МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Кафедра комп'ютеризованих систем управління

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри

_____Олександр ЛИТВИНЕНКО

«___» _____2022 р.

ДИПЛОМНИЙ ПРОЄКТ

(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬОГО СТУПЕНЯ
"БАКАЛАВР"

Тема: Програмна система для організації роботи відділу технічної підтримки користувачів програмного забезпечення інвестиційної компанії

Виконавець: Анастасія АНДРОЩУК

Керівник: Євген Артамонов

Нормоконтролер: Євгеній ТУПОТА

Київ 2022

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет кібербезпеки, комп'ютерної та програмної інженерії
Кафедра комп'ютеризованих систем управління
Спеціальність 123 "Комп'ютерна інженерія"
(шифр, найменування)

Освітньо професійна програма «Системне програмування»
Форма навчання очна

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____Олександр ЛИТВИНЕНКО

«_____»_____2022 р.

ЗАВДАННЯ
на виконання дипломного проєкту

Андрощук Анастасії Олександрівни
(прізвище, ім'я, по батькові випускника в родовому відмінку)

1. Тема проєкту (роботи): Програмна система для організації роботи відділу технічної підтримки користувачів програмного забезпечення інвестиційної компанії
затверджена наказом ректора від "15" лютого 2022 року № 251/ст

2. Термін виконання роботи: з 16.05.2022 до 18.06.2022

3. Вихідні дані до проєкту (роботи): завдання га розробку *CRM* системи, вимоги до інформаційно-пошукового модулю в системі

4. Зміст пояснювальної записки (перелік питань, що підлягають розробці):
1) визначення необхідності використання контакт-центрів;
2) проєктування бази даних системи;
3) розробка програми та описання програмного забезпечення.

5. Перелік обов'язкового графічного матеріалу:
1) схема взаємозв'язку елементів в *CRM*-стратегії;
2) загальна схема зв'язків між таблицями бази даних, які використовуються в програмі;
3) вікно «Список документів: клієнти»;
4) форма редагування параметрів вибірки;
5) схема алгоритму зчитування даних з форми для автоматичного формування запиту.

6. Календарний план

№ п/п	Етапи виконання дипломного проєкту	Термін виконання етапів	Примітка
1	Провести огляд літератури за темою дипломного проєкту та аналіз існуючих систем.	16.05.22 -20.05.22	
2	Зробити вибір компонентів системи.	21.05.22-25.05.22	
3	Розробити структуру програмних засобів системи.	26.05.22-05.06.22	
4	Розробити програмні засоби.	06.06.22-10.06.22	
5	Провести відладку програмних засобів на модельному зразку.	11.06.22-13.06.22	
6	Написати пояснювальну записку.	14.06.22-20.06.22	

7. Дата видачі завдання « 16 » травня 2022 р.

Керівник дипломного проєкту_____ Євген АРТАМОНОВ
(підпис)

Завдання прийняв до виконання_____ Анастасія АНДРОЩУК
(підпис студента)

РЕФЕРАТ

Пояснювальна записка до дипломного проєкту “Програмна система для організації роботи відділу технічної підтримки користувачів програмного забезпечення інвестиційної компанії”, 58 сторінок, 20 рисунків, 2 таблиці, 14 літературних джерел, 1 додаток.

INTERNET-ТЕХНОЛОГІЇ, *CRM*-СИСТЕМИ, *PHP*, БАЗИ ДАНИХ, СУБД *MS SQL*

Об’єкт дослідження – контакт-центр відділу технічної підтримки.

Предмет дослідження – програмна система для організації роботи відділу технічної підтримки користувачів програмного забезпечення інвестиційної компанії.

Результати дипломного проєкту рекомендується використовувати при розробці програмних систем відділів технічної підтримки користувачів та у навчальному процесі.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ, ТЕРМІНІВ

ВСТУП

Про CRM завжди пишуть такою незрозумілою мовою, начебто це адронний колайдер, а насправді це просто програма управління бізнесом. Щоправда, дуже розумна настільки, що допомагає збільшити прибуток на 40-50%, знизити витрати і вдвічі прискорити обробку заявок. Але що таке CRM і що вона робить? Як розробник власної системи S2, "відповідаю популярно". Отже, CRM-системи що це простими словами?

Як працює CRM-система?

Визначення або абревіатура CRM розшифровується як Customer Relationship Management, тобто управління відносинами з клієнтами. CRM за допомогою автоматизації процесів допомагає ефективніше вибудовувати діалог з покупцем, не допускати помилок у роботі та в результаті продавати йому більше.

"Хмара" чи "коробка"? Розбираємо, які бувають

CRM-системи

Як це виглядає у CRM? Уявіть таблицю Excel з вашою клієнтською базою, але тільки при натисканні на ім'я клієнта відкривається зручна картка, в якій міститься вся хронологія роботи з ним - від першого дзвінка до покупки. Тут можна прослухати дзвінки, переглянути історію покупок, створити документи за шаблоном, написати e-mail або sms, поставити завдання.

Коли клієнт дзвонить вам, CRM пропонує відкрити його картку, і ви одразу вітаєте його на ім'я. Навіть якщо раніше цього покупця вів інший менеджер, ви легко відповісте на його запитання без жодних «уточню та передзвоню». CRM сама надішле клієнту sms-повідомлення про статус замовлення та нагадає про зустріч. У результаті ви заощаджуєте час - і свій, і клієнта. А значить, робите його більш лояльним та налаштованим на покупку.

Автоматизаціястоїть останньою, але не тому, що вона найменш важлива. Навпаки, саме вона допомагає поставити продажі на автомат — щоб усе

працювало швидко та чітко, а помилки «людського фактору» було зведено до мінімуму. CRM бере всю рутину на себе: формує документи за шаблоном, ставить завдання менеджерам на кожному етапі угоди, відправляє SMS клієнтам, створює онлайн-звіти за всіма показниками, через вбудований калькулятор розраховує вартість послуг, а також відстежує важливі дати (нагадує продовжити договір, виставити рахунок на оплату, запропонувати сервіс та ін.).

Як працює автоматизація процесів

Завдяки CRM менеджери припускаються менше помилок, а значить, продають більше і частіше. А керівнику стає легше керувати компанією: він витрачає менше часу на контроль та отримує більше ресурсів, щоб розвивати бізнес.

Який ефект має CRM?

"Маршрут ТВ" після впровадження CRM

Якщо говорити коротко, то CRM допомагає заробляти більше та легше, а значить, з нею ви зможете успішніше розвивати свій бізнес. У 2015 році консалтингова компанія Capterra опитала 500 компаній та з'ясувала, що після впровадження CRM їх прибуток зріс від 25 до 35%. А в одного нашого клієнта – компанії «Маршрут ТВ» – після впровадження CRM прибуток збільшився аж у 1,5 раза!

Завдяки CRM зростають та інші показники. В своїхкейсах впровадженнями описуємо, як S2 допомогла нашим користувачам збільшити прибуток, підвищити продуктивність співробітників і прискорити роботу. Ми зібрали найкращі результати клієнтів в одній інфографіці. Клацніть на посилання, щоб перейти до повного тексту кейсу:

За кожною з цих цифр ховається вигода реальної компанії, і разом вони дозволяють бізнесу швидше розвиватися:

1. Прискорення роботи менеджерівозначає, що вони зможуть більше часу приділяти власне продажам, а значить підвищити обсяги угод.

2. Зростання продажів та підвищення середнього чекадають збільшення прибутку.

3. Підвищення прибуткудає можливість більше ресурсів вкладати у зростання бізнесу.

4. Розвиток бізнесудає ще більший прибуток, а також можливість керівнику стати головою успішної компанії.

РОЗДІЛ 1
ВИЗНАЧЕННЯ НЕОБХІДНОСТІ ВИКОРИСТАННЯ КОНТАКТ-ЦЕНТРІВ

Як зрозуміти, чи потрібна мені CRM?

CRM для вас, якщо

У вас є відділ продажу, і робота з клієнтами заснована на телефонних дзвінках, листах та зустрічах. Історію спілкування потрібно зберігати в одному місці, щоб постійно залучати нові ліди та вибудовувати з ними довгострокові стосунки. Наприклад, CRM-система з інтегрованою телефонією ідеально підійде для інтернет-магазинів чи оптових компаній.

CRM не підійде, якщо

Ви власник роздрібного магазину, і не зацікавлені у вибудовуванні довгострокових відносин з клієнтами, не дзвоніть і, не пишете листів, не надсилаєте SMS-повідомлень. Або ж, якщо ви працюєте за довгостроковими контрактами, зав'язаними на особистих знайомствах. Тут не допоможе жодна програма, прибуток залежить виключно від досвіду менеджера.

Простий приклад: Кирило та CRM

Щоб було зрозуміло, як працює CRM у конкретній компанії, наведу приклад. Кирило керуєфірмою із встановлення вікон. Раніше клієнтів мало, і все було просто: у кожного потрібно прийняти замовлення, виїхати на виміри, погодити вартість, отримати оплату, поставити вікна. Але потім клієнтів стало не 3, а 33. І почалося... Одному клієнту забули передзвонити, іншому не виїхали на виміри, третьому не відправили розрахунок вартості, а у четвертого прийняли оплату ще місяць тому, а вікна все ще не встановили. Покупці почали йти до конкурентів, а витрати на нових співробітників перестали себе окупати. Кирило не встигав контролювати кожен крок співробітників і зрозумів, що настав час щось міняти.

Він впровадив CRM-систему, і тепер взаємодія з кожним клієнтом відбувається за єдиним стандартом:

1. Коли покупець залишає заявку на сайті, у CRM з'являється картка угоди, де вказано етап воронки продажу: «Перший дзвінок». CRM ставить завдання менеджеру: "Перезвонити клієнту протягом 15 хвилин". Якщо завдання буде прострочено, CRM повідомить керівника.

2. Менеджер дзвонить клієнту прямо з CRM, фіксує підсумки переговорів у картці угоди та переводить її на етап «Заміри». CRM автоматично створює завдання для замірника: «Виїхати на виміри угоди [дата, час]».

3. Після виїзду фахівець із вимірів прикріплює до картки угоди документ із розмірами та ТЗ, переводить угоду на етап «Узгодження».

4. Відповідальний менеджер отримує завдання: "Розрахувати вартість та зателефонувати клієнту протягом 2 годин". Він фіксує розрахунки в CRM та дзвонить.

5. Угода переходить на етап Оплата, CRM автоматично формує документ за шаблоном, куди вставляє ім'я, адресу, назву послуги, суму, реквізити оплати. Менеджеру залишається надіслати документ клієнту, отримати оплату та перевести угоду на останній етап – «Установка».

6. Фахівець з монтажу відразу ж отримує автоматичне повідомлення про те, що він має встановити вікна угоди до певного терміну.

7. Керівник тим часом відстежує онлайн-звіти: скільки угод закрито, скільки дзвінків здійснив кожен менеджер, яка сума та кількість угод, яка конверсія заявок, з яких джерел приходить найбільше клієнтів та ін.

Отже, що дає CRM?

Програма допомогла Кирилу систематизувати дані про клієнтів та угоди, співробітники перестали забувати про справи та зривати терміни. Підвищилася конверсія заявок у продажу, клієнти стали лояльнішими, а прибуток зріс. Тепер Кирилу необов'язково бути присутнім в офісі, щоб усі працювали, як треба, і він може більше часу приділяти стратегії компанії.

Які проблеми вирішує

CRM захоплює заявки із сайту, призначає відповідальних менеджерів, ставить їм завдання на кожному етапі продажу. Якщо завдання прострочене, керівник моментально дізнається про це. Ви більше не втратите жодного клієнта.

Важко проаналізувати продажі?

Кількість нових лідів, сума угод, кількість дзвінків та зустрічей - CRM видасть наочні звіти з усіх бізнес-процесів. CRM дасть звіт про кожного співробітника та допоможе обчислити ледарів у відділі продажів.

Плинність кадрів відбивається на продажах?

Історія роботи з клієнтами зібрана в CRM: новий менеджер може одразу увійти до курсу справи. Йому потрібно просто виконувати завдання, які ставить програма, перекладати угоду на нові етапи та грамотно спілкуватися з клієнтами.

Менеджер йде та забирає базу клієнтів?

Налаштуйте права доступу в CRM так, щоб менеджери бачили тільки своїх клієнтів - тепер ніхто, крім вас, не матиме доступу до повної клієнтської бази, і не вкраде її.

то має бути у CRM?

Спершу потрібно зрозуміти, чого ви хочете від CRM-системи. Розробники постійно розширюють функціонал програм: додають нові інтеграції, елементи гейміфікації, сканування візиток та інше. Але часто компанії не використовують ці опції, і, впровадивши таку CRM, ви переплатите за надлишковий функціонал.

Проте єнабір функцій, які обов'язково повинні бути присутніми в CRM:

1. Модуль обліку клієнтів,у якому зберігається вся історія взаємодії із клієнтами.

2. Модуль для керування продажамиз наочною вирвою продажів, де

зазначено, якому етапі перебуває кожна угода.

3. Автоматизація бізнес-процесів,яка дозволяє не просто ставити завдання, а й відправляти sms-розсилки, змінювати дані про об'єкти, нагадувати про наближення важливих дат — наприклад, про термін закінчення договору або день народження.

4. Аналітика та звітиу реальному часі як наочних графіків і діаграм, і навіть таблиць у детальними даними.

5. Управління завданнямивибудуване таким чином, щоб керівник моментально отримував повідомлення про виконані та прострочені співробітниками справи.

6. Інтеграціяз поштою,сайтомIP-телефонією,щоб усі вхідні заявки, яким каналом вони б не надійшли, відразу фіксувалися в CRM.

7. Інтерфейс програмування API,який дозволяє налаштувати інтеграцію з 1С, корпоративним ПЗ, мобільними та іншими програмами.

1.1. Порівняння функціоналу і можливостей сучасних CRM-систем

CRM-система – це спосіб управління взаємовідносинами з клієнтами та оптимізації бізнес-процесів. Ключовою складовою цього підходу є CRM-система – спеціальне програмне забезпечення для організації роботи з лідами, відстеження дій клієнтів та автоматизації комунікацій.

CRM-система збирає дані про кожного клієнта на одній панелі управління. За допомогою цієї інформації ваша команда може відстежуватишлях покупцята робити релевантні пропозиції на кожному етапі. В результаті дохід компанії може зрости до41%на одного торгового представника, а цикл продажів скоротиться на8-14%.

CRM допомагає визначити інтереси та переваги ваших клієнтів. Це дозволяє надати персоналізований досвід та створювати релевантні маркетингові кампанії. ЗгіднодослідженнямCapterra, компанії, які використовуютьCRM маркетинг, відзначають збільшенняутримання клієнтівта підвищення задоволеності на 47%.

Бажаєте дізнатися, як управління взаємовідносинами з клієнтами допомагає досягти таких результатів? Тоді читайте далі. Настав час дізнатися, які завдання вирішують CRM-системи.

Ринок програмного забезпечення CRM повний пропозицій. Існує безліч сервісів для малих, середніх та великих компаній із різних сфер бізнесу. Проте є низка основних завдань, які має виконувати кожна CRM-система:

• Консолідація даних клієнтів.CRM-система повинна збирати контакти вашихлідівта покупців, їх демографічні дані та іншу інформацію, забезпечуючи до неї легкий доступ.

• Відстеження взаємодій та активності.CRM-системи дозволяють відстежувати комунікацію з клієнтами в чатах з менеджерами, телефоном, email та іншими каналами.

• Вимірювання продуктивності та продуктивності.Хороша CRM-система дозволяє отримувати звіти з детальними даними щодо ефективності взаємодії компанії з клієнтами.

• Автоматизація рутинних процесів.Автоматизація маркетингута продажів – це основа будь-якої CRM-системи.

CRM-система може вирішувати різні завдання, із якими стикаються компанії. Давайте розглянемо основні види програмного забезпечення управління взаємовідносинами з клієнтами.

Види CRM-систем

Універсальна CRM-система – рідкість. Зазвичай одне програмне забезпечення виконує одне завдання краще, ніж інші. Залежно від своїх можливостей, будь-яка CRM-система потрапляє в одну з наступних категорій:

• Операційні системи CRM.Допомагають виконувати повсякденні процеси вашої компанії та автоматизувати рутинні завдання.

• Аналітичні системи CRM.Це величезні бази даних з детальною інформацією про ваших клієнтів та бізнес-процеси.

• Колективні системи CRM.Допомагають підвищити ефективність взаємодій між різними відділами вашої компанії.

Тепер, коли ми ознайомилися із завданнями та функціями маркетингу

взаємин із клієнтами, настав час знайти оптимальні варіанти CRM-систем для досягнення різних бізнес-цілей.

Кожне бізнес-завдання потребує конкретного рішення. Те, що працює для досягнення маркетингових цілей, може не підходити для обслуговування клієнтів. Тому ми розділили CRM-системи на групи відповідно до потреб бізнесу, які вони задовольняють. Дотримуйтесь нашого списку ефективних інструментів для продажу, маркетингу, малого бізнесу, управління соціальними мережами та агентств нерухомості.

CRM-системи для маркетингу

Якщо ви розглядаєте можливість використання CRM-системи для своїх маркетингових цілей, ви можете ігнорувати складні вирви або вбудовані виклики. Натомість зосередьтеся на подробиці статистики взаємодії з клієнтами та надійності інструментів автоматизації маркетингових процесів. Ознайомтеся із CRM-системами для маркетологів.

SendPulse

SendPulse – відмінний вибір для тих, хто шукає інструмент з ефективним набором функцій автоматизації маркетингу. Для залучення лідів ви можете створювати професійні мультиканальніформи передплати,pop-upта чат-ботів у соціальних мережах. Конструюйте автоматизовані ланцюжки повідомлень длявирощування лідів, своєчасних відповідей на взаємодії клієнтів та економії часу на подальшій роботі. Також сервіс допомагає вибудовувати сценарії повідомлень чат-ботів для комунікації з користувачами Telegram, ВКонтакте і Facebook.

CRM-система від SendPulseДобре підійде тим, хто ще не намагався використовувати CRM і веде облік продажів на папері. Сервіс – абсолютно безкоштовний. Інтуїтивний інтерфейс в єдиному стилі з іншими можливостями компанії допоможе легко розібратися в системі, якщо ви раніше використовували SendPulse.

У нашій CRM ви можете зберігати та поповнювати інформацію про клієнтів, створювати угоди та контролювати процес їх укладання, додавати потрібні вам статуси, щоб візуалізувати процеси на канбан-дошці.

Натиснувши на карту угоди, ви отримаєте повну інформацію по ній: дані про клієнта, про угоду (коли створена, вручну або з автоматичного джерела, історію її переміщення по етапах, фахівця, відповідального за її укладання), коментарі. Ви можете її редагувати та додавати потрібні поля, залишати коментарі, зв'язуватися з клієнтом через email або чат-бот, якщо він був ініціатором спілкування через цей канал. Теги допоможуть швидко знайти потрібний контакт та сегментувати клієнтів.

Якщо ви надсилаєте email розсилки через SendPulse та використовуєте чат-ботів, ви можете робити автоматичні розсилки прямо з CRM по тригеру. Це може бути створення угоди, тобто, коли користувач натискає кнопку "Купити" в чаті з вашим ботом, або перехід угоди на новий етап, наприклад, "Чекає на оплату" або "Готов до відправки".

Ви також можете додавати членів своєї команди до CRM і призначати відповідальних за кожну угоду. Колеги, яких ви додасте до сервісу, зможуть бачити та працювати з усіма угодами, але ви можете обмежити їх доступ до інших сервісів SendPulse.

Neaktor

Це зручна у використанні CRM-система. Її можна адаптувати майже під будь-яку сферу бізнесу. Для роботи в сервісі вам не знадобляться особливі навички. За допомогою простого та зрозумілого візуального редактора необхідно виконати налаштування процесів для свого бізнесу.

Neaktor дозволяє закривати завдання будь-якого відділу компанії. Ви можете автоматизувати бізнес-процеси у продажах, маркетингу, закупівлі, постачанні, бухгалтерії, виробництві, IT і так далі. Сервіс допомагає контролювати виконання завдань та ефективно розподіляти роботу незалежно від розташування команди. Існує корпоративний чат для комунікації в режимі реального часу.

Серед плюсів варто відзначити велику кількість готових шаблонів бізнес-процесів. Це прискорює та спрощує роботу. У Neaktor ви знайдете шаблони комерційних пропозицій, рахунків, листів, карток постачальників та багато іншого. Є чотири тарифні плани.

Безкоштовний тариф:доступний необмежену кількість часу, можна підключити до семи користувачів.

Платний тариф:ціна стартує від $3 на місяць за одного користувача.

EnvyCRM

Інтуїтивно зрозумілий інтерфейс сервісу дозволяє відразу розпочати роботу. EnvyCRM допомагає оптимізувати бізнес-процеси, контролювати виконання завдань та керувати взаєминами з клієнтами. Можна підключати IP-телефонію, інтегруватися із сайтом, соцмережами та email сервісом. Є можливість інтеграцій з Facebook, ВКонтакті, Google Analytics, Telegram, Яндекс.Диск, 1С-Бітрікс, WordPress, Joomla, SendPulse, MailChimp та іншими сервісами.

Серед переваг користувачі відзначають високу швидкість адаптації працівників через простоту інтерфейсу. З плюсів часто згадується автоматична постановка завдань, можливість управління угодою кнопками, що в рази спрощує роботу, відправка нагадувань, формування звітів, наявність більш ніж сорока інтеграцій, “автопідвантаження” завдань по клієнтам.

Сервіс надає три тарифні плани.

Безкоштовний тариф:відсутня, є пробний період на 7 днів.

Платний тариф:вартість одного місяця стартує від $9 за умови покупки на три місяці.

CRM-системи для автоматизації продажу

Таке програмне забезпечення орієнтоване оптимізацію складних процесів реалізації товарів та послуг. CRM-система спрощує залучення лідів, відстеження пересуванняпотенційного клієнтаповирві продажівта скорочує цикл продажів.

Важливо, щоб сервіс автоматизував рутинні завдання та мав потужні інструменти аналітики для відстеження продуктивності та виявлення слабких місць у процесі укладання угод. Ознайомтеся з трьома популярними системами CRM, які ми підібрали для вас.

Bitrix24

У Bitrix24 зареєстровано понад 7 мільйонів компаній. Сервіс допомагає вести клієнтів з маркетингової вирви, автоматизувати продажі та контролювати

канали комунікацій. Дані з кожної угоди можна переглянути у спеціальних картках, де система збирає всю інформацію про клієнтів, включаючи SMS, записи дзвінків, зустрічі та інше. У них є можливість створення чату зі співробітниками якщо необхідно щось обговорити.

Bitrix24 дозволяє підключати телефонію чи орендувати номер. Система інтегрується з сайтами, Яндекс-чатом, ВКонтакте, Facebook та іншими сервісами. Також ви можете прямо з CRM надсилати своїм клієнтам листи. В системі фіксується отримання та прочитання кожного email, а також зберігається історія листування.

Нижче ви бачите розділ CRM-аналітики в Bitrix24. Тут відображається динаміка продажів, ефективність роботи менеджерів, інформація про клієнтів та інші дані.

Зверніть увагу, що Bitrix24 - це сервісавтоматизації бізнесуз величезними можливостями та CRM-система є лише одним із інструментів. Тому купити її можна лише у комплекті з усім функціоналом.

Безкоштовний тариф:мінімальний набір інструментів доступний необмежену кількість часу.

Платний тариф:від $12 на місяць.

Zendesk Sell

Zendesk Sell – це модель операційної CRM-системи, яка дозволяє телефонувати клієнтам, надсилати їм листи та планувати зустрічі в рамках одного сервісу. Ви можете автоматизувати обмін даними прямо всередині системи або за допомогою зовнішніх сервісів, таких як Mailchimp, Pandadoc та ваші власні програми, створені за допомогою платформи Zendesk Apps.

Цей інструмент дає відмінні можливості для керування лідами, починаючи від пошуку клієнтів по кількох каналах і до створення їх точних профілів. Більше того, Zendesk Sell має вбудовану базу даних із більш ніж 20 мільйонами компаній та 200 мільйонами професіоналів, тому ви можете знаходити потенційних клієнтів на основі створених профілів покупців.

Ще однією важливою перевагою Zendesk Sell є його глибока аналітика, яка включає більше 30 готових до використання звітів та інформаційних

панелей, що настроюються. Останні дозволяють вам відстежувати активність продажів, кількість дзвінків, тривалість укладання угод, результативність та багато іншого.

Нижче ви бачите розділ “Звіти”. У ньому відображаються найважливіші показники, які можна порівнювати зі своїми фінансовими цілями як реального часу.

Pipedrive

Pipedrive - це візуальна, проста у використанні і операційна CRM-система, що легко настроюється. Як і Zendesk Sell, Pipedrive дозволяє дзвонити, надсилати листи та планувати зустрічі прямо на панелі управління. Ви також можете зберігати документи для швидкого доступу.

CRM-система надає звіти, що настроюються, і велика кількість інтеграцій, включаючи власні мобільні додатки, сервіси Google і Microsoft, а також більше 150 додаткових додатків та інструментів. Pipedrive дозволяє створитичат-ботадля свого сайту або підключити чат до реального часу.

У сервісі є віртуальний консультант, створений на основі штучного інтелекту, який рекомендує методи автоматизації процесів для підвищення ефективності роботи. Більше того, на підставі вашої поведінки, він підказує про те, як підвищити результативність продажів.

Pipedrive дозволяє налаштовувати вирви відповідно до вашого циклу продажів. На скріншоті показаний спосіб організації такого процесу на підставі взаємодії відділу продажів із потенційним клієнтом.

CRM-системи для малого бізнесу

CRM використовують не лише для великого бізнесу. Невеликі компанії та стартапи також отримують вигоду з цього інструменту. CRM-системи для малого та середнього бізнесу повинні відповідати наступним критеріям:

- доступна ціна;
- простота;
- можливість налаштування;
- підтримка інтеграцій.

Нижче описано три CRM-системи, які підходять для малого бізнесу.

Давайте розберемо, які можливості вони мають.

Fillin

Простий CRM-система для роботи зі своїм смартфоном. Є можливість підключення телефонії та інтеграції із сайтом. Доступно оповіщення клієнтів та співробітників через email, SMS taweb push повідомлення.

Fillin допомагає вести фінансовий облік, контролювати замовлення та роботу складу. За допомогою сервісу можна відслідковувати дзвінки, формувати досьє клієнтів в автоматичному режимі, налаштовувати повідомлення з лід та замовлень для клієнтів та менеджерів, підраховувати маржинальність замовлень, контролювати оборотні кошти, вести облік залишків необмеженої кількості складів та багато іншого.

retailCRM

Ця CRM-система чудово підходить для онлайн-продажів. Сервіс надає низку інструментів для омніканальної торгівлі. RetailCRM інтегрується з месенджерами, соціальними мережами, CMS, 1C, IP-телефоніями та службами доставок. Загалом сервіс надає понад 90 готових інтеграцій. Є можливість відправлення SMS та email.

CRM-система допомагає контролювати замовлення, вести облік клієнтів, відстежувати статистику та ефективність роботи менеджерів. Є мобільний додаток, що дозволяє працювати навіть на ходу. Інтерфейс retailCRM досить простий та зрозумілий. Нижче ви бачите демо-дані у розділі "Менеджери". Як бачите, у цій вкладці можна відслідковувати доходи, кількість оброблених замовлень, середній чек, сумуupsellта інші дані.

KeepinCRM

Відмінне рішення для малого та середнього бізнесу. Сервіс дозволяє вести комунікацію з клієнтами, керувати продажами, підрядниками та складами, вести фінансовий облік та документообіг. Ви можете налаштовувати мультиворонки, ставити завдання, контролювати їх виконання та відслідковувати результативність у розділі "Статистика", який показано на скріншоті нижче.

KeepinCRM надає можливість інтеграції з Новою Поштою та УкрПоштою. Сервіс дозволяє автоматично заповнювати дані про клієнтів та товари,

формувати та друкувати ТТН, відстежувати статуси доставки, формувати списки кур'єра. У CRM-системі щомісяця проводять оновлення та додають нові функції для підвищення ефективності роботи.

KeepinCRM забезпечує контроль залишків на складах, надає можливість передавати товари між філіями, налаштовувати автоматичне повернення товарів на склад, робити розсилку повідомлень клієнтам по SMS та багато іншого. CRM-система інтегрується з Instagram та Facebook Leads, Rozetka, Prom, WordPress, TurboSMS та іншими сервісами. На даний момент компанія вже працює над створенням інтеграцій із Telegram, Viber, ПриватБанк, Monobank, Liqpay, Google Sheets, Хорошоп, MailChimp та SendPulse.

Безкоштовний тариф:доступний для двох користувачів необмежену кількість часу.

Платний тариф:від $8 на місяць за одного користувача.

CRM-системи для соціальних мереж

Додавання соціальних мереж у CRM підвищує продуктивність відділу продажів26,4%. Проте найбільше від застосування CRM-системи виграють команди маркетингу та обслуговування клієнтів. Цей інструмент дозволяє планувати публікації, аналізуватиSMMдіяльність, швидко реагувати на запити користувачів, відстежувати репутацію бренду та багато іншого. Ми вибрали три CRM-системи, які допоможуть керувати соціальними мережами та збільшити кількість лідів.

Umnico

Ця CRM-система дозволяє керувати месенджерами та соцмережами. Сервіс допомагає автоматизувати роботу та відповідати на повідомлення з різних джерел в одному вікні. Ви можете підключити Instagram, ВКонтакті, Facebook Messenger, Telegram WhatsApp та чат вашого сайту.

Для ефективної роботи з лідами та клієнтами Umnico надаєвирву продажів. Відстежуйте статистику, аналізуйте дані та покращуйте своюмаркетингову стратегіюСервіс допомагає моніторити шлях ліда зміни його статусу при переході з одного рівня вирви на інший. Є звітність, сховище файлів, продуктовий каталог, моніторинг ефективності співробітників,

управління замовленнями та інші можливості.

На скріншоті нижче ви бачите розділ аналітики, в якому можна відстежувати джерела звернень, середній час відповіді, конверсії та іншу корисну інформацію.

SocialCRM

SocialCRM є розширенням для браузера. Інтегрується з Facebook, ВКонтакті, Instagram, Twitter, Одноклассники, Skype, Telegram, WhatsApp, поштою. За допомогою SocialCRM можна створювати автоворонки, налаштовувати шаблони відповідей, розсилати повідомлення на базі контактів.

Подивіться, як просто можна налаштувати вирву продажів. Просто переміщуйте задані етапи та додавайте свої.

Сервіс дозволяє вести звітність, переглядати історії взаємодії з лідами, управляти замовленнями, формувати базу клієнтів тощо. Є система завдань та нагадувань.

З плюсів CRM-системи користувачі відзначають відображення статусів клієнтів як іконок і можливість відстежувати всі бізнес-процеси в одному вікні. Сервіс допомагає прискорити та спростити процес комунікації з клієнтами, а також мінімізує ймовірність втрати лідів.

Безкоштовний тариф:відсутня, є пробна версія на 14 днів.

Платний тариф:від $10 на місяць.

AmoCRM

Проста система, що дозволяє контролювати роботу співробітників, вести облік угод, клієнтів та продажів. Сервіс надає автоматизацію вирви продажів та вбудований месенджер, є інтеграція з поштою, можливість підключення телефонії, створенняemail розсилокта інші функції.

Ось як виглядає робочий стіл AmoCRM. Тут можна переглядати статистику з вирви продажів.

Для використання цієї CRM-системи не потрібне спеціальне навчання, інтерфейс простий та зрозумілий, є мобільний додаток. AmoCRM збирає всі запити з різних каналів і додає їх у вашу вирву. Така автоматична фіксація заявок не допускає втрат потенційних клієнтів та підвищує результативність

роботи.

Також у сервісі передбачена вирва періодичних покупок. У ній відображаються постійні клієнти, які нещодавно купили та ті, що збираються купити знову. Ви можете працювати з тими, хто не звернувся повторно та вибудовувати таким чином довгострокові взаємини.

Безкоштовний тариф:відсутня, є 14 днів пробного періоду.

Платний тариф:від $5,95 за місяць.

Якщо ви не знайшли ідеального рішення для свого бізнесу у наведеному вище списку, прочитайте нашепосібник з інструментів SMMдля прокачування соцмереж. Щоб збільшити свою присутність у соціальних мережах без додаткових витрат, ознайомтеся зконструктором чат-ботів SendPulse. Він дозволяє створювати ботів для Facebook, ВКонтакті, Telegram та безкоштовно надсилати до 10 000 повідомлень на місяць.

CRM-системи для сфери нерухомості

CRM-система длямаркетингу нерухомостіповинна дозволяти ріелтору працювати в дорозі та створювати гнучкі вирви, що налаштовуються під конкретні цілі. Важливо, щоб вона забезпечувала ефективну взаємодію всередині команди та автоматизувала рутинні процеси. На ринку є багато CRM-систем, які надають перераховані можливості. Ось ті, що ми підібрали для вас.

YUcrm

Сервіс, який допомагає керівникам контролювати роботу співробітників, а менеджерам – обробляти заявки та закривати угоди. У YUcrm є інтеграція з поштою, можливість побудови вирви продажів, сховище файлів, продуктовий каталог та інші функції. Ви можете підключити телефонію, записувати розмови всередині CRM, відстежувати вартість заявок, моніторити ефективність роботи команди.

Сервіс має понад 200 інтеграцій із майданчиками на кшталт Авіто та Яндекс. Є можливість автоматичного розвантаження об'єктів нерухомості на сайт та рекламні платформи. YUcrm надає детальну статистику переглядів об'яв, кількості показів номера телефону та дзвінків.

З переваг CRM-системи користувачі відзначають функціональність,

можливість адаптації до потреб бізнесу та наскрізну аналітику. YUcrm дозволяє дізнатися все про клієнта, починаючи з моменту перегляду оголошення та дзвінка менеджеру, закінчуючи закриттям угоди.

Щоб скористатися пробним періодом, необхідно на головній сторінці натиснути кнопку “Спробувати безкоштовно” та залишити свій номер телефону для зв'язку з менеджером.

Безкоштовний тариф:відсутня, є 10 днів пробного періоду із можливістю використання всього функціоналу.

Платний тариф:від $7 на місяць за одного користувача для команди із 5-10 осіб. Чим більше користувачів, тим менша ціна.

HomeCRM

Допомагає автоматизувати рутинні процеси та підвищити ефективність роботи. Сервіс надає можливість створення добірок нерухомості прямо на карті. Завдяки цьому ви можете оцінювати інфраструктуру районів, знаходити об'єкти інших агенцій та оперативно вибирати лише підходящу клієнту нерухомість. Це значно заощаджує час.

У розділі “Об'єкти” ви можете додавати об'єкти нерухомості, а потім знаходити їх на карті за допомогою фільтра та переглядати інфраструктуру.

HomeCRM дозволяє працювати як з комп'ютера, так і телефону або планшета. Можна відстежувати ефективність роботи всього агентства та кожного співробітника окремо. Є вивантаження на дошки нерухомості, можливість підключення телефонії, шаблони проектів, звітність, історія взаємодій із клієнтами та багато іншого.

Безкоштовний тариф:немає, є безкоштовний пробний період на 14 днів.

Платний тариф:від $16 на місяць.

REALTSOFT

Сервіс інтегрується з такими месенджерами як Facebook, Telegram, WhatsApp та Viber, що дозволяє надсилати клієнтам релевантні оголошення зручним каналом комунікації. Доступна інтеграція з поштою, Google Calendar та такими рекламними майданчиками як DomRia, МирКвартир, Dom2000, FN.ua та іншими. У сервісі є можливість пошуку об'єкта на карті та доступний пошук

оголошень за номером телефону.

REALTSOFT дозволяє підключати телефонію, вести запис дзвінків та вибудовувати вирви продажів. Докладніше ознайомитися з цією CRM-системою можна у демоверсії. Нижче ви бачите функціонал сервісу та розділ "Зведення", в якому можна відслідковувати завдання, заявки, переглядати суми угод і так далі.

1.2. Висновки до розділу

Розглянуто 15 CRM-систем, що підходять для вирішення завдань бізнесу у різних сферах. Якщо ви ще не використовували CRM і не хочете інвестувати в неї гроші, оцініть переваги та зручність використання нашої системи.

РОЗДІЛ 2
ВИБІР ІНСТРУМЕНТІВ ПРОЄКТУВАННЯ І РОЗРОБКИ ПРОГРАМНОЇ СИСТЕМИ

2.1. Порівняння фреймоврків React-native та Flutter

Обмірковуючи, що використовувати для вашої наступної мобільної програми, вам доведеться дослідити, які фреймворки можуть надати вам інструменти, які вам знадобляться. React Native — найкраща собака в індустрії, а Флаттер — висхідна зірка. Якщо ви не впевнені, який з них вибрати, ця публікація допоможе вам вирішити, який з них найкраще відповідає вашим потребам.

Щоб дізнатися більше про обидві технології, було розроблено Dwipper: програму для соціальної мережі (подібну до Twitter), яку можна використовувати для публікації думок про душ (Dwipps).

Слід шукати гібридну структуру лише тоді, коли ви хочете, щоб ваша програма працювала на iOS і на Android. В іншому випадку вам краще дотримуватися вбудованої розробки для підвищення продуктивності, налагодження, тестування та простого способу випуску програми. Гібридна розробка робить процес швидшим і, отже, дешевшим, одночасно збільшуючи потенційну кількість користувачів. Нативна розробка, однак, пропонує кращу продуктивність і покращений захист даних.

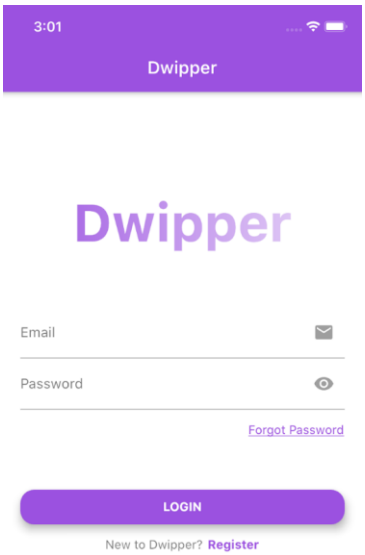


Рис. 2.1. Екран входу в Dwipper у Flutter

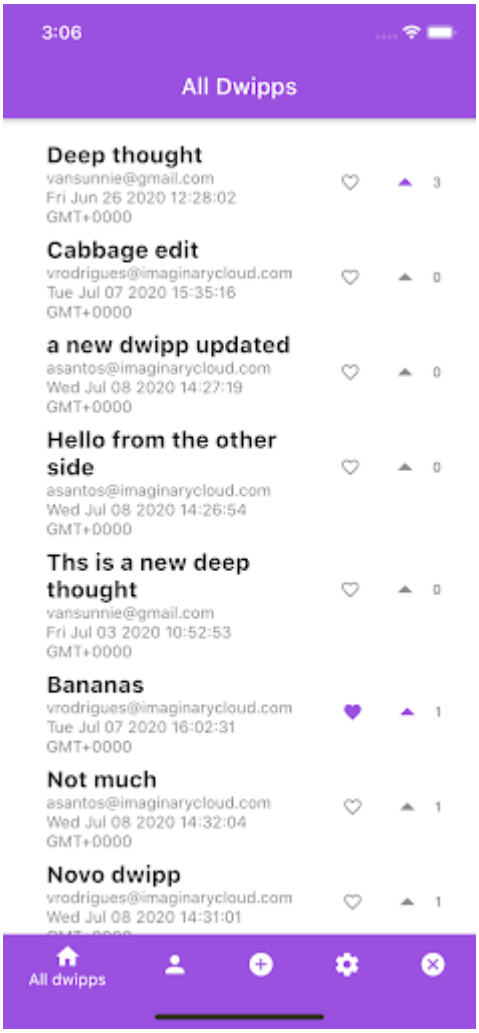


Рис. 2.2. Головний екран Dwipper у Flutter

React Native — це фреймворк мобільних додатків з відкритим кодом, створений Facebook, випущений у березні 2015 року. Він використовується як для мобільних, так і для веб-додатків, дозволяючи розробникам використовувати React та інші можливості рідної платформи. Будується на основі ReactJS іJavascript, імовірно, що розробники, які звикли до таких мов, віддадуть перевагу React Native перед іншими фреймворками.

Flutter — це набір програмного забезпечення для розробки інтерфейсу користувача з відкритим вихідним кодом, створений компанією Google, випущений у травні 2017 року. Він використовується як для мобільних, настільних та веб-додатків, усі з однієї кодової бази. На основіоб'єктно-орієнтоване програмуваннямовою Dart, ті, хто віддає перевагу цим типам парадигм, можуть віддати перевагу Flutter.

Незважаючи на те, що Flutter постачається зі своїми рекомендаціями щодо дизайну матеріалів, хтось може бути схильний його використовувати. Однак не існує проблем із невикористанням, оскільки Flutter створено з урахуванням власного дизайну бренду. Це просто має якийсь матеріальний дизайн як бонус.

Який із них популярніший: React Native чи Flutter?

Щоб дати вам певне уявлення про те, наскільки популярні обидві фреймворки, ось графік, що відображає кількість запитань на StackOverflow, які React Native поставив у порівнянні з Flutter.

Як бачимо, Flutter швидко набирав популярності, особливо з 2018 року, тому з кінця 2019 року StackOverflow ставив більше запитань про Flutter, ніж React Native. Flutter став трохи популярнішим, оскільки Google вкладає в нього багато часу.

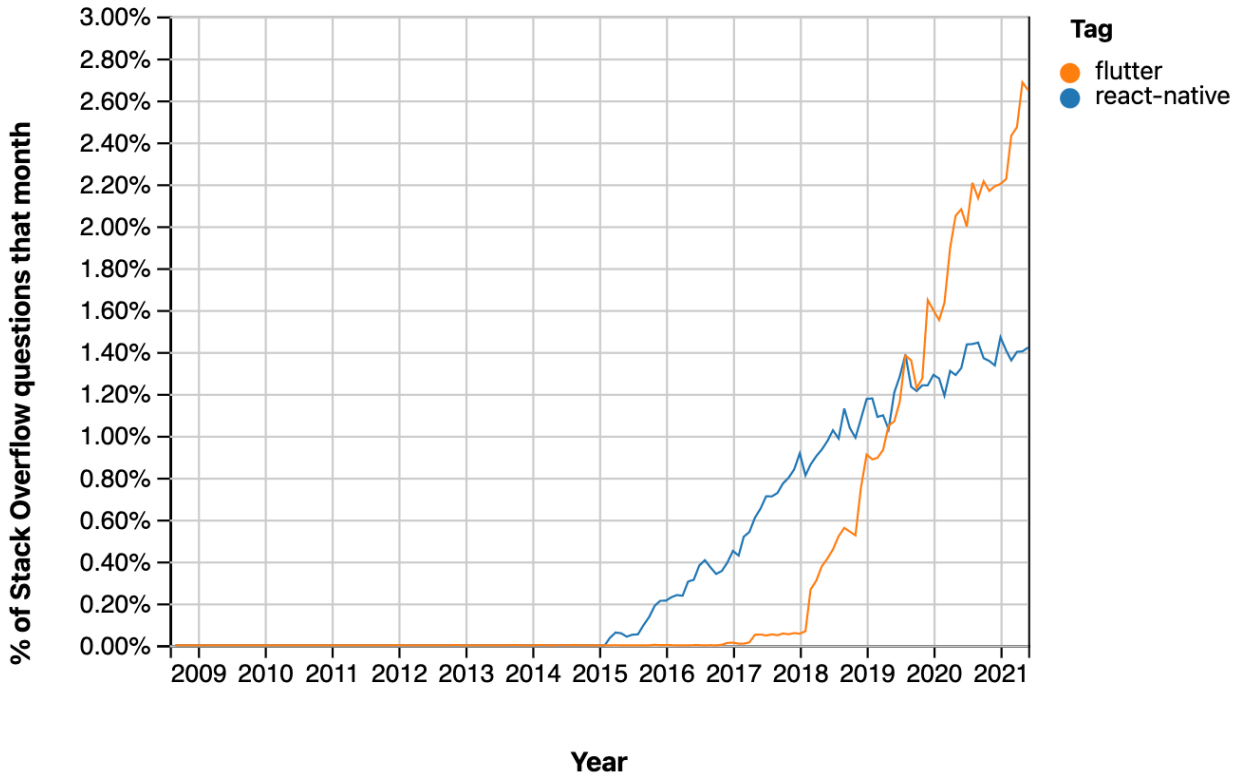


Рис. 2.3. Графік зі StackOverflow, що порівнює кількість запитань

Оскільки він походить від JavaScript, його легше вивчати та налагоджувати, ніж інші мови. Це означає, що ви маєте більш прямий підхід до розвитку. Однак ця легкість забезпечується ціною багатьох помилок, які люди можуть знайти у своєму коді під час виконання, оскільки JavaScript не є суворою мовою. Щоб уникнути цього, пропонує Facebook, творець React NativeПотік, перевірка типів. Це дозволяє включати анотації до JavaScript для визначення типів і виявлення небажаної поведінки, що використовується для пом'якшення цієї проблеми шляхом виявлення проблем під час написання коду.

Flutter не так легко вивчити, як React Native. Тим не менш, знання Dart або будь-якої іншої об'єктно-орієнтованої мови полегшить вам життя. Завдяки зростанню популярності пошук допомоги в Інтернеті робиться швидко і не викликає проблем.

React Native пропонує компоненти, еквівалентні веб-аналогу React. Span в React — це текст у React Native, що було б еквівалентно UIView в iOS і TextView в Android.

Для кожної програми React Native є притаманними два потоки:

Основний потік запускає стандартну рідну програму, обробляючи відображення елементів і обробляючи жести користувача.

Інший виконує весь код JavaScript в окремому механізмі, JavaScriptCore або V8, який має справу з бізнес-логікою програми. Він також визначає функціональність і структуру інтерфейсу користувача.

Ці потоки ніколи не спілкуються безпосередньо і не блокують один одного.

Приблизний еквівалент UIView iOS та Android View у Flutter — це віджет, який дещо відрізняється від аналогів, згаданих вище. Для початку вони мають різну тривалість життя: вони незмінні і існують лише до тих пір, поки їх не потрібно змінити. Коли це станеться, фреймворк Flutter створить новий набір екземплярів віджетів. Для порівняння, подання в iOS і Android малюються один раз і лише перемальовуються, коли викликаються setNeedsDisplay() і invalidate, відповідно.

Наявність такої реактивної фреймворку дозволяє розробнику відкинути необхідність отримувати посилання на віджети, полегшуючи структуру для всього бекенда за допомогою однієї мови.

Компоненти інтерфейсу користувачамають вирішальне значення при розробці кросплатформних мобільних додатків. Тому обидві рамки повинні забезпечувати плавність і легкістьAPIдоступ до рідних модулів. У цьому відношенні React Native має менше компонентів, ніж Flutter.

З одного боку, React Native дуже залежить від сторонніх бібліотек для отримання рідних модулів. З іншого боку, у Flutter інтегровані віджети інтерфейсу користувача; таким чином, розробнику не доведеться шукати сторонні бібліотеки.

React Native використовує JavaScript, а Flutter — мову програмування Dart. JavaScript — це динамічно типізована мова, яка була надзвичайно популярна протягом багатьох років і часто використовується з React, а також з іншими фреймворками JavaScript. Оскільки багато розробників зазвичай знайомі з JavaScript, прийняти React Native досить просто.

Навпаки, з Flutter розробники обов’язково повинні вивчати Dart, який не

дуже вживана мова. Однак це також не головоломка, враховуючи, що його синтаксис має багато подібностейJavai JavaScript.

Flutter має гаряче перезавантаження, що означає, що розробник може безпосередньо вводити новий код у запущену програму, що може заощадити величезну кількість часу та прискорити процес розробки. Крім того, щоб уникнути втрати під час перезавантаження, гаряче перезавантаження також підтримує стан програми.

React Native дуже усвідомлював успіх цієї функції серед розробників, і її версія 0.61 була запущена з функцією «швидке оновлення», яка відповідає «гарячому перезавантаженню». «Швидке оновлення» також дозволяє розробнику переглядати зміни, внесені в програму, без необхідності її перекомпіляції.

Ще одна велика перевага використання React Native полягає в тому, що він має велику спільноту розробників, які долучилися до багатьох бібліотек. Ці бібліотеки можна використовувати для створення блоків і, отже, прискорення роботипроцес розвитку.

Flutter має дуже вичерпну та розширену документацію, яка надає розробникам детальні посібники та підручники. Крім того, Flutter додатково включає інспектор і налагоджувач Flutter, щоб допомогти розробникам під час розробки програми.

Документація React Native також дуже багата і містить багато навчальних посібників, бібліотек, фреймворків інтерфейсу користувача, статей та інших матеріалів. Крім того, він вигідно мати велику спільноту та бути частиною сім'ї React.

Навпаки, у Flutter (поки що) немає такої масової спільноти розробників, враховуючи, що це ще досить молодий фреймворк. Тим не менш, він досяг дуже поважного розміру, розраховуючи на велику підтримку з боку інших розробників, і він стає все більш популярним.

З одного боку, React Native вимагає стороннього рішення для автоматизації доставки та розгортання, що означає, що він не надає розробникам рішення CI/CD для Google Play Delivery або App Store.

З іншого боку, Flutter дозволяє розробнику використовувати інтерфейс

командного рядка (CLI) для створення та розгортання програм для Android та iOS. Тим не менш, це може також потребувати сторонніх рішень, якщо потрібна розширена автоматизація.

Про проект: Dwipper

У рамках свого досвіду навчання я працював з обома фреймворками, щоб розробити додаток під назвою Dwipper, де користувачі публікують свої думки про душ як Dwipps.

Програма мала такі екрани:

Базовий логін, реєстрація та забутий пароль

Усі Dwipps, щоб побачити Dwipps кожного

Мої Dwipps, де ви можете побачити всі Dwipps, які ви опублікували

Новий Dwipp, щоб написати та опублікувати Dwipp

Змінити пароль, де користувач може змінити свій пароль

Вийти, де користувач може вийти

Для мене це був чудовий досвід навчання. Flutter був більш інтуїтивним, наприклад, як ви працюєте з віджетами, було простіше, ніж компоненти в React Native. Єдиною справжньою проблемою, яку ми спочатку мали, було спілкування з бекендом програми. Це не було щось безпосередньо пов’язане з самим фреймворком, лише питання з’ясування того, що і як надсилати.

React Native, з іншого боку, був трохи більш клопітким, можливо, тому, що я не мав попереднього досвіду роботи з JavaScript. Найбільш неприємною частиною була робота над тим, як налаштувати навігацію між частиною входу та «справжньою» частиною програми. Встановлювати новий компонент щоразу, коли нам щось потрібно для конкретного завдання, було не дуже приємно.

Порівняння з іншими фреймворками

Flutter проти Ionic

Оскільки Ionic спочатку був випущений ще в 2013 році, він трохи відстає від Flutter. Якщо розробник не працює в JavaScript,CSSi HTML5, час підвищення рівня володіння буде коштувати часу та грошей на вивчення фреймворку, тоді як Flutter має більш плавний досвід навчання. Можливість гарячої перезавантаження для швидкого експериментування також дозволяє

пришвидшити розвиток.

2.2. React Native проти Ionic

Коли ми говоримо про React native у порівнянні з Ionic, можна сказати, що Ionic має чудову документацію, а її інструменти добре розуміють веб-розробники. Але оскільки він побудований на основі браузера, код не може легко отримати доступ до рідних функцій.

React Native, з іншого боку, має велику спільноту, яка підтримує його, що призводить до того, що інші розробники вирішують більше проблем або проблем в Інтернеті. React Native перекладається на нативний код, що дозволяє легко досягати 60 кадрів в секунду, ніби це нативний додаток. Однак його підтримує велика компанія, а це означає, що будь-які нові інструменти будуть випущені лише відповідно до потреб цієї компанії.

React Native проти Flutter: що краще?

Зрештою, коли запитуєте себе, який із них найкращий, Flutter чи React Native, все зводиться до особистих уподобань.

Хтось із більшими знаннями та досвідом роботи з JavaScript чи React, швидше за все, віддасть перевагу використанню React Native, оскільки перехід є досить плавним. З іншого боку, розробник, якому подобається об’єктно-орієнтована парадигма, швидше за все, віддасть перевагу Flutter.

У разі потребирозробити мобільний додатокскладніше, ніж Dwipper, ви можете розглянути можливість використання Flutter, оскільки зараз він трохи популярніший, ніж React Native. Ця популярність може стати в нагоді під час пошуку помилок у вашому додатку. Більша кількість людей означає більш широкий спектр проблем, що підвищує ймовірність знайти когось із такою ж проблемою, як і ви, або отримати її вирішення іншим членом спільноти, який уже стикався з нею.

2.3. Вибір СУБД

На ринку доступно багато баз даних, і знати, яку з них вибрати, може бути надзвичайно важко. Відмінний спосіб розпочати виключення деяких параметрів — спочатку чітко зрозуміти основні відмінності між базами даних SQL і NoSQL.

У цій статті ми представляємо детальне порівняння між цими двома різними типами баз даних щодо структури, схеми, масштабованості, запитів і транзакцій.

SQL є мовою програмування; однак, це не така мова програмування загального призначенняJava,Javascript, абоPython. Замість цього SQL виконує конкретну мету: доступ до даних і керування ними.

Якщо бути більш точним, SQL означає Structured Query Language. Це мова запитів, яка дозволяє отримувати конкретні дані з баз даних, і в цьому сенсі вона призначена для доступу, зберігання та керування реляційними базами даних.

Реляційна база даних — це тип бази даних (зазвичай організований у таблиці), який дає змогу розпізнавати та отримати доступ до даних стосовно іншої частини даних у тій самій базі даних. Іншими словами, він зберігає пов’язані дані в кількох таблицях, які організовані в стовпці та рядки, і дозволяє користувачеві одночасно запитувати дані (або інформацію) з різних таблиць.

Реляційна база даних - це база даних, яка слідує зареляційна модель даних. Для підтримки реляційної бази даних використовується система управління реляційною базою даних (RDBMS). Отже, для роботи в цій системі багато баз даних, як правило, використовують SQL для керування базою даних і запитів до неї. Таким чином, SQL є мовою, яка дозволяє спілкуватися з даними в СУБД.

Важливим аспектом, який слід прояснити, є те, що SQL сама по собі не є системою баз даних. По правді кажучи, порівнюючи SQL з NoSQL, основні відмінності, які оцінюються, — це реляційні бази даних і нереляційні бази даних (а також розподілені бази даних).

Інший аспект, який слід враховувати, полягає в тому, що SQL — не єдина мова програмування, яка може запитувати реляційні бази даних, але, безперечно, є найпопулярнішою. Тому терміни «бази даних SQL» і «реляційні бази даних»

часто використовуються як взаємозамінні.MySQL,PostgreSQL,Microsoft SQL ServeriБаза даних Oracleє одними з найвідоміших СУБД, що використовують SQL.

NoSQL відноситься до нереляційних баз даних і до розподілених баз даних. NoSQL також може означати «не тільки SQL», щоб підкреслити, що деякі системи NoSQL також можуть підтримувати мову запитів SQL. Насправді, перш ніж йти далі, важливо мати на увазі, що NoSQL не обов'язково означає, що база даних не підтримує SQL. Натомість це означає, що база даних не є СУБД.

У той час як традиційні СУБД покладаються на синтаксис SQL для зберігання та запиту даних, з іншого боку, системи баз даних NoSQL використовують інші технології та мови програмування для зберігання структурованих, неструктурованих або напівструктурованих даних.

SQL проти NoSQL: історичний контекст

Реляційна модель даних була введена в 1970 році Е. Ф. Коддом. Чотири роки потому (1974) Реймонд Бойс і Дональд Чемберлін представили SQL, який спочатку був розроблений для запитів.IBM System R, система управління базами даних.

Не пройшло багато часу, поки SQL став величезним успіхом серед систем реляційних баз даних завдяки своїй неймовірній практичності та здатності зменшити дублювання даних. Таким чином, незабаром і тривалий час вона вважалася переважною мовою систем реляційних баз даних. Але згодом сталася одна крихітна річ: Всесвітня мережа в 1989 році.

Наслідки? Більше даних.Набагато більше даних. Як ми знаємо, розвиток Інтернету був не повільним, і оскільки нові джерела та обсяги даних продовжували руйнувати наш світ, реляційні бази даних почали боротися.

На початку 21-го століття, щоб обробляти цю величезну кількість даних, нереляційні системи, такі якВеликий стіл(від Google, 2006 р.) іДинамо(від Amazon, у 2007 році), почали робити свій власний шлях. Зосереджено на масштабованості та швидкому застосуванні.

Протягом цих років, коли почало з'являтися все більше нереляційних баз даних, концепція NoSQL стала дуже популярною (незважаючи на те, що цей

термін вперше ввів у 1998 році Карло Строцці, а нереляційні бази даних існують з 60-х років).

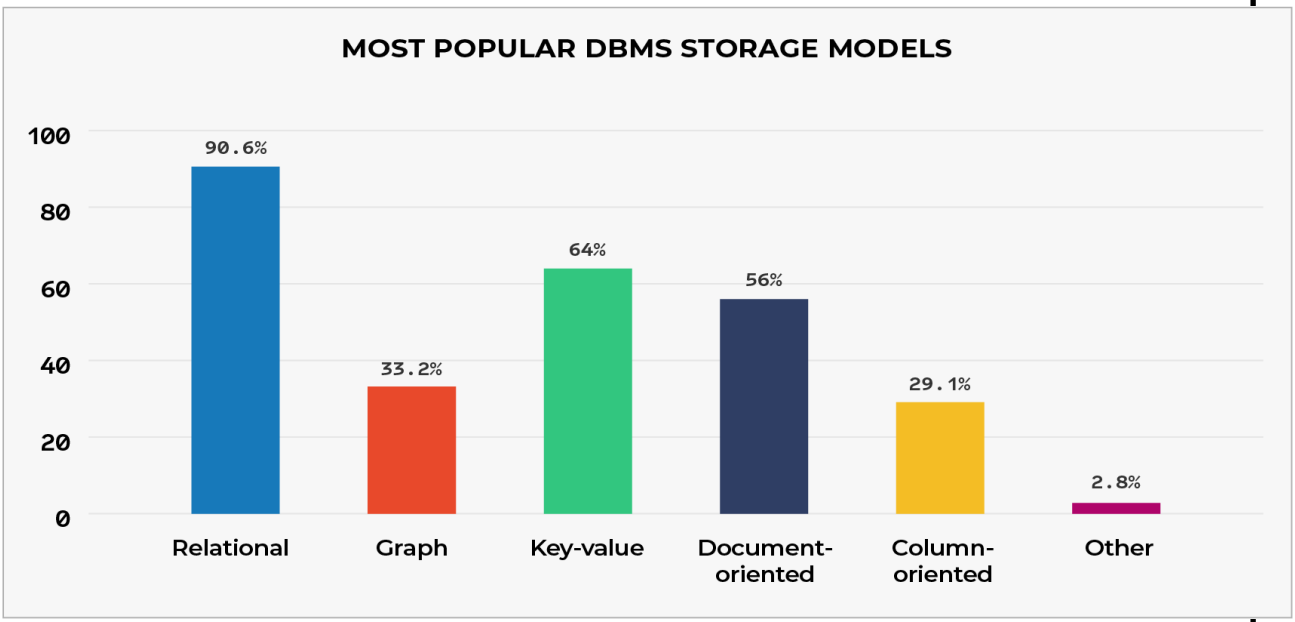
Чи означав початок століття кінець SQL і, отже, реляційних баз даних? - Ні, звичайно, ні. З двох причин:

Реляційні бази даних були (і залишаються) досі неймовірно корисними і мали багато переваг. Крім того, SQL — це дуже добре розроблена і популярна мова запитів, яка продовжує домінувати в системах баз даних.

NoSQL мав свої недоліки. Оскільки кожна база даних NoSQL мала різну мову запитів, потрібно було вивчити набагато більше мов. Крім того, деякі додаткові проблеми включали додаткові труднощі з підключенням баз даних до програм, а сторонні екосистеми (для забезпечення візуалізації та операційних інструментів) були відсутні.

Час навчив нас, що бази даних SQL не кращі і не гірші за бази даних NoSQL. Вони є просто кращими і більш підходящими для різних застосувань, що стосуються систем управління базами даних (СУБД).

Відповідно зЗвіт про тенденцію збереження даних DZone за 2021 рік(зображення нижче), реляційні бази даних є найпопулярнішою СУБД. Однак бази даних NoSQL відносяться до всіх нереляційних СУБД (включаючи графіки, документоорієнтовані, ключ-значення, стовпці та інші). Тому комбіновані бази даних NoSQL на даний момент є більш популярними, ніж реляційні бази даних.



Найпопулярніші моделі сховища СУБД Джерело: Звіт про тенденцію збереження

даних DZone

Загалом, правильний вибір, коли справа доходить до SQL і NoSQL, залежить насамперед від знання типу бази даних, яка краще відповідає цілям кожного бізнесу чи організації. Перш ніж ми перейдемо до того, коли використовувати кожен, давайте спочатку розглянемо їх відмінності.

SQL проти NoSQL: порівняння

Структура

Бази даних SQL організують і зберігають дані за таблицями з фіксованими стовпцями і рядками. Навпаки, бази даних NoSQL можна зберігати різними способами:

документ (JSON);

Широкі колонки (таблиці, організовані за допомогою рядків і динамічних стовпців);

пари ключ-значення;

Графічні бази даних (організовані за допомогою вузлів і ребер).

Схема

Бази даних SQL вимагають фіксованої попередньо визначеної схеми, і всі дані повинні мати подібну структуру. Отже, заздалегідь потрібна велика підготовка щодо системи. Крім того, гнучкість скомпрометована, враховуючи, що потенційні модифікації в структурі можуть бути складними, дуже складними і можуть порушити роботу системи.

У свою чергу, бази даних NoSQL дотримуються динамічної схеми для неструктурованих даних. Оскільки він не вимагає попередньо визначеної структури, модифікації легше виконувати. Тому бази даних NoSQL мають більшу гнучкість; однак, незважаючи на переваги, гнучкість також може поставити під загрозу надійність.

Масштабованість

Що стосується масштабованості, бази даних SQL дотримуються вертикального підходу, також відомого як «масштабування». У базах даних це означає, що можна збільшити обсяг даних на одному сервері, додавши більше потужності існуючій машині, використовуючи, наприклад, CPU, RAM або SSD.

З іншого боку, бази даних NoSQL масштабуються горизонтально (також відомі як «масштабування»), оскільки вони масштабуються між стандартними серверами, що означає, що до пулу ресурсів додається більше серверів, і дані можуть бути розподілені між цими ресурсами.

Операції JOIN дозволяють з'єднувати і пов'язувати фрагменти даних. Як правило, бази даних NoSQL (можуть, але) не розроблені для ефективної підтримки JOIN. Об’єкти можуть бути на різних серверах у нереляційних системах баз даних, не турбуючись про з’єднання таблиць із кількох серверів.

Таким чином, NoSQL дозволяє легко масштабувати, розподіляючи дані та маючи рівень маршрутизації, який може перенаправляти запит на відповідний шард, роблячи бази даних NoSQL дуже масштабованими та швидкими для запитів. Однак це ставить під загрозу цілісність даних і не дотримується підходу ACID.

«Масштабування» в СУБД, як правило, важче реалізувати через концепцію ACID, якої дотримуються реляційні бази даних. Щоб багатосерверна СУБД підтримувала цілісність даних між транзакціями, потрібен швидкий бекенд канал зв’язку. Цей канал мав би синхронізувати всі записи та транзакції, а також запобігти можливим тупикам.

Незважаючи на те, що технічно можливо масштабувати в RDBMS, ці системи баз даних зазвичай масштабуються, щоб забезпечити цілісність даних і принципи ACID замість розподілу даних між кількома серверами.

Запит

Як згадувалося раніше, SQL існує протягом тривалого часу; таким чином, вона користується пошаною як зріла і популярна мова, яка користується надійною репутацією. Це неймовірно ефективно, коли справа доходить до запитів даних, маніпулювання та отримання даних з реляційних баз даних. Крім того, він також виділяється декларативним і легким.

Ще однією великою перевагою SQL є те, що його відносно легко навчитися, а це означає, що маркетологи та бізнес-аналітики можуть використовувати його, не вимагаючи допомоги технічного персоналу.

Коли справа доходить до виконання запитів NoSQL, це може бути не так

просто, як бази даних SQL, оскільки зазвичай потрібно виконувати додаткову обробку даних і не має декларативної мови запитів. Тому ці завдання зазвичай виконуєнауковці данихабо розробники.

Загалом, те, як виконувати запити в базах даних NoSQL, багато в чому залежить від бази даних, про яку йдеться. Наприклад, вMongoDB, щоб запитувати дані з бази документів JSON, необхідно вказати документи з властивостями, яким мають відповідати результати, і застосувати таку функцію: db.collection.find(). Інші популярні рішення можуть включати створення функціональних можливостей запиту безпосередньо на рівні програми (а не на рівні бази даних) або впровадженняMapReduce.

Транзакції бази даних: ACID проти BASE

Бази даних SQL зазвичай відповідають властивостям ACID щодо транзакцій. ACID означає атомарний, послідовний, ізольований і довговічний. Давайте розглянемо детальніше, щоб точніше зрозуміти, що це означає:

атомний:гарантує, що всі дані в базі даних обов'язково перевірені. Якщо кожна транзакція даних не виконується належним чином, процес повертається до початкового стану.

послідовний:гарантує, що оброблена транзакція даних не пошкоджує структурну цілісність бази даних.

ізольовані:кожна транзакція ізольована від інших транзакцій даних. Отже, транзакція не може порушити цілісність іншої транзакції.

Довговічний:дані, пов’язані з обробленою транзакцією, не впливатимуть на дані, якими маніпулюють, навіть якщо транзакція завершиться невдачею.

Як можна помітити, модель ACID забезпечує надійність і послідовність транзакції. Тому бази даних, які дотримуються цієї моделі, найкраще підходять для організацій і підприємств, які не можуть ризикувати недійсними та перерваними транзакціями даних або будь-якими іншими помилками (наприклад, фінансові установи).

Системи керування реляційними базами даних (такі як MySQL, SQLite, PostgreSQL тощо) сумісні з ACID. Однак, незважаючи на те, що підхід до баз даних NoSQL зазвичай суперечить принципам ACID, деякі бази даних NoSQL

(наприклад, MongoDB, IBM Db2 і CouchDB від Apache) також можуть інтегрувати та дотримуватися правил ACID.

У нереляційних базах даних надійність і узгодженість даних відповідно до ACID зазвичай не є пріоритетом номер один, враховуючи, що це може поставити під загрозу швидкість і високу доступність.

Для баз даних NoSQL пріоритет має тенденцію зосередитися на гнучкості та високій швидкості транзакцій. З цієї причини моделі BASE дотримуються в багатьох системах баз даних NoSQL. Це означає «Основно доступний», «М'який стан» і «Зрештою узгоджений».

В основному доступні:забезпечити доступність даних шляхом розширення та реплікації даних на вузлах кластера бази даних.

М'який стан:розробники відповідають за забезпечення узгодженості бази даних.

Зрештою послідовний:узгодженість не є миттєвою, але її можна досягти, а тим часом можна зчитувати дані.

Бази даних NoSQL зазвичай дотримуються моделі BASE, що забезпечує більшу еластичність, ніж модель ACID. Як уже згадувалося, ACID краще для підприємств і організацій, яким необхідно забезпечити узгодженість, передбачуваність та надійність кожної транзакції.

Навпаки, модель BASE більше підходить для компаній, які віддають перевагу високій доступності, масштабованості та гнучкості транзакцій даних. Наприклад, програма соціальної мережі обробляє величезні обсяги даних, які часто не дуже добре структуровані; таким чином, у цьому випадку модель BASE може полегшити (і пришвидшити) зберігання даних.

SQL проти NoSQL: порівняльна таблиця

СУБД	**Бази даних SQL**	**Бази даних NoSQL**
Тип	Реляційна база даних.	Нереляційна база даних.

Структура	Бази даних SQL організують і зберігають дані за таблицями з фіксованими стовпцями і рядками	Бази даних NoSQL можуть бути: графічними, орієнтованими на документ, ключ-значення, стовпцями та іншими.
Схема	Виправлена схема.	Динамічна схема.
Масштабованість	Бази даних SQL мають вертикальний підхід.	Бази даних NoSQL масштабуються горизонтально.
Запит	SQL, як правило, є переважною мовою запитів.	Спосіб виконання запитів у базах даних NoSQL багато в чому залежить від розглянутої бази даних; немає декларативної мови запитів.
Транзакції бази даних	Бази даних SQL зазвичай відповідають властивостям ACID щодо транзакцій.	Модель BASE дотримується в багатьох системах баз даних NoSQL.
Пріоритет	Цілісність, узгодженість і стабільність даних.	Гнучкість, швидкі запити та масштабованість.

SQL проти NoSQL: коли використовувати?

Тепер, коли ми розглянули основні відмінності між SQL і NoSQL, настав час пояснити, коли використовувати той чи інший. Перш ніж прийняти остаточне

рішення, важливо врахувати наступні аспекти:

Тип даних, про які йде мова;

Обсяг даних;

Як буде керуватися базою даних?

Коли використовувати SQL?

Що стосується першого аспекту, бази даних SQL є більш підходящим варіантом, ніж NoSQL, коли цілісність і узгодженість даних є ключовими в організації.

Часто існує помилкова думка, що реляційні бази даних не є хорошим варіантом для обробки великих обсягів даних. Це не зовсім так. Багато баз даних SQL, такі як PostgreSQL і MySQL, справді можуть обробляти дуже поважні обсяги даних.

Однак, оскільки СУБД, які використовують SQL, мають фіксовану схему і вимагають структурування даних, імовірно, буде дуже складно підтримувати необхідне обслуговування, гнучкість і продуктивність, яких, наприклад, може вимагати бізнес, який обробляє великі дані.

На перший погляд може здатися, що наявність фіксованої схеми обмежує. Ну, знову ж таки, це залежить від мети. Наявність попередньо визначеної бази даних схем також робить бази даних SQL найбільш підходящим варіантом для роботи з системами управління заробітною платою або навіть для обробки бронювання авіаквитків. Насправді більшість банківських установ покладаються на систему баз даних SQL.

Як ми вже пояснювали раніше, реляційні бази даних зазвичай сумісні з ACID, що означає, що транзакції даних забезпечують цілісність, валідність і надійність. Крім того, SQL може обмежувати деякі функції, але це також дуже зріла технологія.

Крім того, реляційна база даних і SQL пропонують велику підтримку щодо випадкових запитів. Таким типом баз даних зазвичай легше керувати. Оскільки SQL є популярною мовою запитів і її відносно легко вивчити, для її підтримки не обов’язково потрібна велика команда інженерів.

Коли використовувати NoSQL

Бази даних NoSQL здатні зберігати різні типи даних і не повинні бути такими структурованими, як бази даних SQL. Таким чином, нереляційні бази даних забезпечують велику адаптивність і гнучкість, що робить їх більш підходящим вибором під час роботи з великими наборами неструктурованих і непов’язаних даних.

Як правило, чим більший набір даних, тим імовірніше, що база даних NoSQL є кращим варіантом. Нереляційні бази даних, як правило, відрізняються вимогами до масштабованості та доступності, будучи ідеальними для соціальних мереж і додатків реального часу (наприклад, онлайн-ігор, обміну миттєвими повідомленнями), наприклад.

Бази даних NoSQL вимагають знання програмування. На відміну від SQL, який також можуть вивчати співробітники з інших галузей, таких як менеджмент і маркетинг, базам даних NoSQL зазвичай потрібен хтось із досвідом програмування та вмінням отримувати інші мови відповідно до використовуваних систем баз даних.

2.4. Висновки до розділу

Вибір належної бази даних не є прямим і точним рішенням навіть для експертів. Вирішити, чи вибрати реляційні чи нереляційні бази даних, є чудовим способом для початку. Тим не менш, важливо також розглянути безліч варіантів SQL і NoSQL, доступних на ринку.

Наприклад, для великої кількості неструктурованих даних CouchDB або MongoDB можуть бути хорошим варіантом, але, можливо, для високої доступності,RedisiКассандраможе бути більш підходящим. І це все нереляційні системи баз даних!

З іншого боку, бази даних SQL пропонують багато переваг щодо транзакцій даних і загальної цілісності даних. Більше того, зв’язки між реляційними базами даних можна легко ідентифікувати та визначити, що спрощує визначення важливих ідей.

РОЗДІЛ 3
РОЗРОБКА ПРОГРАМИ ТА ОПИСАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1. Описання середовища програмування

Короткий посібник із асинхронних шаблонів JavaScript

https://www.imaginarycloud.com/blog/async-javascript-patterns-guide/

Від веб-додатків до серверів і мобільних додатків, від невеликих програм до великих проектів, JavaScript є всюди. Це основний вибір, щоб прийняти будь-який проект, тому що зараз 2020 рік, а JS — ще більш зріла мова, яку підтримує величезна спільнота.

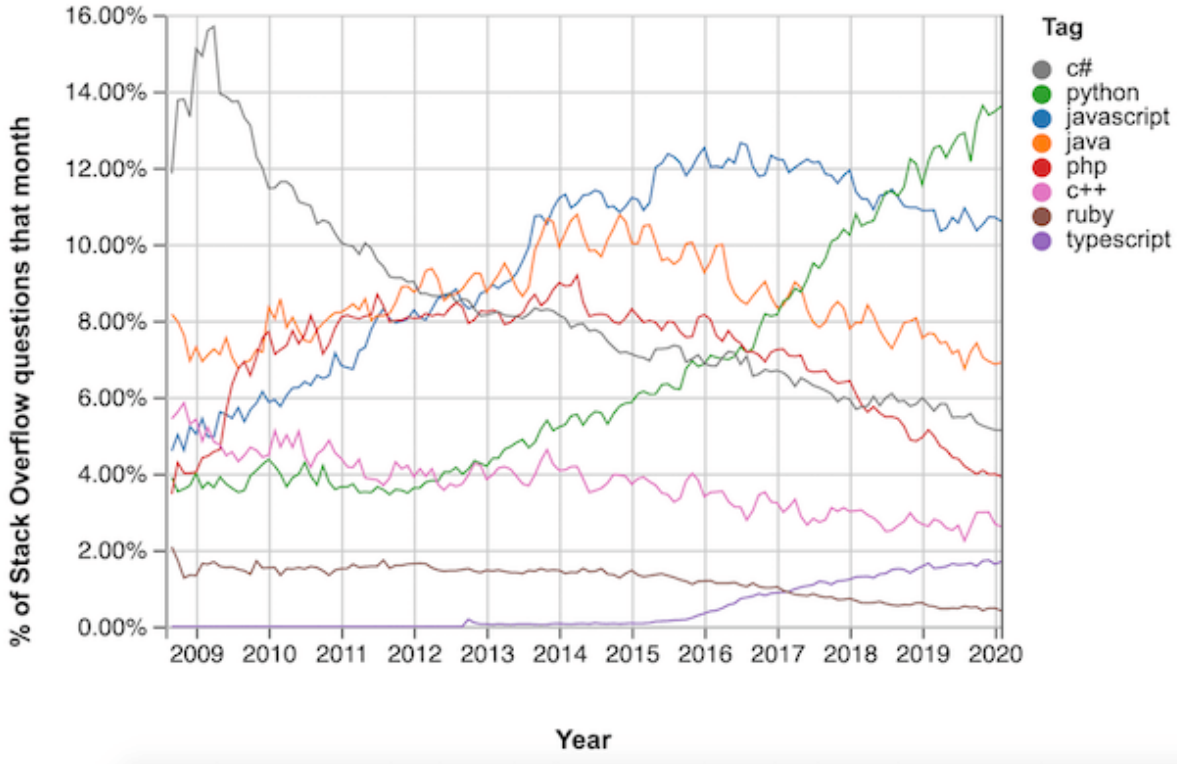


Рис. 3.1. Тенденції переповнення стека (insights.stackoverflow.com)

3.2. Написання асинхронного JavaScript

У JavaScript весь код виконується синхронно в циклі подій, який послідовно виконує невеликі фрагменти нашої програми. У циклі подій кожна ітерація називається «галочкою» і виконується, доки черга не буде порожня. Кожен фрагмент має «галочку» для обробки, і після його завершення починається наступний. Для невеликих програм цього достатньо, але оскільки ми починаємо виконувати більш важкі операції, які потребують більше часу, наприклад, доступ до бази даних або отримання даних через Інтернет, нам потрібні кращі механізми для їх обробки.

У JavaScript функції є першокласними об’єктами і азворотний дзвінокце просто функція, яка передається як аргумент іншій функції. Також відомі як функції високого порядку, зворотний виклик слід викликати щоразу, коли асинхронна робота закінчується.

```
fs.readFile('./imaginary.txt', (помилка,результат) => {
 якщо (помилка) {
консоль.помилка("Помилка:",помилка)
 повернутися
 }
консоль.журнал('Результат: ',результат)
})
```

Оскільки зворотні виклики — це лише функції, вони підтримуються всіма середовищами, які запускають JavaScript, від наших браузерів до серверів, які працюютьNode.js. Простий, але потужний, цей шаблон є основним для асинхронності. Однак у нього є і свої недоліки.

Коли проекти починають рости, і нам потрібно почати робити більш складний код, стає важче впроваджувати загальні рішення в наші програми, що ускладнює їх читання та обслуговування. Коли це станеться, ми починаємо мати форму піраміди})подібно до того, що ми бачимо в наступному прикладі.

```
fs.readFile('./imaginary.txt', (помилка,уявнийРезультат) => {
```

```
 якщо (помилка) {
консоль.помилка(`Помилка:${помилка}`)
 повернутися
 }
fs.readFile('./cloud.txt', (помилка,cloudResult) => {
 якщо (помилка) {
консоль.помилка(`Помилка:${помилка}`)
 повернутися
 }
 констдані=уявнийРезультат+cloudResult
fs.writeFile('./imaginarycloud.txt',дані,помилка=> {
 якщо (помилка) {
консоль.помилка(`Помилка:${помилка}`)
 повернутися
 }
консоль.журнал("Успіху!")
 })
 })
})
```

Це зазвичай відомо як"Пекло зворотного дзвінка".

Однак найгірше, що ми можемо мати із зворотними викликами, — це інверсія керування. Якщо це станеться, ми передаємо контроль над послідовністю програмного потоку іншим сторонам, що ускладнює (або навіть неможливо!) його належну перевірку.

Архітектура, керована подіями, також може використовуватися для написання асинхронного коду JavaScript. Ця архітектура складається з одного випромінювача подій з відповідним прослуховувачем подій, який надсилає події після завершення асинхронного коду. Надсилання різних типів подій дозволяє мати різні зворотні виклики для кожного типу слухачів. Одним із основних прикладів і дійсно важливою частиною розробки інтерфейсу є запит даних через Інтернет. Для досягнення цього ми могли б

використовуватиXMLHttpRequestоб'єкт, який активно використовується в програмуванні AJAX.

Об’єкт XMLHttpRequest вже має деякі обробники подій, визначені для обробки потоку запитів, тому нам просто потрібно скористатися ними. Але цей шаблон містить багато шаблонного коду, оскільки ми можемо додавати та видаляти слухачів залежно від різних типів подій, які нам потрібні. Це ідеально працює на невеликій веб-сторінці, але щойно складність і функціональні можливості зростають, її обслуговування стає роздутим і громіздким, тому потрібні кращі абстракції!

Що таке Обіцянка

Обіцянкиїх важче освоїти, але вирішує проблему інверсії контролю. Вони трохи повільніше, ніж зворотні виклики, але натомість ми отримуємо велику надійність.

Ми завжди можемо бути впевнені, що Promise розв’яжеться або відхилиться, оскільки вони є «обгорткою» навколо значення, яке може ще не існувати. Обіцянки є надійним механізмом, який також допомагає більш послідовно виражати асинхронний код. Вони можуть мати щонайбільше одне значення дозволу, що означає, що обіцянку завжди потрібно вирішити або відхилити.

Так вони вирішують інверсію контролю. Не видаляючи зворотні виклики, а створюючи механізм на обгортці, який вирішує цю проблему.

Ми можемо ланцюгувати кілька Promises у нашому коді, не встановлюючи новий рівень відступів після кожного з них, використовуючи.потім().

```
констобіцянка= новий Обіцяйте((вирішити,відхилити) => {
fs.readFile('./imaginarycloud.txt', (помилка,результат) => {
 якщо (помилка) {
 відхилити(помилка)
 } інше {
 вирішити(результат)
 }
 })
```

```
})

обіцянка.потім(текст=> {
консоль.журнал(текст)
}).виловити(помилка=> {
консоль.помилка(`Помилка:${помилка}`)
})
```

Promises надають більше функціональних можливостей, як, наприклад, thePromise.all()Promise.race()порівняно з останніми доповненнями APIPromise.allSettled()iPromise.any(). З більш складними інтерфейсними веб-додатками нам потрібно більше та кращих механізмів.

```
констresolveSync= новий Обіцяйте(вирішити=> вирішити('Привіт'))
констrejectAsync= новий Обіцяйте((_,відхилити) =>
 setTimeout(() => відхилити(новий Помилка()), 2500))
констresolveAsync= новий Обіцяйте(вирішити=>
 setTimeout(() => вирішити('уявний'), 4000))

// Promise.all проти Promise.allSettled

// Повертає масив з усіма вирішеними значеннями або відхиляє, якщо якісь не вдаються
Обіцяйте.всі([resolveSync,rejectAsync,resolveAsync])
// Це відхилить обіцянку через 2500 мс

// Повертає один масив об'єктів зі статусом і значенням/причиною
Обіцяйте.все врегульовано([resolveSync,rejectAsync,resolveAsync])
/**
* [
* { статус: 'виконано', значення: 'привіт' },
```

```
* { статус: 'відхилено', причина: помилка },
* { статус: 'виконано, значення: 'уявне' },
* ]
**/

// Promise.race vs Promise.any

// Повертає перше встановлене значення
Обіцяйте.раса([rejectAsync,resolveAsync])
// Буде відхилено, оскільки rejectAsync відхиляє через 2500 мс
// і resolveAsync розв’яжеться лише після 4000

// Повертає перше вирішене значення
// Відхиляє, якщо в масиві немає вирішеного значення
Обіцяйте.будь-який([rejectAsync,resolveAsync])
// Вирішується через 4000 мс зі значенням 'imaginary'
```

Це покращує читабельність коду, а також ремонтопридатність програми, але не все ідеально. Оскільки ця функція знаходиться на рівні платформи, кілька реалізацій можуть відрізнятися від поведінки, а також від накладних витрат часу та пам’яті.

Генераторибули представлені в ECMAScript 2015 і є функціями, в яких ми можемо використовувати і керувати ітератором, що означає, що функції можна призупинити та відновити в будь-який час. Це потужний інструмент, коли ми хочемо отримати кожне значення лише тоді, коли нам потрібно, замість того, щоб отримувати всі їх одразу. Це можливо з додаванням словаврожайністьна JavaScript.

На цьому прикладі ми бачимо, що для кожногонаступний()ми отримуємо об’єкт зі значенням і прапорцем, що вказує, чи завершилися функції генератора.

Але генератори також можна використовувати для керування асинхронними потоками в сполученні з іншими бібліотеками, як усоаборедукс-сага, про який я розповім далі.

Як використовувати шаблон Async/Await

Нарешті представлено ES2017асинхронні функціїзначно легше писати та читати асинхронний код у JavaScript!

Вони набагато чистіші, ніж останні обговорювані шаблони, і повернення асинхронної функції є Promise! Це дуже потужно, тому що в нас є доброта обох світів. Як ми вже обговорювали раніше, Promises є безпечним вибором під час роботи зі складними асинхронними операціями, але їх не так легко читати та освоїти, як код асинхронного/очікування.

Один з недоліків полягає в тому, що для нього потрібен інструмент транспіляції, як-от Babel, оскільки Async/Await все ще є синтаксичним цукром над кодом обіцянок.

Оскільки результатом є Promise і його можна вирішити/відхилити, важливо обернути наш код очікування в спробу/перехоплення. Таким чином ми можемо правильно обробляти помилки в нашому асинхронному коді.

```
асинхронний функція() {
 спробуйте {
 констрезультат= чекати отримати("https://imaginaryAPI")
 повернутисярезультат
 } виловити (помилка) {
консоль.помилка(`Помилка:${помилка}`)
 }
}
```

Web-воркери як асинхронні фонові завдання

Використовуючи веб-воркерів, можна запускати сценарії та функції в іншому потокі, запускаючи код в асинхронних фонових завданнях. Це не вплине на зручність використання користувальницького інтерфейсу і може надсилати

дані між працівниками та основним потоком.

Service Worker у наших браузерах активно використовується в прогресивних веб-додатках. Це полягає в реєстрації веб-воркера на нашому веб-сайті та вирішенні, які файли можна кешувати чи ні, і це прискорить використання програми. Крім того, якщо користувач не в мережі, деякі функції все одно будуть доступні. Їх також можна використовувати для виконання важких операцій без заморожування інтерфейсу користувача або основного потоку JS.

Бібліотеки NPM

Кілька інших бібліотек намагаються вирішити ці проблеми, кожна з яких використовує свої методи. Попереду ви можете знайти кілька прикладів:

Асинхронний: ця бібліотека підходить для роботи з зворотними викликами, які намагаються вирішити деякі проблеми, які існують у них, а також усувати проблему пекла зворотного виклику! В останніх реалізаціях також можна використовувати код Async/await.

```
асинхронний.водоспад(
 [зворотний дзвінок1,зворотний дзвінок2],
помилка=>консоль.помилка(`Помилка:${помилка}`)
)
```

Bluebird: дуже ефективна реалізація Promises, яка також включає багато додаткових функцій, таких як скасування, повторення та Promisify! Останній є обгорткою функцій, які працюють із зворотними викликами, повертаючи Promise для цих функцій.

```
констмодуль= вимагати("модуль уявного виклику")
Обіцяйте.pomisifyAll(модуль)

// РЕЗУЛЬТАТ:
// Тепер ми можемо викликати .then() для всіх функцій модуля, так!
```

co: керування асинхронними потоками за допомогою генераторів. Ця бібліотека є середовищем виконання навколо генераторів, що поєднує ключове слово yield з обіцянками, виконує результат генератора та повертає його як об'єкт обіцянки.

```
co(функція* () {
 констант= врожайність увійти(ім'я користувача,пароль)
 повернутисяавт
}).потім(результат=> {
консоль.журнал(результат)
},помилка=> {
консоль.помилка(помилка)
})
```

Redux-сага: Внутрішня бібліотека для стека React/Redux. Це проміжне програмне забезпечення Redux, яке має на меті зробити побічні ефекти додатків більш ефективними та простішими в управлінні, оскільки їх можна запускати або скасовувати за допомогою дій Redux. Ця реалізація інтенсивно використовує генератори для отримання даних через Інтернет і застосування необхідних побічних ефектів на нашому веб-сайті.

```
функція* (ім'я користувача,пароль) {
 спробуйте {
 констант= врожайність дзвонити(увійти,ім'я користувача,пароль)
 врожайність покласти(someActionToStore(авт))
 } виловити (помилка) {
консоль.помилка(`Помилка:${помилка}`)
 }
}
```

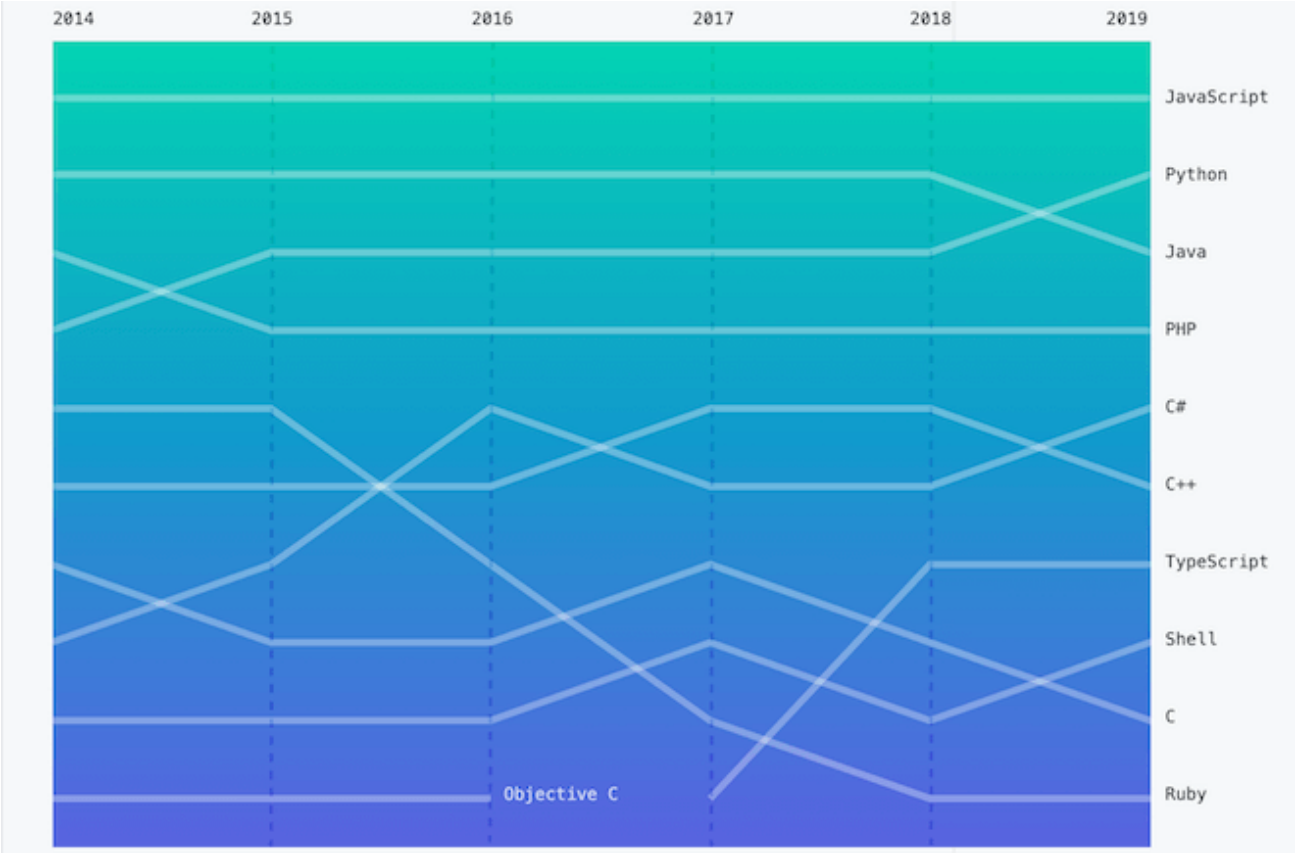
RxJS: це шаблон, який використовується в програмах Angular, і це реактивний шаблон. Ми створюємо спостерігач, на який ми можемо підписатися та чекати змін, про які ми будемо сповіщені. Використовуючи цей шаблон, можна, наприклад, скасувати підписки та ланцюжки спостережуваних.

```
Спостерігається.труба(спочатку()).підписатися(результат=> {
консоль.журнал(`Результат:${результат}`)
},помилка=> {
консоль.помилка(`Помилка:${помилка}`)
})
```

Які асинхронні шаблони ми повинні використовувати?

Для простих проектів зворотні виклики є найпростішим і легшим способом обробки асинхронних потоків. У великих проектах із належним налаштуванням я б вибрав шаблон асинхронність/очікування, оскільки асинхронність легко читається, має природну обробку помилок і немає піраміди смерті.

Це такий синтаксичний цукор, який нам потрібен у нашій роботі, що дозволяє нам писати більш читабельну та зручну програму.



Найпопулярніші мови програмування в GitHub 2014 - 2019 (octoverse.github.com)

Як видно на малюнку вище, JavaScript продовжує залишатися найуживанішою мовою на GitHub разом із його активною спільнотою.

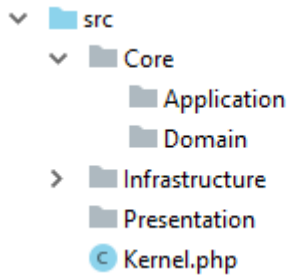
Це наш найкращий вибір для обробки асинхронних потоків, але є інші способи досягнення тих самих результатів, крім тих, які описані в цьому посібнику. Загалом, ви можете вибрати те, що найкраще відповідає вашим потребам.

3.3. Додаткові подулі мовою PHP

Налаштування шарів – сюди входять основні пакети, необхідні для будь-якого загального застосування.

composer create-project symfony/website-skeleton my-project

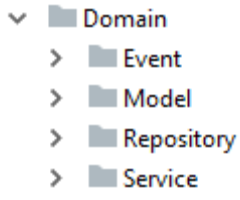
Наступним кроком є зміна структури каталогів, щоб вона нагадувала шари архітектури Onion.



У нас є внутрішні шари, які ми називаємо Core. Тут домен і додаток. Тоді зовнішні шари – це наша інфраструктура та презентація.

Домен

Але давайте глибше розглянемо шар домену.

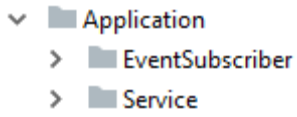


Як бачите, ми розділили це на кілька підкаталогів. Як ви це зробите, вирішувати вам. Це лише деякі основні пропозиції, з яких я починаю. Це може

змінитися залежно від вимог проекту. Я завжди розміщую свої об’єкти домену (або сутності) у каталозі Model. У Repository я розміщую свої інтерфейси репозиторію. Події та сервіс для подій вашого домену та служб домену. Ви можете створити більше каталогів, якщо потрібно.

Застосування

А потім прикладний рівень.



На прикладному рівні ми завжди маємо принаймні деякі сервіси додатків і підписників подій. Але найчастіше вам доведеться створити тут кілька підкаталогів на основі вашого проекту.

доктрина

У doctrine.yaml ми збираємося змінити відображення на XML. Нам також слід створити новий каталог src/config/domain/doctrine для наших файлів XML.

відображення:

додаток:

is_bundle:помилковий

тип: xml

режисер:'%kernel.project_dir%/config/domain/doctrine'

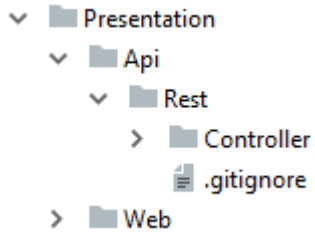
префікс:'App\Core\Domain'

Важливо пам’ятати, що ім’я файлу зіставлення ORM має відповідати структурі каталогів, а назва об’єктів вашого домену починатися з App\Core\Domain\Model.

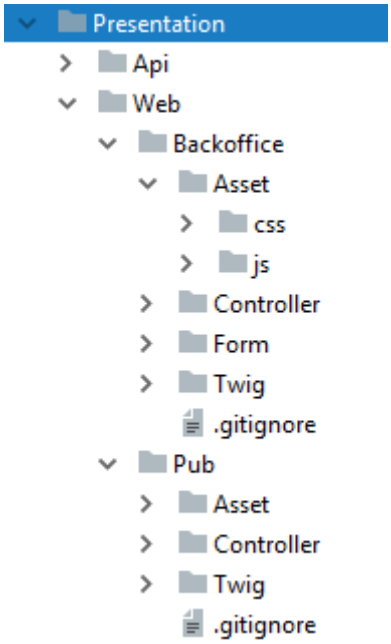
Якщо у вас є об’єкт домену App\Core\Domain\Model\User\User, то файл XML має називатися User.User.orm.xml.

Оскільки наші міграції тепер знаходяться на рівні інфраструктури, нам також потрібно змінити файл doctrine_migrations.yaml.

Для рівня презентації ми створимо API та веб-каталог. У більшості веб-програм, які ви створюєте, вам, ймовірно, знадобляться веб-контролери та/або контролери REST.



Тоді для ваших веб-контролерів ви можете створити бек-офіс і загальнодоступний інтерфейс. У цьому скелеті ми створимо базові налаштування для вирішення цих ситуацій.



Зверніть увагу, що всередині нашого каталогу Backoffice і Pub ми маємо Asset, Controller, Form і Twig.

Всередині Asset ми розміщуємо наші файли js і CSS. Контролер і форма — це наші контролери та форми Symfony. А в каталозі Twig ми розмістимо наші файли шаблонів twig.

Перш ніж ми зможемо використовувати це налаштування, нам потрібно змінити деякі налаштування. Symfony має знати, де знайти контролери та який шлях до них. У services.yaml ми замінюємо конфігурацію контролера за замовчуванням на таке:

Замість використання каталогу шаблонів за замовчуванням для twig. У нас є 2 окремих каталога Twig для веб-сторінок Back office та Public. Ми можемо зробити це за допомогою просторів імен Twig.

У twig.yaml вам потрібно створити таку конфігурацію:

Інтерфейси, які мають реалізувати адаптери у вашій інфраструктурі. Ну, ви розміщуєте їх у домені або додатку. Чи потрібно вашому домену знати про існування інтерфейсного адаптера? Потім створіть інтерфейс у своєму домені. Чи потрібно вашій програмі знати про інфраструктуру? Тоді створіть інтерфейс на рівні програми.

Ніколи не буває правильної чи неправильної відповіді. Це все про те, щоб тримати деякі концепції та ідеї в глибині свого розуму, а потім думати логічно та застосовувати їх.

3.4. Висновки до розділу

Протягом багатьох років у світі з’явилися шаблони та бібліотеки екосистеми js для обробки асинхронного програмування, такого як зворотні виклики, події, обіцянки, генератори, async/await, веб-воркери та пакети в реєстрі NPM, як-от async, bluebird, co або RxJS.

Що стосується розробників інтерфейсу, нам може знадобитися використовувати різні шаблони для вирішення складних щоденних проблем залежно від фреймворку, над яким ми працюємо. Знання доступних інструментів у світі JavaScript дозволяє нам вибрати найкраще рішення для кожної проблеми.

Ви можете запитати, чи цей посібник призначений лише для веб-розробки? Ні, більшість із цих шаблонів використовуються у всіх середовищах і платформах JavaScript, тому знання їхньої роботи завжди цінне для будь-якого розробника.

Тепер повинні побудувати простий каркас архітектури Onion за допомогою Symfony. Можна взяти це за основу для початку розробки.

Ідея цього скелета полягає в тому, щоб дати вам базовий шаблон для початку. Адаптуйте його під свої потреби та проект. І може бути гарною ідеєю подумати про деякі інші ідеї та концепції, які ви будете використовувати.

ВИСНОВКИ

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1.

Додаток А

Лістинг отриманого вихідного коду сторінки