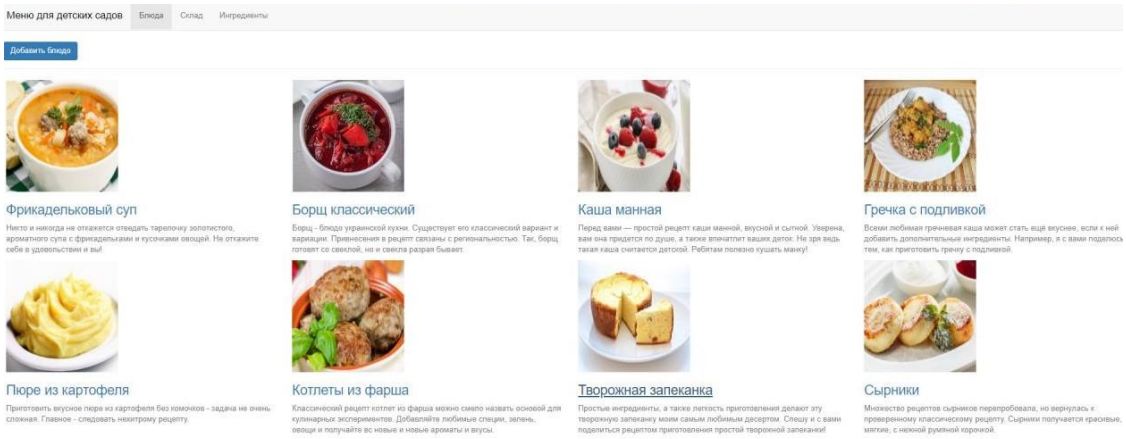


Рисунок 18 – Головна сторінка

При натисканні кнопки додавання страви ми переходимо на заповнення форми нової страви (рис. 19 та рис. 20):

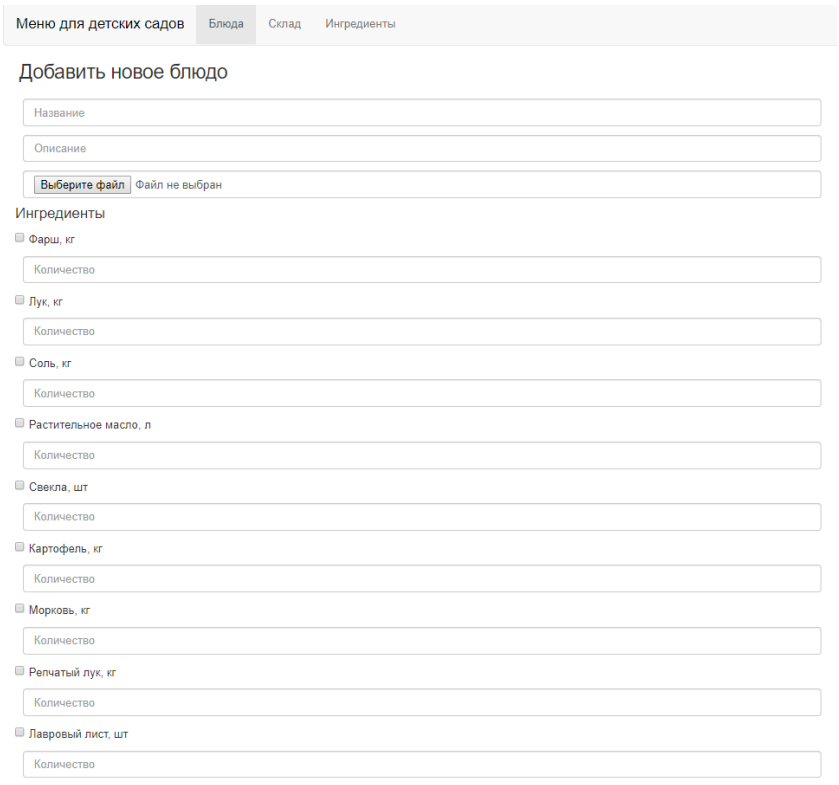


Рисунок 19 – Форма для заповнення нової страви

Рисунок 20 – Форма для заповнення нової страви

При натисканні кнопки «Додати» нова страва з’являється на головній сторінці. Вкладка «Склад» виводить усі види продуктів та їх наявну кількість. Також має такі кнопки, як Відредагувати кількість продукту, Видалити його та Додати нові продукти. Все це зображено на рис. 21.

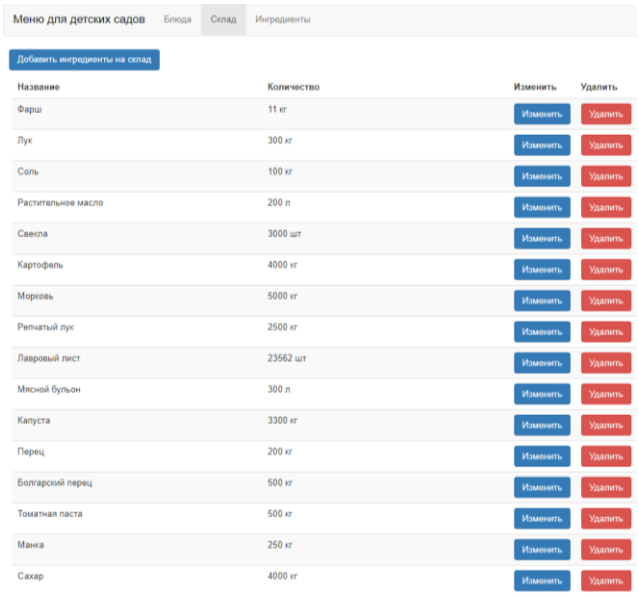


Рисунок 21 – Вкладка Склад

Якщо ми захочемо відредагувати кількість продукта, нам потрібно натиснути на кнопку «Відредагувати» та відкриється форма для внесення змін (рис. 22):

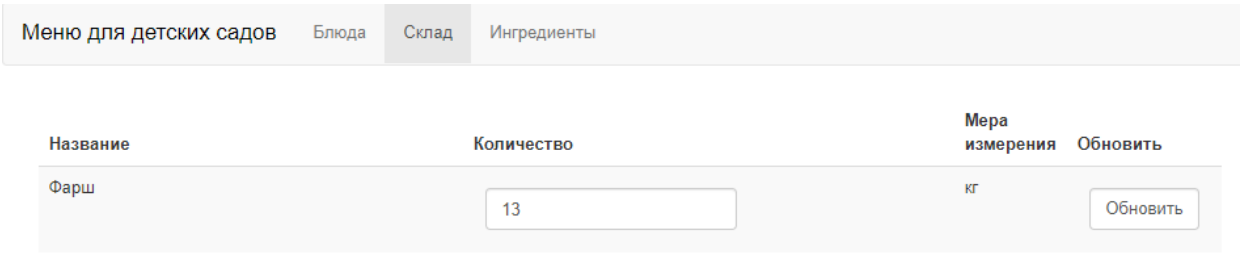


Рисунок 22 – Редагування кількості продукту

Наступною вкладкою є вкладка «Інгредієнти». Тут ми маємо список усіх можливих інгредієнтів та міра виміру кожного продукту. Також напроти кожного найменування є кнопка Видалити. А на початку сторінки присутня кнопка «Додати новий інгредієнт» (рис.23):

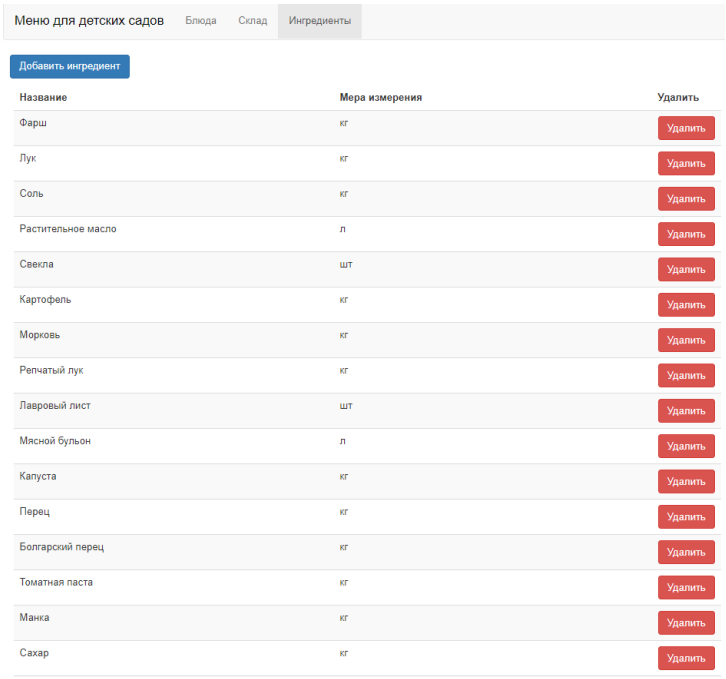


Рисунок 23 – Вкладка Інгредієнти

Щоб додати новий інгредієнт, потрібно натиснути на кнопку з відповідним написом, тоді відкриється форма для заповнення (рис.24):

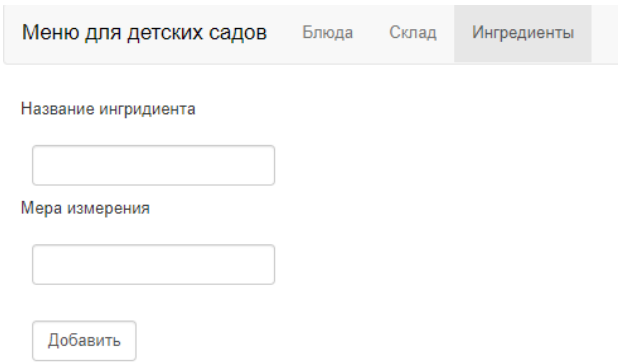


Рисунок 24 – Форма заповнення нового інгредієнта

ВИСНОВКИ

Отже, дослідивши предмету область, можна зробити висновок, що на даний момент не існує програмних аналогів даному додатку.

Для отримання відповідей на запитання про актуальність було проведено соціальне опитування робітників дитячих садків і це дало змогу зрозуміти, що даний додаток значно може спростити та покращити роботу персоналу.

До того ж XXI століття це час автоматизації, тому навіть додаток з створення меню та підрахунку кількості необхідних продуктів надає можливість облегшити та поліпшити роботу багатьох дитячих садків. Кінцевим результатом буде програмний додаток, який використовується на персональному комп’ютері.

Кожний етап розробки обговорюється з замовником та тестується , після чого складається звіт.

Додаток розрахований на довгочасне використання, з можливістю внесення додаткових функцій та керування версіями.

При вдалому використанні впродовж деякого часу додаток буде впроваджено в більшість дитячих садків Одеської області.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. История появления детских садов. URL: http://lubopitnie.ru/istoriya–poyavleniya–detskih–sadov/. (дата звернення 25.03.2019)
2. Лабораторія СЕТ. Термін «application» українською URL: http://www.setlab.net/?view=application–term. (дата звернення 25.03.19)
3. MySQL. Руководство администратора MySQL. Administrator's Guide. М.: Издательский дом «Вильямс», 2005. С. 624.
4. HTML – язык разметки страницы. URL: http://html.find–info.ru/html/articles/article_8.htm. (дата звернення 26.03.19)
5. IIT Lectures. Каскадные таблицы стилей. URL: https://iit–web–lectures.readthedocs.io/ru/latest/www/css.html. (дата звернення 27.03.19)
6. Professor Web. Библиотека jQuery. URL: https://professorweb.ru/my/javascript/jquery/level1/jquery_index.php (дата звернення 26.03.19)
7. Habr. Почему мы выбрали Electron. URL: https://habr.com/ru/post/439946/. (дата звернення 27.03.19)
8. A Medium Corporation. Начало работы с Nodejs. URL: https://medium.com/devschacht/node–hero–chapter–1–239f7afeb1d1. (дата звернення 28.03.2019)
9. Metanit.com. Введение в Nodejs. URL: https://metanit.com/web/nodejs/1.1.php (дата звернення 27.03.2019)
10. Ильин В.В. Моделирование бизнес–процессов. Практический опыт разработчика. Вильямс. 2006. 365 с.
11. Катренко А.В. Управління ІТ-проектами. Львів: «Новий світ 2000», 2013. 550с.
12. Управление проектом. Основы проектного управления: учебник /под ред. проф. М. Л.Разу.:М. : КНОРУС, 2006. 450 с.

13. Черемных С.В., Семенов И.О., Ручкин В.С. Моделирование и анализ систем. IDEF-технологии. Учебник-практикум: М.: Финансы и статистика, 2006. 188 с.
14. A Guide to the Project Management Body of Knowledge. Project Management Body Of Knowledge. Project Management Institute, Four Campus Boulevard, Newtown Square, PA 19073–3299 USA. 2004. 792 p.
15. Грекул В.И. и др. Проектирование информационных систем. Курс лекций. Интернет-Университет Информационных Технологий, М.2005. 624 с.
16. Мазур, И. И. Управление проектами : учеб. пособие / И. И. Мазур, В. Д. Шапиро, Н. Г. Ольдерогге; под общ. ред. И. И. Мазура.: М. : Омега-Л, 2005. 412 с.
17. James Rumbaugh, Ivar Jacobson, Grady Booch 1999. The unified modeling language reference manual. Addison Wesley Longman Inc. ISBN 0-201-30998-X. 732 p.