CS251: Tiền điện tử và Công nghệ Blockchain Mùa thu năm 2023

Dự án lập trình số 3

Ứng dụng thanh toán Ethereum

Hạn chót: 11:59 tối Thứ Ba, ngày 31 tháng 10 năm 2023 Nộp qua Gradescope (mỗi câu trả lời trên một trang riêng) mã: 7DVJKY

Trong bài tập này, bạn sẽ sử dụng Solidity và ethers.js để triển khai một ứng dụng phi tập trung phức tạp, hay DApp, trên Ethereum. Bạn sẽ viết cả hợp đồng thông minh và ứng dụng khách ngư ời dùng truy cập hợp đồng thông minh đó, tìm hiểu về quá trình phát triển 'full-stack' của DApp. Để tiết kiệm thời gian, vui lòng đọc toàn bộ bài tập - đặc biệt là phần ghi chú - trư ớc khi bạn bắt đầu phát triển.

1 Blockchain Chia tách

Chúng tôi muốn tạo ra một hệ thống phi tập trung để theo dõi ghi nợ và tín dụng - phiên bản blockchain của Splitwise. Nếu bạn chư a từng nghe đến ứng dụng này, thì đây là cách đơn giản để theo dõi xem ai nợ ai tiền trong một nhóm ngư ời (có thể sau khi chia tiền ăn trư a, tiền tạp hóa hoặc hóa đơn). Để minh họa cho ứng dụng này, hãy xem xét tình huống sau:

Alice, Bob và Carol đều là bạn bè thích đi ăn cùng nhau. Bob đã trả tiền cho bữa trư a lần trư ớc anh ấy và Alice đi ăn, vì vậy Alice nợ Bob 10 đô la. Tư ơng tự, Carol đã trả tiền khi cô ấy và Bob đi ăn, vì vậy Bob nợ Carol 10 đô la.

Bây giờ, hãy tư ởng tư ợng Carol hết tiền mặt và vay 10 đô la từ Alice. Lư u ý rằng tại thời điểm này, thay vì mỗi ngư ời trả lại 'khoản vay' của mình tại một thời điểm nào đó, họ có thể đồng ý rằng không ai nợ ai cả. Nói cách khác, bất cứ khi nào có một chu kỳ nợ, chúng ta có thể xóa nó khỏi sổ sách kế toán của mình, làm cho mọi thứ trở nên đơn giản hơn và giảm số lần tiền mặt cần phải trao tay.

Chúng tôi sẽ xây dựng một cách phi tập trung để theo dõi ai nợ ai cái gì, để không phải dựa vào bên thứ ba đáng tin cậy nào. Nó sẽ hiệu quả: sẽ không tốn một lư ợng gas quá lớn để lư u trữ dữ liệu này. Không có giá trị nào đư ợc chuyển 'trên blockchain' khi sử dụng ứng dụng này; ether duy nhất liên quan sẽ dành cho gas.

Vì nó nằm trên blockchain, khi Carol nhận đư ợc séc cho bữa ăn của cô và Bob, cô có thể yêu cầu Bob nộp một IOU (anh ấy có thể làm điều này bằng DApp của chúng tôi) và cô có thể xác minh rằng anh ấy thực sự đã làm. Kho lư u trữ công khai trên chuỗi sẽ đóng vai trò là nguồn duy nhất để xác định ai nợ ai. Sau đó, khi chu trình minh họa ở trên đư ợc giải quyết, Carol sẽ thấy rằng Bob không còn nợ tiền cô nữa.

Như một phần của việc này, chúng tôi cũng sẽ xây dựng một giao diện ngư ời dùng để tính toán thông tin hữu ích cho ngư ời dùng. và cho phép những ngư ời không phải lập trình viên cũng có thể sử dụng DApp.

2 Bắt đầu

2.1 Thiết lập

1. Cài đặt phần mềm tiên quyết: bạn sẽ cần tải xuống và cài đặt Node.js từ https://nodejs.org/en/. Chọn phiên bản LTS (phiên bản bên trái).
2. Tải xuống và giải nén mã bắt đầu từ trang web của khóa học.
3. cd vào thư mục mã khởi động.
4. Chạy npm install --save-dev hardhat để cài đặt môi trường phát triển Ethereum Hardhat, môi trường mà bạn sẽ sử dụng để mô phỏng một nút Ethereum trên máy cục bộ của mình.
5. Chạy npm install --save-dev @nomiclabs/hardhat-ethers ethers để cài đặt plugin Hardhat mà tập lệnh scripts/deploy.js của bạn để triển khai một nút Hardhat sẽ sử dụng.
6. Chạy npm install ethers@5.4.0 để hạ cấp ethers xuống phiên bản ổn định, đang hoạt động cho việc này dự án.

2.2 Biên dịch, triển khai và kiểm tra

1. Mở thư mục mã khởi động trong IDE hoặc trình soạn thảo văn bản yêu thích của bạn (chẳng hạn như Sublime Text, Atom hoặc Visual Studio Code hoạt động tốt). Bạn sẽ chỉ sửa đổi contract/mycontract.sol để xác định hợp đồng Solidity của mình và web_app/script.js để xây dựng máy khách. Xem các tệp khác có thể giúp hiểu cách hoạt động của nút Hardhat và máy khách web. Có những nơi được đánh dấu bằng các hàm để sửa đổi và bạn có thể thêm các hàm trợ giúp vào web_app/script.js.
   Vui lòng không sửa đổi bất kỳ mã nào khác hoặc cài đặt thêm các gói nút.
2. Xem xét kỹ mã khởi động, ethers.js và tài liệu Solidity [8]. Hãy suy nghĩ cẩn thận về thiết kế tổng thể của hệ thống trước khi bạn viết mã. Dữ liệu nào nên được lưu trữ trên chuỗi? Hợp đồng sẽ thực hiện phép tính nào so với trên máy khách?
3. Sau khi hoàn tất triển khai, hãy chạy npx hardhat node để khởi động node cục bộ. Nếu node được khởi động đúng cách, bạn sẽ thấy trong terminal: Đã khởi động máy chủ HTTP và WebSocket JSON-RPC tại https://localhost:8545, theo sau là 20 tài khoản có khóa riêng tương ứng.
4. Mở một tab hoặc cửa sổ terminal khác. cd vào thư mục mã khởi động.
   Chạy npx hardhat run --network localhost scripts/deploy.js để biên dịch và triển khai hợp đồng của bạn. Khi thành công, hãy mong đợi thấy thông báo này trong terminal của bạn: Đã hoàn tất việc viết địa chỉ hợp đồng: .... Lưu địa chỉ này để sử dụng trong bước tiếp theo.
5. Cập nhật địa chỉ hợp đồng và ABI trong web_app/script.js. ABI có thể được sao chép từ tệp tự động tạo ra artifacts/contracts/
   mycontract.sol/Splitwise.json. Để cập nhật ABI chính xác, vui lòng sao chép toàn bộ danh sách sau trường 'abi', bắt đầu từ dấu ngoặc vuông. Địa chỉ được lưu trong bước trước. Đảm bảo địa chỉ hợp đồng của bạn là một chuỗi.
6. Mở tệp web_app/index.html trong trình duyệt của bạn. Bạn có thể truy cập trang web được tạo bởi web app/index.html bằng cách mở trực tiếp, kéo vào tab trình duyệt hoặc sử dụng

Plug-in máy chủ trực tiếp VS Code. Bạn có thể chơi với hệ thống của mình thông qua giao diện ngư ời dùng hoặc chạy kiểm tra tính hợp lý của chúng tôi! Để biết thêm thông tin, hãy xem phần 7.2 Phát triển thực tế & Gỡ lỗi.

Lư u ý về hệ điều hành: Tất cả các bư ớc trên đều có thể thực hiện trên hệ thống Unix và Windows. Các lệnh chúng tôi yêu cầu bạn thực hiện sẽ hoạt động trong thiết bị đầu cuối Unix chuẩn và Dấu nhắc lệnh Windows.

3 Thành phần

Dự án có hai thành phần chính: một hợp đồng thông minh, đư ợc viết bằng Solidity và chạy trên blockchain, và một máy khách chạy cục bộ trong trình duyệt web, quan sát blockchain bằng thư viện ethers.js và có thể gọi các hàm trong hợp đồng thông minh. Để biết thêm thông tin về cách ethers.js hoạt động và thiết lập bài tập này, hãy xem bản ghi phần 3 trên panopto.

3.1 Chức năng trong máy khách

Xin lư u ý, tất cả các hàm máy khách mà chúng tôi yêu cầu bạn triển khai đều đư ợc cung cấp dư ới dạng hàm bất đồng bộ trong mã khởi động. Điều này có nghĩa là chúng trả về một Promise theo mặc định. Hệ thống chấm điểm của chúng tôi sẽ giả định rằng một Promise đư ợc trả về từ mỗi hàm máy khách. Để biết thêm về các hàm bất đồng bộ, Promise và await, hãy xem tài liệu tham khảo ở cuối tài liệu này.

1. getUsers(): Trả về một Promise cho một danh sách các địa chỉ. Danh sách sẽ là địa chỉ của 'tất cả những ngư ời đã từng gửi hoặc nhận IOU'. Bạn có thể thấy điều này hữu ích như một trợ giúp cho các hàm khác.

2. getTotalOwed(user): Trả về một Promise cho tổng số tiền mà ngư ời dùng đã cho nợ.

3. getLastActive(user): Trả về Promise cho dấu thời gian UNIX (giây kể từ ngày 1 tháng 1 năm 1970) của hoạt động đư ợc ghi lại gần đây nhất của ngư ời dùng này (gửi IOU hoặc đư ợc liệt kê là 'chủ nợ' trên IOU). Trả về null nếu không tìm thấy hoạt động nào.

4. add_IOU(chủ nợ, số tiền): Nộp một IOU cho hợp đồng, với chủ nợ đã chuyển như ợng và số lư ợng. Xem lư u ý về cách giải quyết vòng lặp bên dư ới.

3.2 Chức năng trong hợp đồng

1. lookup(địa chỉ con nợ, địa chỉ chủ nợ) chế độ xem công khai trả về (uint32 ret): Trả về số tiền mà con nợ nợ chủ nợ.

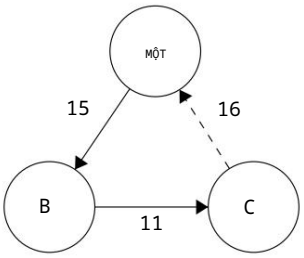
2. add_IOU(address creditor, uint32 amount, ...): Thông báo cho hợp đồng rằng msg.sender hiện nợ chủ nợ nhiều đô la hơn. Đây là phép cộng: nếu bạn đã nợ tiền, phép cộng này sẽ cộng vào số tiền đó. Số tiền phải là số dư ơng. Bạn có thể làm cho hàm này lấy bất kỳ số lư ợng đối số bổ sung nào miễn là hai đối số đầu tiên là creditor và amount. Xem lư u ý về việc giải quyết vòng lặp bên dư ới.

Bạn đư ợc chào đón để viết thêm trợ lý cho khách hàng hoặc hợp đồng. Khách hàng có thể gọi các hàm hợp đồng với BlockchainSplitwise.functionName(arguments) và

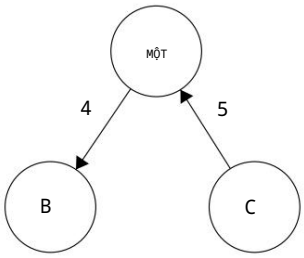
BlockchainSplitwise.connect(anotherSigner).functionName(arguments). Vui lòng xem tài liệu hư ớng dẫn tại đây để biết cách tư ơng tác với các hàm hợp đồng Solidity của bạn từ máy khách bằng ethers.js. Hãy đảm bảo bạn biết sự khác biệt giữa hai phư ơng pháp đó. Hãy nhớ rằng các hàm máy khách sẽ đư ợc viết bằng JavaScript và các hàm hợp đồng sẽ đư ợc viết bằng Solidity.

4 Giải quyết vòng lặp nợ nần

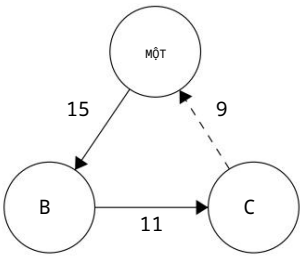
Sẽ hữu ích khi nghĩ về các IOU như một biểu đồ nợ. Nghĩa là, giả sử mỗi ngư ời dùng là một nút và mỗi cạnh có hư ớng có trọng số từ A đến B với trọng số X biểu thị thực tế 'A nợ B $X'. Chúng ta sẽ viết điều này là A X B. Chúng ta muốn ứng dụng của mình 'giải quyết' bất kỳ chu kỳ nào trong biểu đồ này bằng cách trừ giá trị tối thiểu của tất cả các trọng số trong chu kỳ khỏi mọi bư ớc trong chu kỳ (do đó làm cho ít nhất một bư ớc trong chu kỳ có trọng số '0'). 15 Ví dụ, nếu A B và B 4 11 khi C cộng C 16 A, số dư thực tế sẽ là C, 0 đư ợc cập nhật để phản ánh rằng A B, B C và C 5 A.



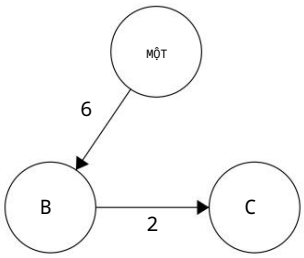
trở thành. . .



Tư ơng tự, nếu C tiến hành cộng C B, 2 9 A, số dư thực tế sẽ đư ợc cập nhật để phản ánh rằng A 6 0 B C, và C A.



trở thành. . .



Yêu cầu là nếu bất kỳ chu kỳ tiềm năng nào đư ợc hình thành khi bạn sắp thêm một IOU bằng cách sử dụng máy khách (add_IOU), bạn phải 'giải quyết' ít nhất một trong số chúng. Bạn không cần phải lo lắng về các trư ờng hợp phức tạp liên quan đến nhiều vòng lặp hoặc tối ư u hóa đư ờng dẫn nào để thực hiện (một cái gì đó giống như luồng tối đa) trong các trư ờng hợp đó. Bạn có thể giả định rằng như một điều kiện tiên quyết cho cả hai hàm hợp đồng (add_IOU và lookup), không có chu kỳ nào trong biểu đồ. Cuối cùng, bạn cũng có thể giả định rằng bất kỳ chu kỳ nào đư ợc tìm thấy sẽ hơi nhỏ (ví dụ, nhỏ hơn 10).

Chúng tôi cung cấp cho bạn thuật toán tìm kiếm theo chiều rộng trong mã - để sử dụng thuật toán này, hãy truyền vào một nút bắt đầu và kết thúc, và một hàm để lấy 'hàng xóm' của bất kỳ nút nào đã cho. Bạn cũng có thể không sử dụng triển khai này.

Việc thực hiện giải pháp này một cách an toàn là tùy thuộc vào bạn. Không thể có khả năng cho một kẻ xấu khách hàng bằng cách nào đó 'xóa bỏ' khoản nợ của mình sau khi đã ghi sổ.

Bây giờ chúng ta có thể minh họa chính xác cách bạn có thể trả lại một IOU trong hệ thống này. Giả sử Alice vay Bob 10 đô la; bây giờ, cô ấy muốn trả lại Bob bằng tiền mặt. Khi Alice đưa Bob 10 đô la tiền mặt, Bob sẽ thêm một IOU là 10 đô la với chủ nợ là Alice. Điều này sẽ tạo ra một chu kỳ: cụ thể là, A 10 B 0 0 cầu về độ 10 và B A. Theo yêu phân giải chu trình ở trên, chu trình này sẽ kết thúc bằng A B và B A.

5 Yêu cầu chung

Bạn có thể viết hợp đồng theo bất kỳ cách nào bạn thích, miễn là nó có các hàm lookup và add_IOU được chỉ định. Mục tiêu của bạn là viết một hợp đồng giảm thiểu lượng lưu trữ và tính toán được sử dụng bởi cả hai hàm hợp đồng. Điều này sẽ giảm thiểu chi phí gas.

Bạn có thể cho rằng khối lượng giao dịch đủ nhỏ để có thể tìm kiếm toàn bộ blockchain trên máy khách, nhưng bạn không nên cho rằng những người dùng duy nhất là những người dùng trong ví của bạn - nói cách khác, provider.listAccounts() không chứa mọi người dùng có thể có của hệ thống.

6 Gửi mã của bạn

Chúng tôi sẽ sử dụng Gradescope để nộp bài. Vui lòng CHỈ nộp tệp mycontract.sol và script.js của bạn! Bài nộp của bạn sẽ được chấm điểm dựa trên việc trả lời đúng các câu hỏi, có phát sinh một lượng gas và bảo mật hợp đồng hợp lý hay không.

7 Ghi chú

Chúng tôi sẽ đăng danh sách cập nhật tất cả các giải thích và lời khuyên về Ed. Vui lòng theo dõi bài đăng đó để biết thông tin mới nhất khi chúng tôi giải quyết mọi vấn đề với bài tập.

7.1 Kiến trúc hệ thống

- Trước tiên, bạn nên quyết định cấu trúc dữ liệu nào sẽ được lưu trữ trên blockchain. Hãy suy nghĩ cẩn thận về thông tin bạn cần cung cấp cho khách hàng. Bạn không cần sử dụng bất kỳ cấu trúc dữ liệu đặc biệt nào. Quyết định của bạn có thể khiến việc triển khai trở nên khó khăn hơn, vì vậy bạn nên quay lại và thay đổi kiến trúc của mình.

- Chúng tôi chưa đề cập đến việc phải làm gì trong trường hợp một chu kỳ hình thành chỉ giữa hai người.
  Chúng tôi khuyên bạn nên thiết kế hệ thống của mình sao cho đây không phải là trường hợp đặc biệt - khi con nợ đã 'trả nợ' cho chủ nợ, chủ nợ chỉ cần cố gắng thêm một IOU theo hướng ngược lại, kích hoạt giải quyết chu kỳ và kết thúc bằng việc cả hai đều nợ nhau 0. Chúng tôi cũng khuyên bạn nên tránh bất kỳ khái niệm nào về nợ 'âm', vì điều này có thể làm mọi thứ trở nên phức tạp quá mức.

- Hãy nhớ rằng khi tối ưu hóa chi phí gas, các chức năng chạy trên máy khách là miễn phí - chúng phải chịu không mất phí.

- Chúng tôi đề xuất rằng sau khi thiết kế hệ thống của bạn, bạn bắt đầu bằng cách viết và gỡ lỗi kỹ lưỡng hợp đồng.

- Bạn không cần một lượng lớn mã để hoàn thành bài tập. Giải pháp của chúng tôi chỉ gồm khoảng 40 dòng Solidity và khoảng 70 dòng JavaScript mới (không bao gồm ABI).

7.2 Phát triển thực tế & Gỡ lỗi

- Để gỡ lỗi mã phía máy khách, hãy sử dụng console.log một cách thoải mái. Bạn sẽ thấy kết quả của các cuộc gọi và số dòng mà chúng bắt nguồn từ trong bảng điều khiển JavaScript của trình duyệt.
- Có thể bỏ qua các cảnh báo về XMLHttpRequest đồng bộ.
- Solidity có một chức năng rất hữu ích require cho phép bạn kiểm tra các điều kiện tiên quyết
- Trình tự động chấm điểm sẽ cho rằng các hàm của máy khách trả về Lời hứa cho một số giá trị khác. Để biết thêm thông tin về lời hứa và mã không đồng bộ, vui lòng xem các tài liệu tham khảo ở cuối tài liệu này. Chúng tôi cũng đã cung cấp một hàm kiểm tra tính hợp lệ để bạn có thể hiểu cách chúng tôi sẽ kiểm tra mã của bạn. Bạn có thể gọi hàm này từ bảng điều khiển trình duyệt hoặc bỏ chú thích lệnh gọi hàm này trong script.js. Bạn nên triển khai lại hợp đồng trước khi chạy hàm kiểm tra tính hợp lệ để đặt lại hệ thống về trạng thái ban đầu. Bạn được khuyến khích viết các bài kiểm tra khác, nhưng vui lòng đảm bảo mã của bạn vượt qua được bài kiểm tra tính hợp lệ được cung cấp trước khi gửi.
- Bộ giải mã ABI sẽ giải mã các đầu vào hàm ở dạng chữ thường. Để giải quyết vấn đề này, mã khởi động sẽ cố gắng trả về tất cả các giá trị dưới dạng phiên bản chữ thường của chúng bằng cách sử dụng hàm toLowerCase(). Hãy nhớ giữ nguyên các giá trị chữ hoa và chữ thường. Mỗi hàm sẽ hoạt động bất kể chữ hoa hay chữ thường của các ký tự chữ cái được truyền vào như một phần của giá trị thập lục phân.
- Xin lưu ý rằng chúng tôi đang sử dụng Solidity phiên bản 0.8.17 nên không dựa vào tài liệu hướng dẫn cho phiên bản cũ hơn vì chúng có thể hoạt động khác nhau.

8 Tài liệu tham khảo

- Bạn có thể đọc về cách mở bảng điều khiển JavaScript của trình duyệt tại đây.
- Hướng dẫn về async/await trong Javascript tại đây.
- Hướng dẫn về Promises trong Javascript tại đây.
- Có hướng dẫn hữu ích để hiểu về Hardhat tại đây.
- Tài liệu về ethers.js có tại đây.
- Tài liệu Solidity v0.8.17 có tại đây.